**BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH**

**Lab 11 : Lập trình xử lý ngắt**

|  |  |
| --- | --- |
| **Họ tên** | **MSSV** |
| Phạm Minh Hiển | 20235705 |

**Assignment 4:**

Tạo project để thực hiện và thử nghiệm Home Assignment 4. Chạy ở chế độ từng dòng lệnh, quan sát giá trị của các thanh ghi để hiểu cách chương trình hoạt động.

* Kết quả:

A screenshot of a computer program

AI-generated content may be incorrect.

+ Ngắt do thời gian (UTI).

A screenshot of a computer program

AI-generated content may be incorrect.

+ Ngắt ngoài (UEI).

* Chạy từng dòng lệnh:

+ **la** t0, handler: t0 = 0x0040004c

+ **csrrs** zero, utvec, t0: utvec = 0x0040004c

+ t1 = 0x00000100 (8)

+ **csrrs** zero, uie, t1: uie = t1 = 8

+ **csrrsi** zero, uie, 0x10: uie = 0x00000110

+ ustatus = 0x00000001

+ t1 = 0xffff0012

+ t2 = 0x80

+ **li** t1, 0xffff0020: t1 = 0xffff0020

+ **li** t2, 1000: t2 = 0x000003e8

+ *loop* : chờ ngắt xảy ra

+ Khi ngắt xảy ra chương trình nhảy đến *handler*

+ Dành 16 byte trên stack để lưu các thanh ghi a0, a1, a2, a7 (lưu trữ ngữ cảnh).

+ **csrr** a1, ucause / **li** a2, 0x7FFFFFFF / **and** a1, a1, a2 :Đọc ucause để xác định nguyên nhân ngắt. Mask bit cao nhất (để kiểm tra loại ngắt).

+ **li** a2, MASK\_CAUSE\_TIMER / **beq** a1, a2, timer\_isr :Nếu ucause (a1) == 4 (ngắt timer), nhảy đến *timer\_isr*, in thông báo "Time inteval!"

+ Đọc giá trị timer hiện tại (TIMER\_NOW), cộng thêm 1000, và ghi vào TIMER\_CMP để thiết lập ngắt tiếp theo.

+ **li** a2, MASK\_CAUSE\_KEYPAD / **beq** a1, a2, keypad\_isr: Nếu ucause (a1) == 8 (ngắt bàn phím), nhảy đến *keypad\_isr,* In thông báo "Someone has pressed a key!”.

+ Hôi phục các thanh ghi từ stack.

+ Trở về chương trình chính bằng uret.

**Assignment 5:**

Tạo project để thực hiện và thử nghiệm Home Assignment 5. Chạy ở chế độ từng dòng lệnh, quan sát giá trị của các thanh ghi để hiểu cách chương trình hoạt động.

A screenshot of a computer program

AI-generated content may be incorrect.

* Kết quả:

A computer screen shot of a computer

AI-generated content may be incorrect.

* Chạy từng dòng lệnh:

+ **csrrw** zero, utvec, t0: utvec = 0x0040001c.

+ **csrrsi** zero, ustatus, 1: ustatus = 0x00000001.

+ **lw** zero, 0: ustatus = 0x00000010, chương trình nhảy đến địa chỉ trong utvec (catch).

+ In ra “Exception occurred.”

+ **la** và **csrrw** zero, uepc, t0: nạp địa chỉ *finally* vào uepc (ban đầu chứa địa chỉ lệnh gây lỗi **lw** zero, 0), uepc = 0x00400014.

+ **uret** : ustatus = 0x00000001 và nhảy đến *finally*.

**Assigment 6:**

Ngắt mềm có thể được kích hoạt bằng cách thiết lập bit USIP (bit 0) của thanh ghi uip (trước đó cần cho phép ngắt mềm bằng cách thiết lập bit USIE của thanh ghi uie). Viết chương trình kích hoạt ngắt mềm nếu xảy ra tràn số khi thực hiện việc cộng 2 số nguyên có dấu (xem lại Bài thực hành 4), chương trình con xử lý ngắt sẽ in ra thông báo lỗi tràn số và kết thúc chương trình.

**Source Code**:

.data

newline: .asciz "\n"

overflow\_msg: .asciz "Overflow error occurred!\n"

result\_msg: .asciz "The result is: "

num1\_msg: .asciz "First number: "

num2\_msg: .asciz "Second number: "

.text

main:

.data

newline: .asciz "\n"

overflow\_msg: .asciz "Overflow error occurred!\n"

result\_msg: .asciz "The result is: "

num1\_msg: .asciz "First number: "

num2\_msg: .asciz "Second number: "

.text

main:

li sp, 0x7FFFFFF0 # Con tro ngan xep

la t0, handler

csrrw zero, utvec, t0

csrrsi zero, uie, 1 # Kich hoat ngat mem

csrrsi zero, ustatus, 1 # Kich hoat ngat

li t0, 0x7fffffff # So nguyen thu 1

li t1, 1 # So nguyen thu 2

# In so thu nhat

li a7, 4

la a0, num1\_msg

ecall

li a7, 1

add a0, x0, t0

ecall

li a7, 4

la a0, newline

ecall

# In so thu hai

li a7, 4

la a0, num2\_msg

ecall

li a7, 1

add a0, x0, t1

ecall

li a7, 4

la a0, newline

ecall

add t2, t0, t1 # t0 + t1

srli t3, t0, 31 # bit 31 cua t0

srli t4, t1, 31 # bit 31 cua t1

srli t5, t2, 31 # bit 31 cua t2

li a7, 1 # In bit 31

add a0, x0, t3

ecall

li a7, 4

la a0, newline

ecall

li a7, 1

add a0, x0, t4

ecall

li a7, 4

la a0, newline

ecall

li a7, 1

add a0, x0, t5

ecall

li a7, 4

la a0, newline

ecall

bne t3, t4, no\_overflow # Neu bit 31 cua t0 va t1 khac nhau thi khong tran so

bne t3, t5, overflow # Neu bit 31 cua t0 va t2 khac nhau thi tran so

no\_overflow:

li a7, 4 # In ket qua neu khong tran so

la a0, result\_msg

ecall

li a7, 1

add a0, x0, t2

ecall

li a7, 4

la a0, newline

ecall

j exit\_program

overflow:

csrrsi zero, uip, 1 # Kich hoat ngat mem

j exit\_program

handler:

addi sp, sp, -16 # Luu boi canh

sw a0, 0(sp)

sw a7, 4(sp)

li a7, 4 # In thong bao loi

la a0, overflow\_msg

ecall

lw a7, 4(sp) # Khoi phuc va tro ve

lw a0, 0(sp)

addi sp, sp, 16

uret

exit\_program:

li a7, 10

ecall

* Kết quả chạy:

A screenshot of a computer

AI-generated content may be incorrect.

+ Với t0 = 231 – 1, t1 = 1: Xảy ra tràn số.

A screenshot of a computer program

AI-generated content may be incorrect.

+ Với t0 = 5, t1 = 1: Không xảy ra tràn số.

* Cách chương trình hoạt động:

+ Ta chọn 2 số nguyên rồi gán vào t0, t1.

+ Chương trình sẽ in ra bit dấu ( bit 31) của t0, t1 và t2 với t2 = t0 + t1 để kiểm tra.

+ Nếu t0 và t1 khác dấu thì không tràn số, chương trình sẽ in ra kết quả phép cộng và kết thúc chương trình.

+ Nếu t0 và t2 khác dấu thì tràn số, chương trình sẽ in ra thông báo tràn số và kết thúc chương trình.

**Additional Assigment:**

- Sử dụng ngắt từ bộ định thời, đếm và hiển thị các giá trị từ 00 - 99 trên 2 đèn LED 7 đoạn với chu kỳ mặc định là 1 giây.

- Kết hợp với ngắt từ keypad để:

    - Nhấn nút 0 thì đếm tăng dần (tăng đến 99 thì quay về 00)

    - Nhấn nút 1 thì đếm giảm dần (giảm đến 00 thì quay về 99)

    - Nhấn nút 2 thì giảm tốc độ đếm (tăng chu kỳ)

    - Nhấn nút 3 thì tăng tốc độ đếm (giảm chu kỳ)

**Source Code:**

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.eqv SEVENSEG\_LEFT 0xFFFF0011

.eqv SEVENSEG\_RIGHT 0xFFFF0010

.eqv TIMER\_NOW 0xFFFF0018

.eqv TIMER\_CMP 0xFFFF0020

.eqv MASK\_CAUSE\_TIMER 4

.eqv MASK\_CAUSE\_KEYPAD 8

.eqv SPEED 1000

.data

LED: .word 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71

Mode: .byte 0x11, 0x21, 0x41, 0x81

.text

main:

la t0, handler

csrrs zero, utvec, t0

li t1, 0x100

csrrs zero, uie, t1 # uie - ueie bit (bit ? - external interrupt

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

csrrsi zero, ustatus, 1 # ustatus - enable uie - global interrupt

li s11, 100 # Gioi han tren bo dem

li s10, 10 # Hang so

li s9, 4 # 4 bytes

la s8, LED

li s3, 2 # He so tang giam SPEED

li s2, 9000 # Gioi han SPEED

# Enable the interrupt of keypad of Digital Lab Sim

li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li t2, 0x81 # bit 7 of = 1 to enable interrupt

sb t2, 0(t1)

# Enable the timer interrupt

li t6, SPEED

li t1, TIMER\_CMP

li t2, SPEED

sw t2, 0(t1)

loop:

nop

nop

nop

j loop

end\_main:

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:

j count

timer\_update:

# Set cmp to time + 1000

li a0, TIMER\_NOW

lw a1, 0(a0)

add a1, a1, t6 # Them SPEED de tinh thoi gisn ngat ke tiep

li a0, TIMER\_CMP

sw a1, 0(a0) # Ghi thoi gian moi de dinh thoi ngat ke tiep

j end\_process

keypad\_isr:

li a4, IN\_ADDRESS\_HEXA\_KEYBOARD

li a3, 0x81 # bit 7 of = 1 to enable interrupt

sb a3, 0(a4)

li a5, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb s6, 0(a5) # Doc phim nhan

la a4, Mode

lb s5, 0(a4) # So sanh voi mode[0] (0x11)

bne s6, s5, check # Neu khong phai 0x11

addi s4, zero, 0 # Dem tang

j end\_process

check:

lb s5, 1(a4) # mode[1] = 0x21

bne s6, s5, second\_check # Phim khac 1 thi kiem tra second\_check

addi s4, zero, 1 # Dem giam

j end\_process

second\_check:

lb s5, 2(a4) # mode[2] = 0x41

bne s6, s5, down\_speed # Neu la 0x41 thi giam toc do

j up\_speed # Neu la 0x81 thi tang toc do

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

count:

beq s4, zero, up

down:

addi t5, t5, -1

bne t5, zero, show

addi t5, zero, 100

j show

up:

addi t5, t5, 1

bne t5, s11, show

addi t5, zero, -1

show:

rem t4, t5, s10 # Hang don vi

mul t4, t4, s9

add t3, s8, t4

lw t4, 0(t3)

jal SHOW\_7SEG\_RIGHT

div t4, t5, s10 # Hang chuc

rem t4, t4, s10

mul t4, t4, s9

add t3, s8, t4

lw t4, 0(t3)

jal SHOW\_7SEG\_LEFT

j timer\_update

SHOW\_7SEG\_LEFT:

li s7, SEVENSEG\_LEFT # assign port's address

sb t4, 0(s7) # assign new value

jr ra

SHOW\_7SEG\_RIGHT:

li s7, SEVENSEG\_RIGHT # assign port's address

sb t4, 0(s7) # assign new value

jr ra

up\_speed:

bgt t6, s2, end\_process

mul t6, t6, s3

j end\_process

down\_speed:

blt t6, s11, end\_process

div t6, t6, s3

j end\_process

* Kết quả:

+ Chương trình khi kết nối với Digital Lab Sim và Timer Tool thì hoạt động như kỳ vọng.

+ Bộ đếm ban đầu ở trạng thái đếm tăng từ 00 đến 99 rồi về 00, chu kỳ 1s.

+ Nếu nhấn phím 1 thì sẽ đếm ngược lại tại thời điểm hiện tại.

+ Nhấn phím 2 thì sẽ giảm tốc độ đếm tối đa 100ms và phím 3 thì tăng tốc độ đếm tối đa 9s.

**KẾT LUẬN:**

* Kỹ thuật thăm dò (Polling) là phương pháp mà CPU liên tục kiểm tra (thăm dò) trạng thái của thiết bị ngoại vi (ví dụ: bàn phím, chuột...) để xem thiết bị đó có cần phục vụ không.
* Ngắt (Interrupt) là tín hiệu từ thiết bị ngoại vi hoặc phần mềm gửi đến CPU để yêu cầu xử lý một sự kiện ngay lập tức, làm tạm dừng chương trình đang chạy và chuyển sang xử lý sự kiện đó.
* Chương trình con xử lý ngắt là đoạn mã được thực thi khi có một ngắt xảy ra. Nó thực hiện các hành động cần thiết để xử lý sự kiện do ngắt gây ra.
* Ưu điểm của kỹ thuật thăm dò:

+ Dễ triển khai và đơn giản trong thiết kế.

+ Không yêu cầu phần cứng phức tạp để quản lý ngắt.

* Ưu điểm của kỹ thuật xử lý ngắt:

+ **Hiệu quả cao:** CPU không phải lãng phí thời gian kiểm tra thiết bị liên tục.

+ **Phản hồi nhanh:** Xử lý sự kiện gần như ngay lập tức khi nó xảy ra.

+ **Tối ưu tài nguyên:** CPU có thể làm việc khác cho đến khi có sự kiện xảy ra.

* Điểm khác nhau giữa Ngắt, Ngoại lệ và Traps:

|  |  |  |
| --- | --- | --- |
|  | Nguồn gốc | Mục đích chính |
| Ngắt | Từ bên ngoài CPU (thiết bị ngoại vi) | Phản hồi sự kiện từ thiết bị ngoại vi |
| Ngoại lệ | Từ bên trong CPU (lỗi chương trình) | Xử lý lỗi hoặc điều kiện đặc biệt khi thực thi |
| Trap | Từ chương trình (do phần mềm tạo ra) | Gọi dịch vụ hệ điều hành một cách có kiểm soát |