

處理器設計與實作

實習講義

編撰者

成大電通所計算機架構與系統研究室CASLAB

國立成功大學電機系與電腦與通信工程研究所

LAB 4: RISC-V CPU Interrupt handler and ISR

中斷處理程序

大綱

1. Interrupt 簡介
2. RISC-V Interrupt
 - RISC-V Interrupt Processing
3. 實驗: 實踐 Interrupt handler 與 ISR
 - Interrupt Handler
 - Interrupt Service Routine
 - 驗證結果

實驗目的

1. 認識RISCV-V的interrupt機制與RISC-V的interrupt處理流程
2. 實作簡易interrupt handler and ISR (Interrupt Service Routine)

Interrupt 簡介

Interrupt (1/2)

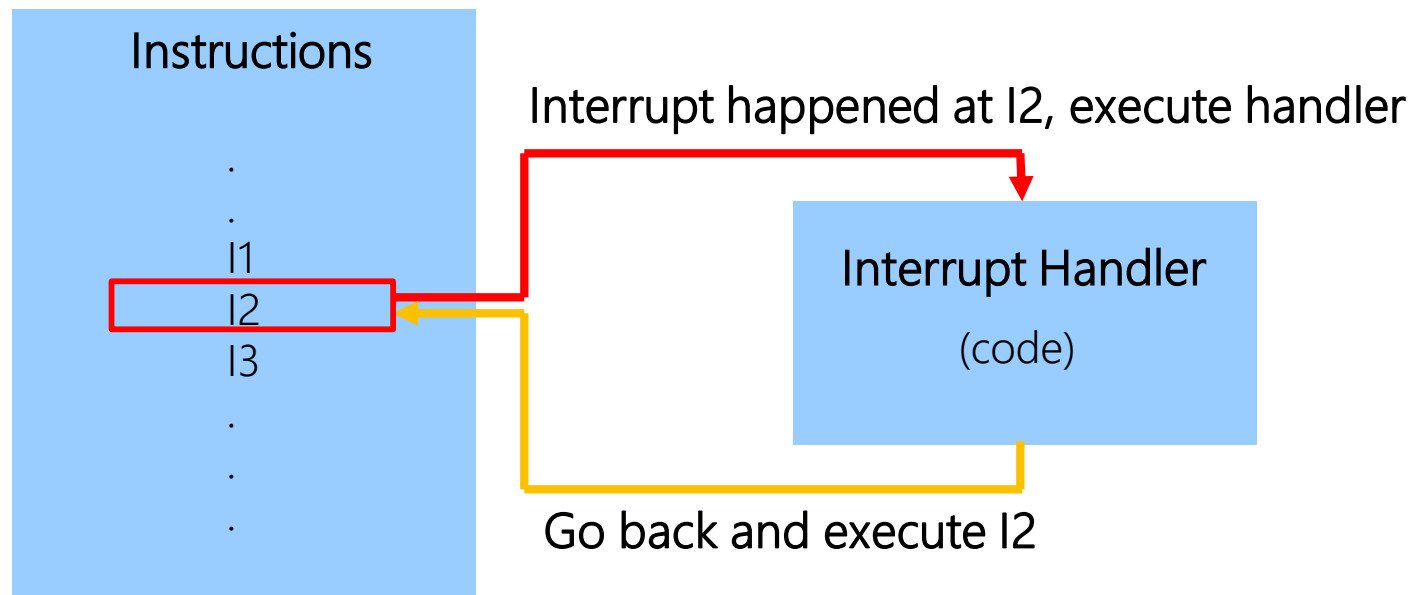
- ⊕ An unscheduled event that disrupts program execution.
 - CPU need to handle it first.
 - It **needs immediate attention**.
 - Emitted by hardware or software.

- ⊕ There are two classes of causes for an interrupt.
 - **Internal Interrupt:** Exception like overflow or illegal operation
 - **External Interrupt:** I/O Device Complete or error

Interrupt (2/2)

⊕ Interrupt Handler

- CPU execute interrupt handler when interrupt happened.
- It'll provide corresponding service for different kinds of interrupt, which called Interrupt Service Routine (ISR).



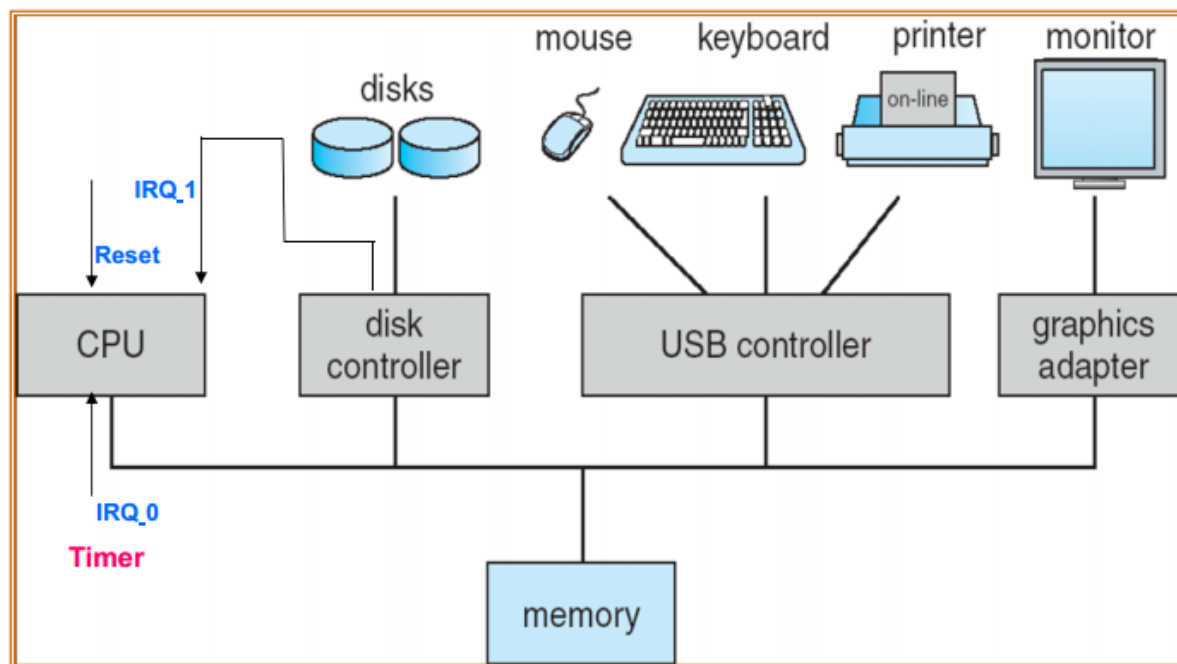
Internal Interrupt

⊕ Interrupts caused by instruction

- Address Error
 - Load from an illegal address
 - Store to an illegal address
- Reserved Instruction (illegal instructions)
 - The operation is not exist
- Arithmetic Overflow
- Floating Point Error

External Interrupt

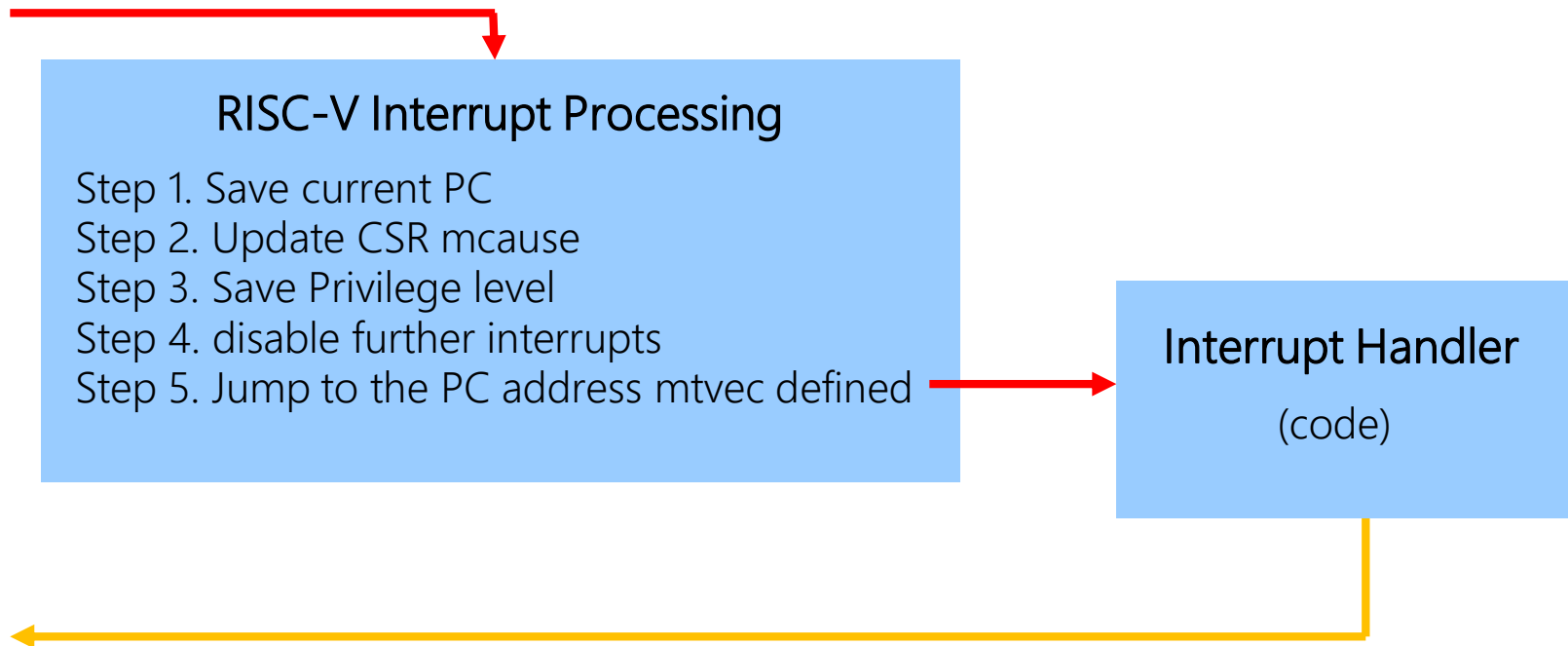
- ⊕ Interrupts generated by external devices.
 - Usually, devices send interrupt signal via external pins.
 - Interrupt ReQuest (IRQ) Lines
 - Enable it by setting appropriate bits in Status register.



RISC-V Interrupt

RISC-V Interrupt Processing

- ⊕ The following steps are **implemented by the processor hardware**, not by software



Step 1. Save current PC

- ⊕ 將當前指令的PC值存到 mepc 暫存器中
- ⊕ **mepc** (Machine Exception Program Counter)
 - 在進入interrupt時，硬體自動更新mepc register的值為遇到interrupt時該指令的PC值
 - mepc裡存的PC值會做為Interrupt結束後的返回地址，回到之前程序的停止執行點

Step 2. Update CSR mcause

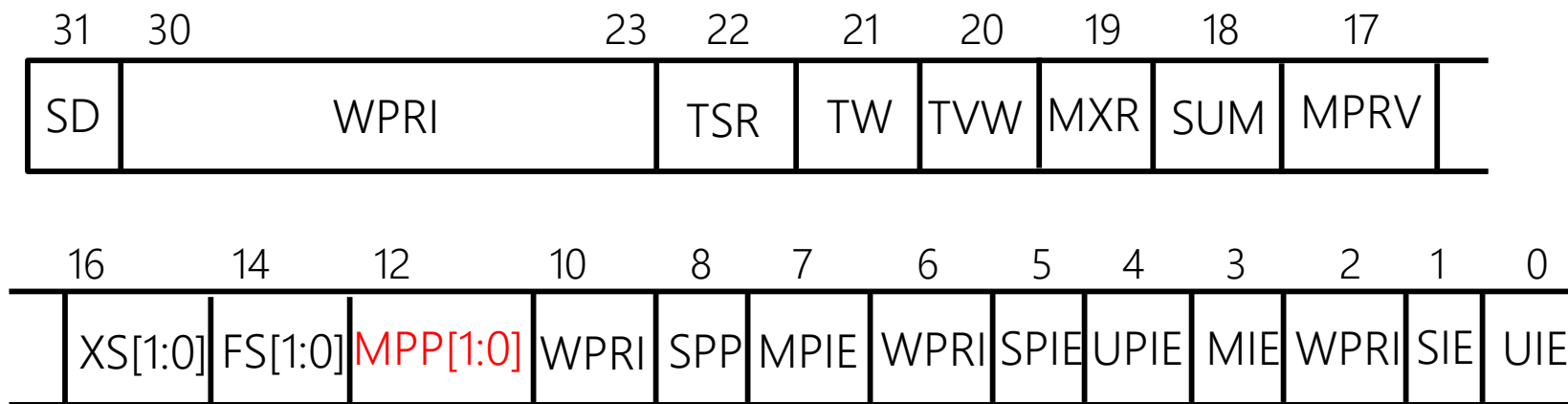
- 將Interrupt的種類寫入mcause暫存器中
- 軟體可透過此暫存器得知Interrupt發生的具體原因

Interrupt = 1 (interrupt)	
Exception Code	Description
0	User Software Interrupt
1	Supervisor Software Interrupt
2	Reserved
3	Machine Software Interrupt
4	User Timer Interrupt
5	Supervisor Timer Interrupt
6	<i>Reserved</i>
7	Machine Timer Interrupt
8	User External Interrupt
9	Supervisor External Interrupt
10	<i>Reserved</i>
11	Machine External Interrupt
12 - 15	<i>Reserved</i>
≥16	Local Interrupt X

Interrupt = 0 (exception)	
Exception Code	Description
0	Instruction Address Misaligned
1	Instruction Access Fault
2	Illegal Instruction
3	Breakpoint
4	Load Address Misaligned
5	Load Access Fault
6	Store/AMO Address Misaligned
7	Store/AMO Access Fault
8	Environment Call from U-mode
9	Environment Call from S-mode
10	Reserved
11	Environment Call from M-mode
12	Instruction Page Fault
13	Load Page Fault
14	Reserved
15	Store/AMO Page Fault
≥16	Reserved

Step 3. Save Privilege level

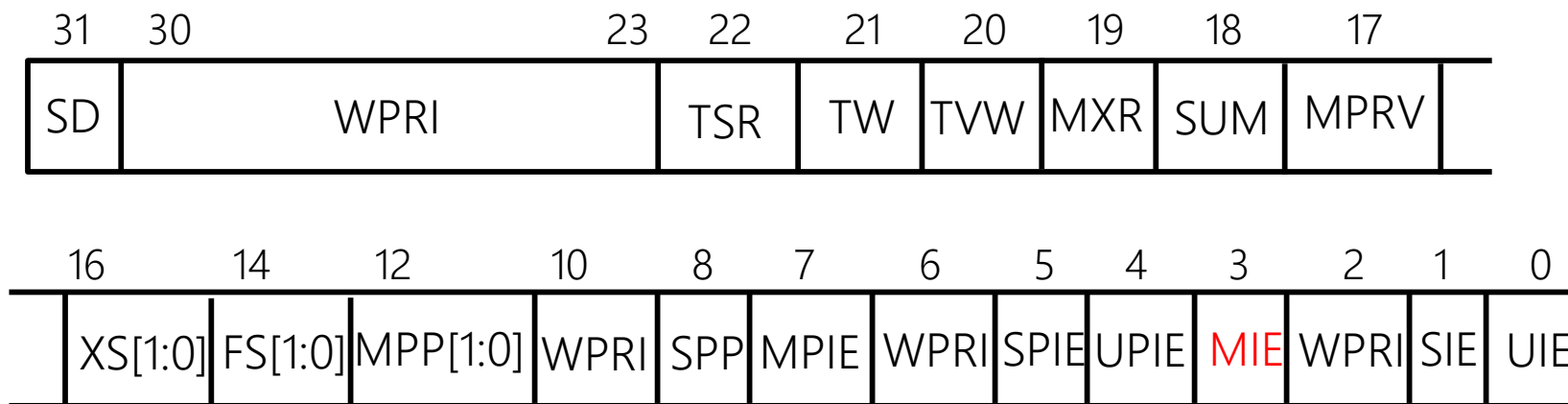
- ⊕ 將異常發生時的privilege level存到mstatus.MPP
- ⊕ 在Interrupt結束後能使用MPP的值恢復之前的工作模式
- ⊕ **mstatus(machine status register)**
 - 負責記錄機器模式(machine mode)下狀態的暫存器



mstatus register格式

Step 4. disable further interrupts

- 將mstatus.mie設為0，暫停處理這個中斷期間來的其他中斷，也就是說在這期間其他的中斷都不會被回應

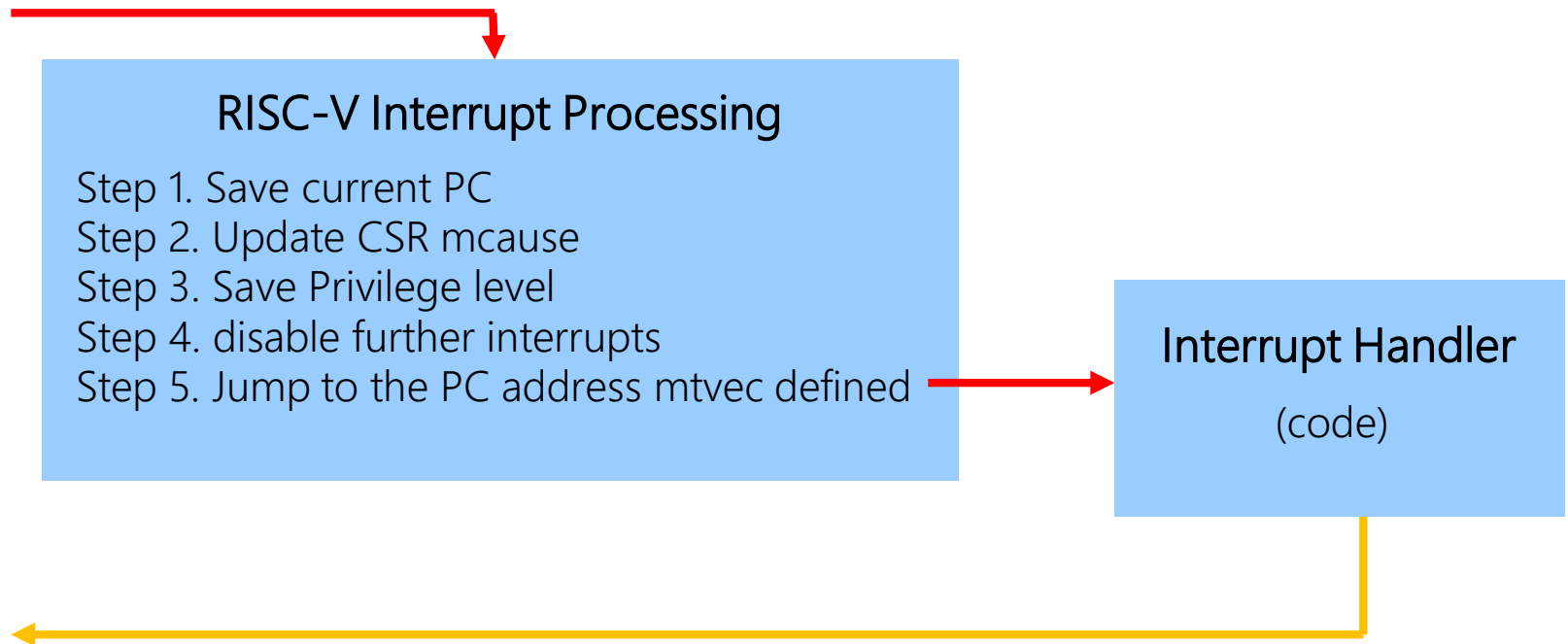


mstatus register格式

Step 5. Jump to the PC address mtvec defined

- ⊕ Processor暫停當前的指令，跳入mtvec暫存器中定義的地址
- ⊕ mtvec(Machine Trap-Vector Base-Address Register)
 - 是一個可讀可寫的暫存器，可透過軟體更改其中的值
 - 儲存地址，遇到interrupt時程式會跳進mtvec中所儲存的地址

RISC-V Interrupt Processing



LAB 4: RISC-V CPU Interrupts handler and ISR

Tool used

➤ 實驗環境：

1. Linux

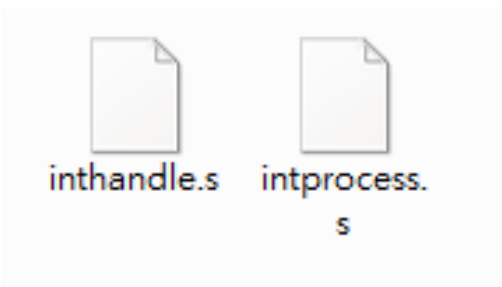
- 因為我們使用的 RISC-V Cross compiler 需要在 Linux 下才能執行

2. Modelsim (Run CPU simulator)


- 看 wave 來驗證執行結果是否正確

實作Interrupt handler 與 ISR

- 請以assembly code完成RISC-V CPU的 interrupt handler 與 ISR 的空白部分，並且使用 RISC-V Cross compiler編譯完成後產生 memory initialization 檔案，將內容放到 main.mem 裡並以 ModelSim 驗證。請同學依照流程以及規定撰寫，否則不予計分

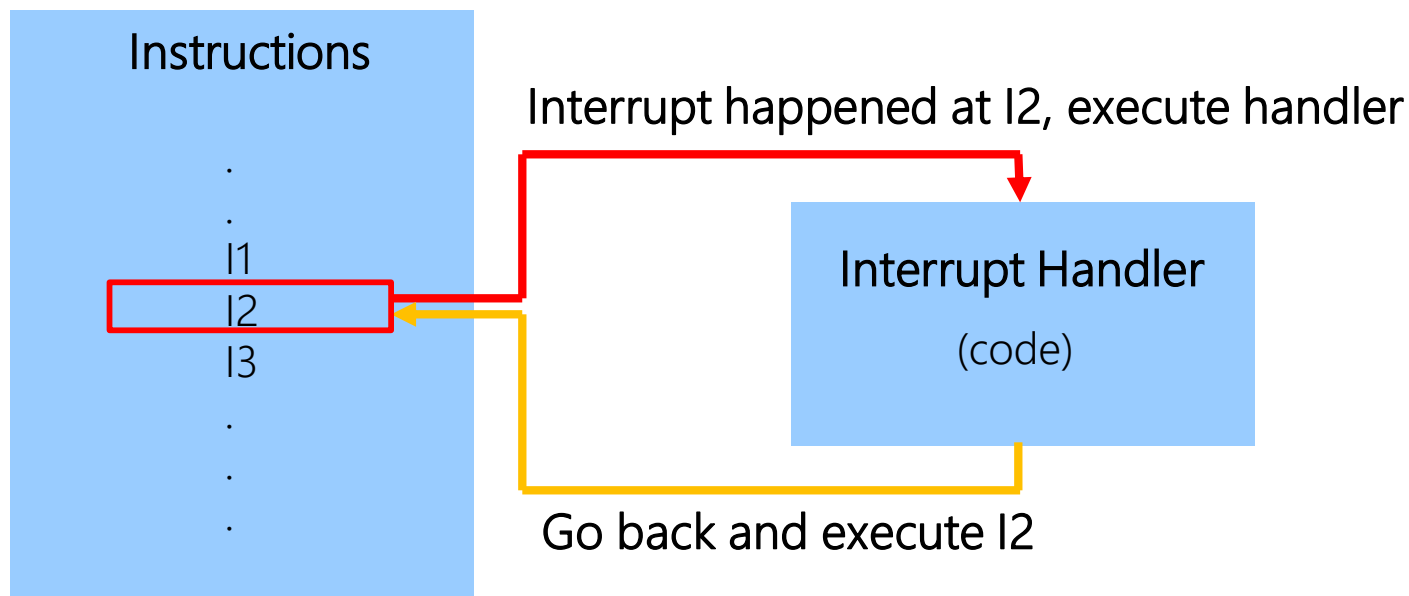


Interrupt handler 與 ISR 流程



Software 流程	
Handler	儲存 ra
	儲存 mepc, mstatus, mcause
	依據 mcause 算出要跳入的 ISR 地址
ISR	複製所有 register
	處理中斷
	恢復所有 register
	跳回 Handler
Handler	恢復 mepc, mstatus, mcause (注意：mstatus改成enable interrupt)
	恢復 ra
	跳回 mepc存的中斷發生地址

Interrupt handler 與 ISR 流程



第一部分

Step1. ①

進入對應的服務位置之前:

必須先備份mepc(833)、mstatus(768)、mcause(834)到memory stack(sp)中,根據發生中斷的原因跳到對應的服務位置執行

Step2. ②

從對應的服務位置跳回來之後:

此時必須要將mepc、mstatus、mcause從memory還原到暫存器中,然後enable interrupt(修改mstatus)並且最後跳回mepc位置

第一部分

請同學參考前面說明，
並且完成程式碼
(inthandle.s)

```
addi sp, sp, -4  
sw    ra, 0(sp)  
addi sp, sp, -12  
csrr  x31, 833      # mepc=833(0x341)  
sw    x31, 0(sp)
```

1

```
andi  x31, x31, 3  
addi  x31, x31, 1197  
jalr  x31
```

2

```
addi sp, sp, 12  
lw    ra, 0(sp)  
addi sp, sp, 4  
jr     x31
```


第二部分

Step1. 1

要進入對應的服務位置前,備份register x1~ x31的內容到memory stack裡,備份完後interrupt服務方可使用register

Step2. 2

返回執行中斷點後,還原register內的值,這樣程式返回執行才不會出錯

第二部分

請同學參考前面說明，
並且完成程式碼
(intprocess.s)

```
addi sp, sp, -128  
sw    x1, 4(sp)
```

1

```
li    x14, 0  
li    x15, 960  
sw    x14, 0(x15)
```

2


```
addi sp, sp, 128  
jr    ra
```

指令介紹

⊕ csrr

- 讀取CSR(control and status register)
- e.x. 833為CSR mepc的地址, 下圖範例為將mepc的值讀入x31 register中


```
csrr x31, 833      #mepc=833(0x341)
```



⊕ csrw

- 寫CSR
- 下圖範例則是將mepc的值寫入x31 register

```
csrw 833, x31
```



編譯程式

- 在linux下打開Terminal接著輸入以下指令
- ⊕ `riscv32-unknown-elf-as -mabi=ilp32 inthandle.s -o inthandle.o`
- ⊕ `riscv32-unknown-elf-ld -b elf32-littleriscv -T link.ld inthandle.o -o inthandle`
- ⊕ `riscv32-unknown-elf-objdump -dC inthandle > inthandle.dump`
- ⊕ `riscv32-unknown-elf-objcopy -O binary inthandle inthandle.bin`
- ⊕ `python3 bin2mem.py --bin inthandle.bin`
- ⊕ 紅字部分請自行替換檔案名稱
- ⊕ 指令詳細介紹請參考LAB1、LAB2敘述

放入編譯好的程式

```
00000013|
1F400793
3C000693
00100613
30579073
00100713
00200713
00300713
00400713
00C6A023
00500713
00600713
00700713
00800713
00900713
00A00713
00B00713
00008067
```

@7D

inhandle.mem

1

@12C

intprocess.mem

2

➤ 將編譯好的.mem檔案內容
複製到main.mem, 並在對
應處貼上

⊕ (1) inhandle.mem

⊕ (2) intprocess.mem

驗證結果

檢查系統是否成功備份資料,並且處理Interrupt handler & ISR
後,返回Program,是否會影響原本的結果

➤ 驗證項目:

- 1) 中斷結束後，原本的程式是否維持原狀，繼續執行
- 2) register x0~x31是否成功備份、還原
- 3) mepc, mstatus, mcause是否成功備份、還原
- 4) 離開中斷程式之前,是否 enable interrupt

modelsim transcript 成功訊息

- ⊕ Modelsim terminal處會顯示成功訊息

```
#  
# =====  
# Interrupt Handler success : )  
# =====  
# Break in Module tb_core_ut at C:/Users/Ahua/Desktop/co2020/2020_lab4/testbench.sv line 180  
VSIM 12>
```


實驗結報

⊕ 結報格式(每組一份)

- 封面 (第幾組+組員)
- 實驗內容(程式碼註解、結果截圖)
- 實驗心得

⊕ 繳交位置

- <ftp://140.116.164.225/> port: 21
- 帳號/密碼 : ca_lab / Carch2020

⊕ DeadLine: 11/2 18:00前

⊕ TA Contact Information:

- 助教信箱 : anita19961013@gmail.com
- Lab: 92617
- Office hour : (Tue)8:00pm~10:00pm