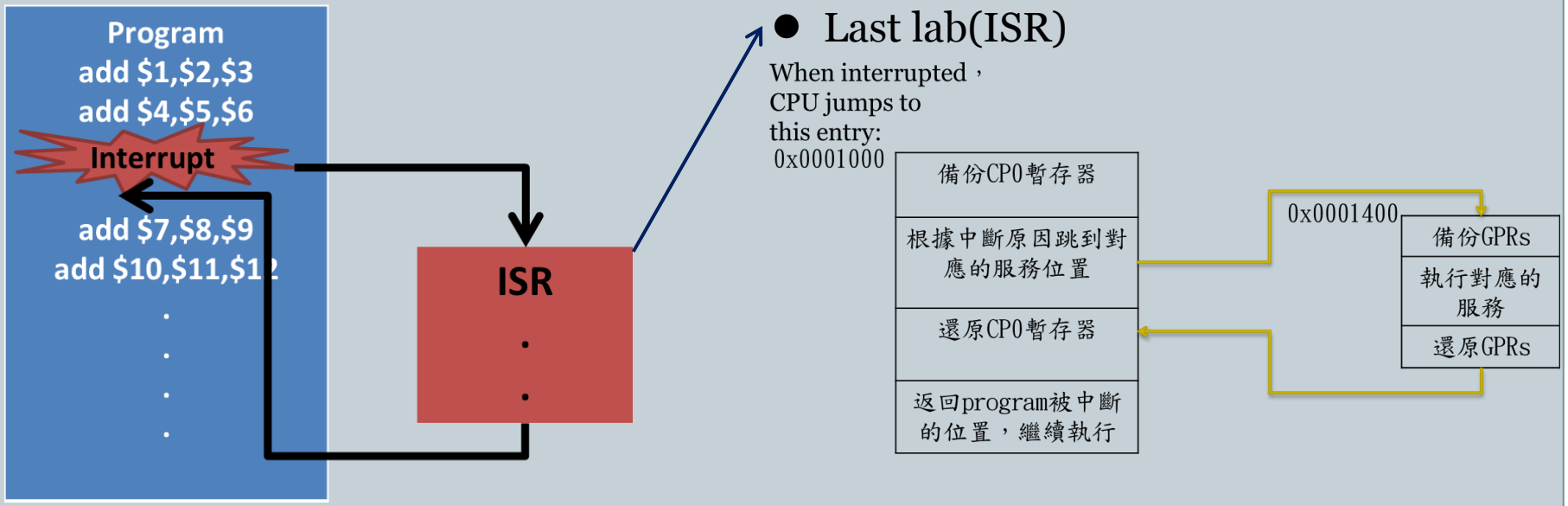
MIPS CPU Interrupt Unit

實驗目的

1. 學習Precise interrupt in MIPS pipeline
2. Review Coprocessors 0 registers
3. 了解MIPS CPU interrupt硬體處理流程
4. Implementation of processor interrupt module and pipeline flushing (IF stage)

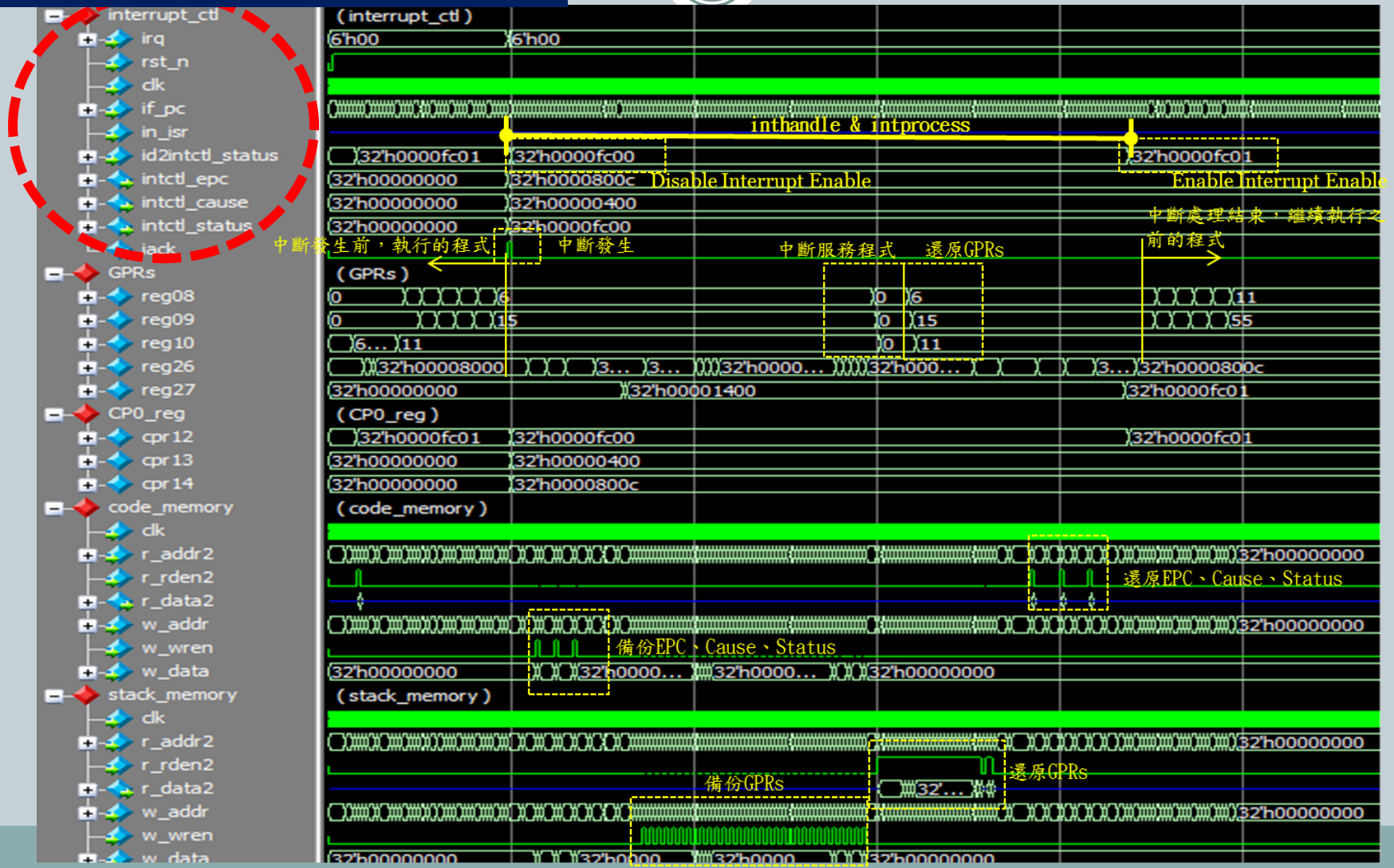
Background

Last lab we did ISR, this lab we focus on the functionality of processor interrupt module.



Background

Interrupt_module(this lab)

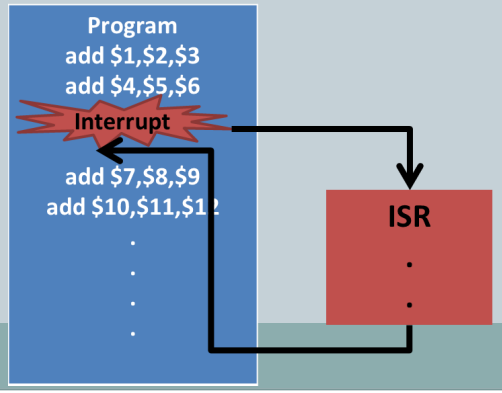


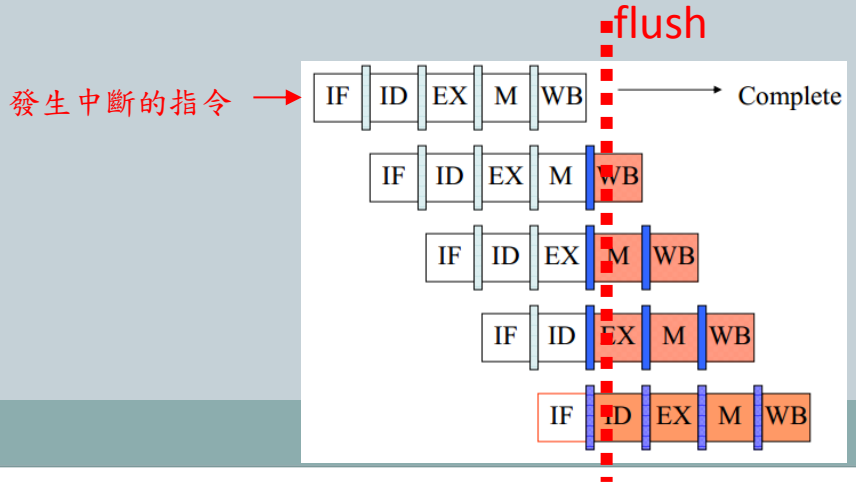
To implement a precise interrupt

1) An interrupt or exception is called precise if the saved processor state corresponds with the sequential model of program execution.

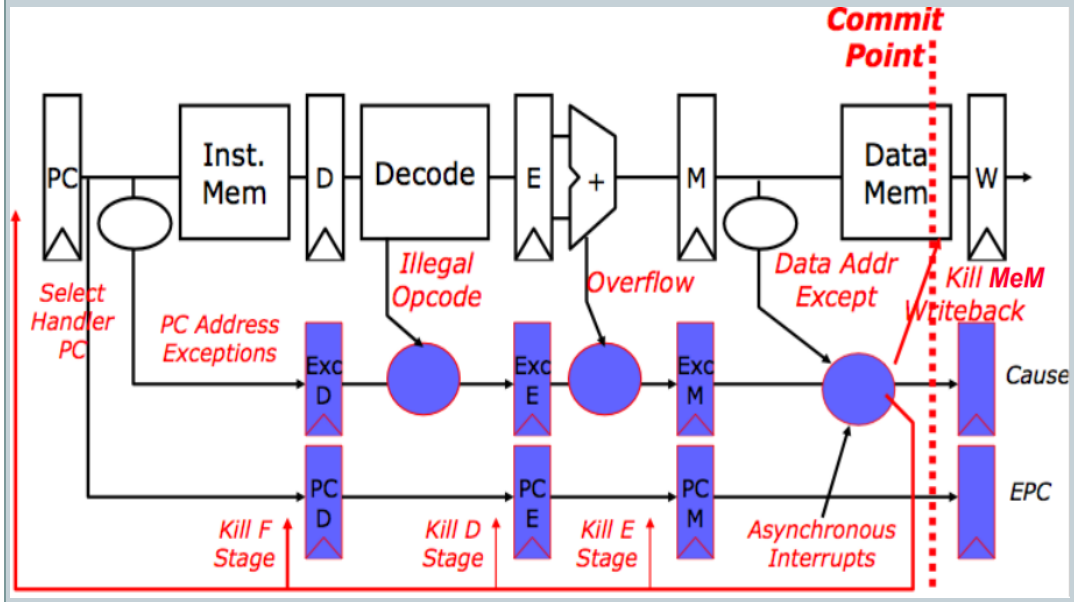
2) Precise interrupt means that all instructions before the interrupt or faulting instruction are committed and those after it can be restarted from scratch.

3) If an interrupt occurred, all instructions that are in program order before the interrupt signaling instruction are committed, and all later instructions are removed (flush).





Processor Pipeline Model and Exception

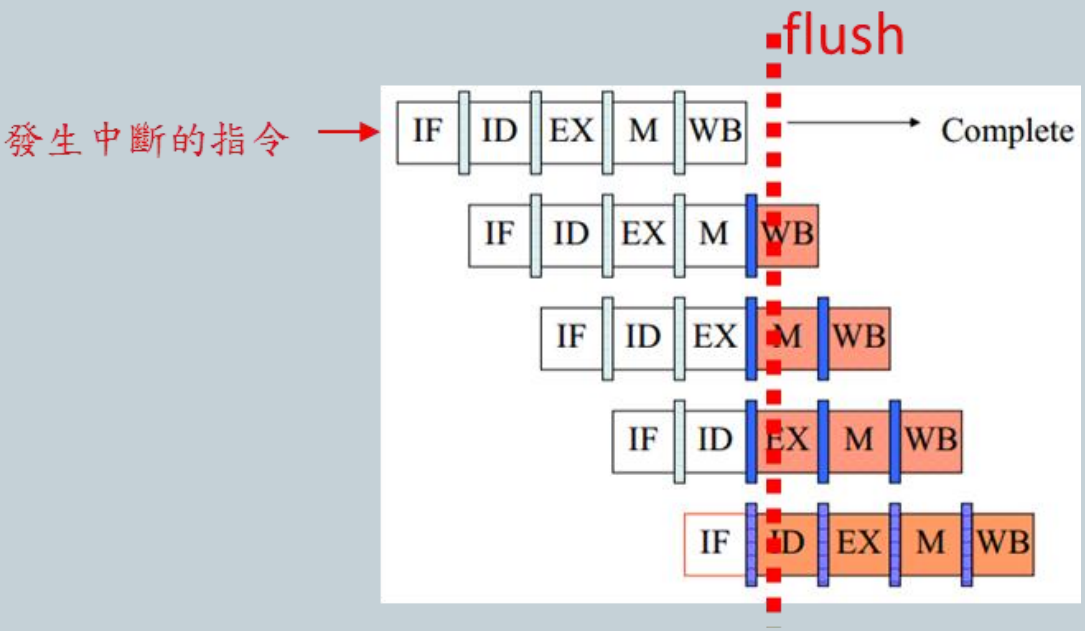


IF	Data abort, misaligned memory access , memory-protection violation
ID	Undefined instructions
EX	Arithmetic interrupt (overflow)
MEM	Data abort , misaligned memory Access ,memory-protection violation
WB	None

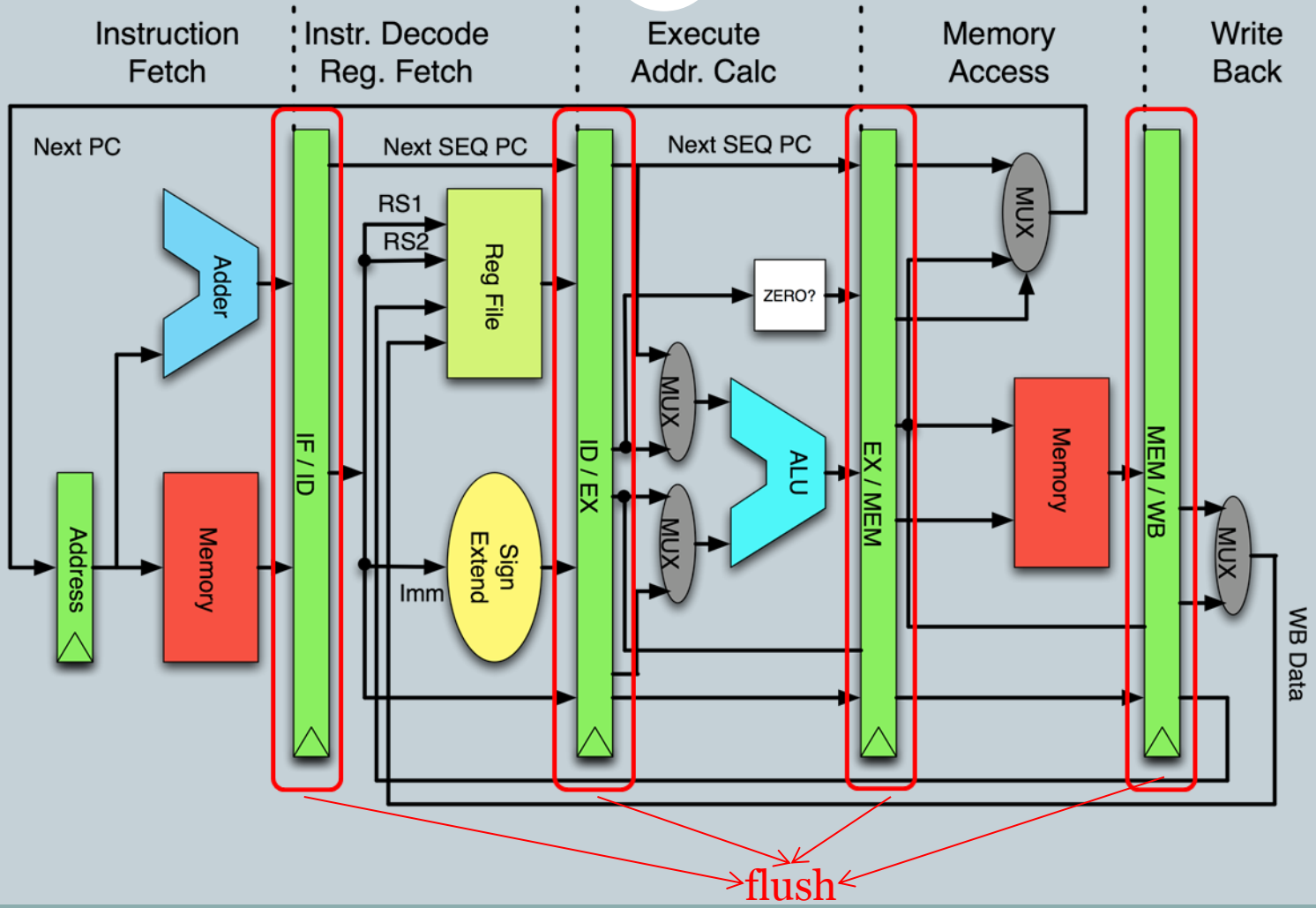
Precise Exceptions in MIPS Pipeline

To implement precise interrupt, the processor needs to flush IF、ID 、EXE 、MEM stages.

0x00	add $1,$2,$3
0x04	add $4,$5,$6
0x08	add $7,$8,$9
0x0c	add $10,$11,$12
0x10	add $13,$14,$15



Flush register between stage and stage



Review Coprocessors 0 registers

➢ CP0 Registers

- The processor is running in Kernel Mode or Debug Mode.

Register Number	Register Name	Function
0	..	..
..	..	..
12	Status	Processor status and control
13	Cause	Cause of last general exception
14	EPC	Program counter at last exception
..	..	..
31	..	..

CP0 Registers- Status

➢ Status:

31–28	27	26	25	24	23	22	21	20	19	18	17	16–15	10–9	8–7	6–5	4	3	2	1	0
CU3..CU0	RP	FR	RE	MX	0	BEV	TS	SR	NMI	ASE	Impl	IM7..IM2	IM1..IM0	0		UM	R0	ERL	EXL	IE
												IPL				KSU				

- IM7…IM2 [15:10] : (控制哪一個硬體中斷被遮蓋)

  ✓ Interrupt Mask:

  Control the enabling of each of the hardware interrupts.

代表被遮蓋

IM7	IM6	IM5	IM4	IM3	IM2
1	1	1	0	1	1

- IE [0] : (是否接受中斷要求)

  ✓ Interrupt Enable:

  Act as the master enable for software and hardware interrupts.

Encoding	Meaning
0	Interrupt request disabled
1	Interrupt request enabled

☐ EX.

CP0_status[15:10]	111110
CP0_status[0]	1
Irq[5:0]	000001

雖然IE=1，但由於IP2被MASK住，所以中斷要求被拒。

☐ EX.

CP0_status[15:10]	111111
CP0_status[0]	0
Irq[5:0]	000001

雖然IP2沒被MASK住，但IE=0，所以中斷要求被拒。

CP0 Registers- Cause

➢ Cause :

31	30	29 28	27	26	25 24	23	22	21	20 17	15 10	9 8	7	6 2	1 0
BD	TI	CE	DC	PCI	ASE	IV	WP	FDCI	000	ASE	IP7…IP2	IP1..IP0	0	Exc Code	0
										ASE	RIPL				

- IP7…IM2 [15:10]:

Record the reason for the exception in the Cause register.

Bit	Name	Meaning
15	IP7	Hardware interrupt 5
14	IP6	Hardware interrupt 4
13	IP5	Hardware interrupt 3
12	IP4	Hardware interrupt 2
11	IP3	Hardware interrupt 1
10	IP2	Hardware interrupt 0

IP7	IP6	IP5	IP4	IP3	IP2
0	0	0	0	0	1

代表IP2硬體發出中斷要求，且被CPU接受

MIPS Interrupt Processing

- If interrupt is enabled and is not masked , then MIPS does interrupt processing in 5 steps.

Step 1. save current PC

Step 2. set state giving the cause of exception

Step 3. change to kernel mode

Step 4. disable further interrupts

Step 5. jump to exception handler address

(The red part will be the work to be implemented by this LAB)

MIPS Interrupt Processing

➢ 判斷是否可接受中斷要求

- First, check interrupt mask field (CP0_status[15:10]) and interrupt enable field (CP0_status[0])

◆ CP0_status

31–28	27	26	25	24	23	22	21	20	19	18	17	16	15–10	9–8	7	6	5	4	3	2	1	0
CU3..CU0	RP	FR	RE	MX	0	BEV	TS	SR	NMI	ASE	Impl		IM7..IM2	IM1..IM0		0		UM	R0	ERL	EXL	IE
													IPL					KSU				

✓ EX.

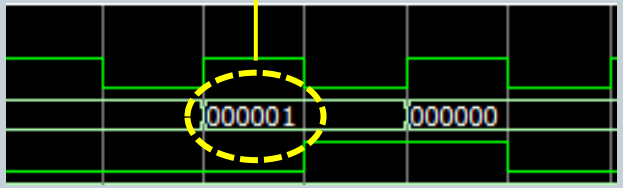
CP0_status[15:10]	111111
CP0_status[0]	1
irq[5:0]	000001

代表可接受IP2的中斷要求

- We need a way to tell the external device that the processor will serve its interrupt.

  We'll solve this problem by adding the pin, called IACK (interrupt acknowledge), that will be an output.

代表IP2硬體發出中斷服務要求

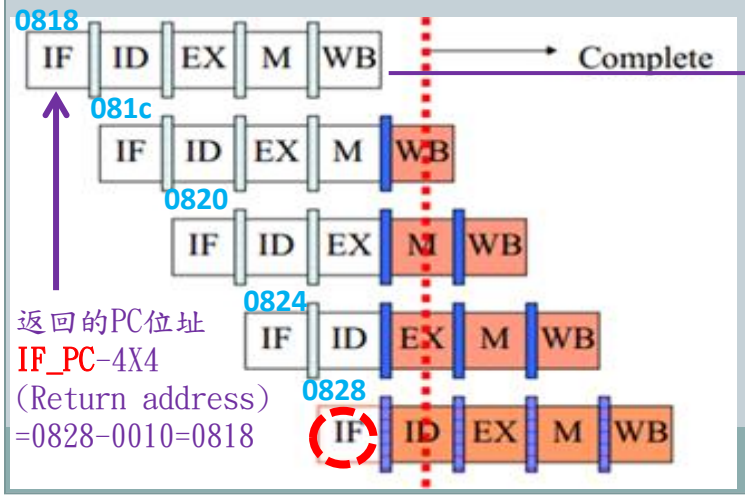


由中斷的硬體所發出的中斷服務要求訊號 ← Irq

由CPU所發出給認可訊號，給發出中斷要求的硬體 ← Iack

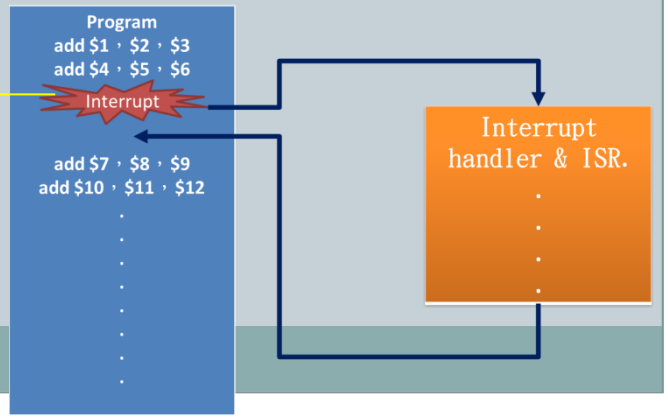
Step 1. save current PC

- CP0_epc <= ? (which PC to save for pipeline CPU?)
- When an interrupt is caused by external devices or by those interrupt causing instructions, we need to give a return address to the user program.
- We can't just save CP0_epc like this CP0_epc = current PC which is being used for fetching instruction.
- Since the interrupt module is located at the WB stage, so the PC to be saved is (current PC) – 4 x 4.



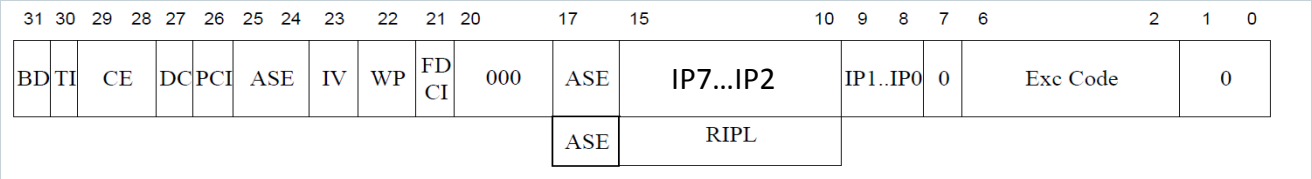
這個指令動作完成後發生中斷

Return address



Step 2. set state giving cause of exception

- CP0_cause[15:0] <= (cause code for event)
- We need to record the reason for the exception in the Cause register.



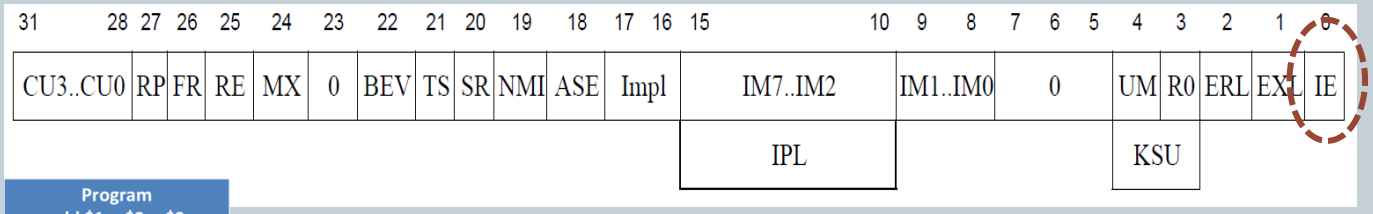
IP7	IP6	IP5	IP4	IP3	IP2
0	0	0	0	0	1

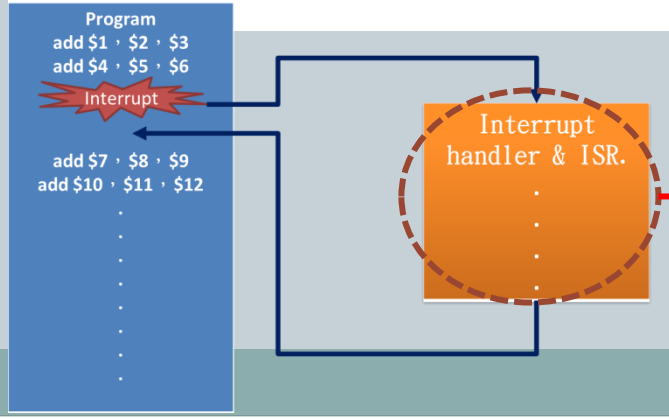
代表IP2硬體發出中斷要求，
且被CPU接受

Step 4. Disable further interrupts

- We require a way to disable interrupts and exceptions. This is necessary to prevent further exceptions and interrupts during this phase.
- Bit 0 of the Status register， the field is called IE (Interrupt Enable)
- Enabled = 1 ，不接受中斷需求。

  Disabled = 0，接受中斷要求。





在執行此中斷服務程式時，拒絕其他的硬體發出的中斷要求。

MIPS CPU Interrupt Unit

Tool used

實驗環境:

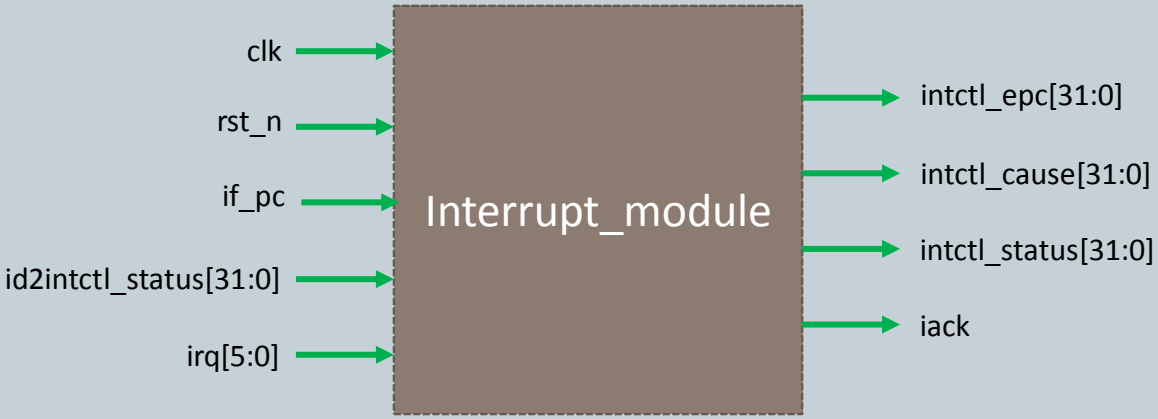
1. Modelsim (Run CPU simulator)

   - 看wave來驗證我們的實作是否正確

實作題

- 請同學以RTL code完成這顆MIPS CPU中interrupt_module.v 空白部分(v code檔在MIPS_cpu_test_interrupt_module),並且使用ModelSim編譯完成,在使用以提供的組合語言程式(已放置在mips_sorftware底下),來驗證硬體行為是否正確。

請同學依照流程以及規定撰寫,請先複習前面投影片



Finish the following code

```
module interrupt_module(irq,rst_n,clk,if_pc,intctl_epc,intctl_status,intctl_
        input [5:0] irq;
        input   rst_n;
        input   clk;
        input   if_pc;
        input   [31:0] id2intctl_status;

        output  [31:0] intctl_epc;
        output  [31:0] intctl_cause;
        output  [31:0] intctl_status;
        output  iack;

        reg     [31:0]  intctl_epc;
        reg     [31:0]  intctl_cause;
        reg     [31:0]  intctl_status;

        wire    [31:0]  if_pc;
        reg             iack;
```

請同學先了解會用到哪些input/output, 再開始撰寫程式碼

- 因為必須修改interrupt_module.v，所以我們必須要Add Existing File 把interrupt_module.v添加到Project中

Project - D:/Wang/Sean/CPU_SEAN_V2/MIPS_cpu_test_interrupt_module/MIPS_cpu_test_interrupt_module

Name	Status	Type	Order	Modified
interrupt_module.v	✗	Verilog	1	10/23/2013 11:12:16 ...
timescale.v	✓	Verilog	0	12/28/2010 09:33:18 ...

Finish the following code

請參考下一頁說明,並依項目完成相對程式碼

```
always@(negedge clk or negedge rst_n)
begin
    if(~rst_n)        //暫存器的初始值設定                    1
            begin
                iack<=
                intctl_epc <=
                intctl_status<=
                intctl_cause <=
            end
    else if( )        //判斷硬體中斷是否可被接受              2
            begin
                intctl_epc <=                          //計算中斷結束後的返回的PC
                intctl_status<=                        //disable interrupt，避免其他中斷要求的干擾
                intctl_cause <=                        //紀錄中斷原因(=irq[5:0])
                iack <=                                //發出中斷認可
            end
    else                                               3
            begin
                if(if_pc == 32'h00001000)              //判斷是否已進入中斷服務程式(ISR)的指令位址
                    begin
                        iack <=                        //中斷認可訊號拉回0
                    end
            end
end
```

Interrupt module實作補充說明

① rst_n表示為reset訊號且為負觸緣,此處必須完成當reset成立時候對interrupt control硬體初始化

② 先檢查IRQ是否被mask住,同時檢查是否enable interrupt ,之後流程如下:

- ✓ save current PC(return address)
- ✓ set state giving the cause of exception(原因為irq[5:0])
- ✓ disable further interrupts

(此三個動作投影片前面已詳細說明,step1、step2、step4)

③ 此部分是當pc值表示0x00001000表示已進入ISR,將iack訊號拉回0

Wave

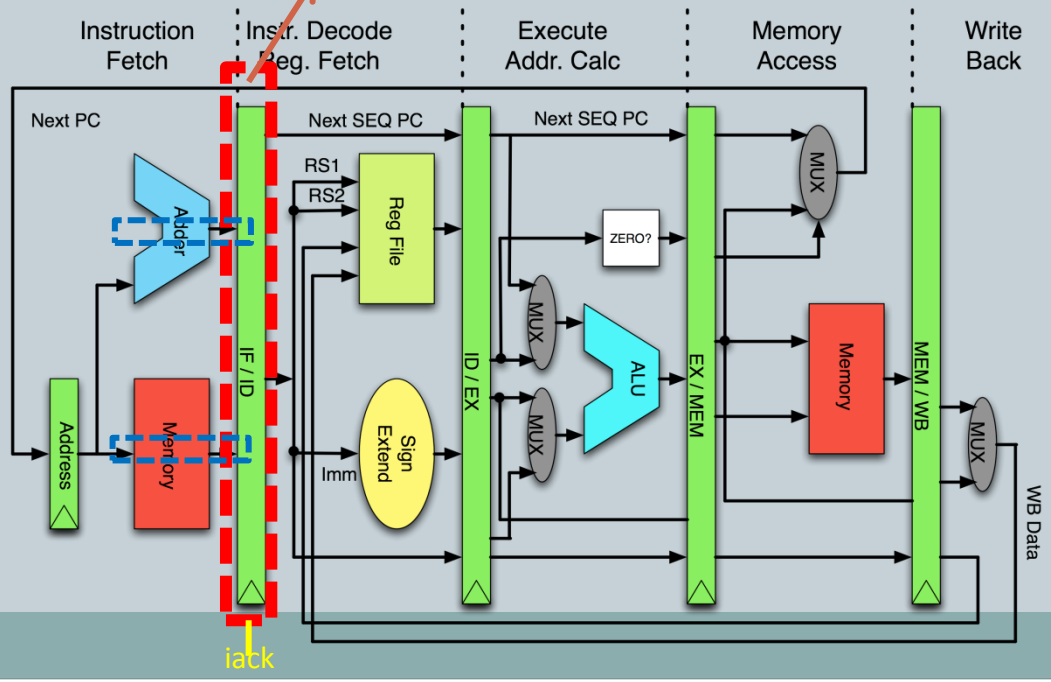
➢ 欲觀察的驗證的波型圖檔在MIPS_cpu_test_interrupt_module/interrupt_Unit.do

進階題

- 本進階題練習實作IF級的flush功能
- 請仔細閱讀cpu_if.v中的IF/ID Register部分
- 請完成flush IF/ID register

Tips:
cpu_if.v已將iack訊號接線,僅需修改cpu_reg32_ne_ex.v即可完成IF flush 功能

Flush the register (IF/ID)




```
//=========================================
//      IF/ID Register
//=========================================

cpu_reg32_ne_ex    u_if2id_pca4(
                    .clk          (clk),
                    .rst_n        (rst_n),
                    .din          (if_pca4),
                    .iack         (iack),
                    .wren         (if2id_r_en),
                    .dout         (id_pca4)
                   );

cpu_reg32_ne_ex    u_if2id_ir(
                    .clk          (clk),
                    .rst_n        (rst_n),
                    .din          (if_ins),
                    .iack         (iack),
                    .wren         (if2id_r_en),
                    .dout         (id_ins)
                   );
```

進階題說明

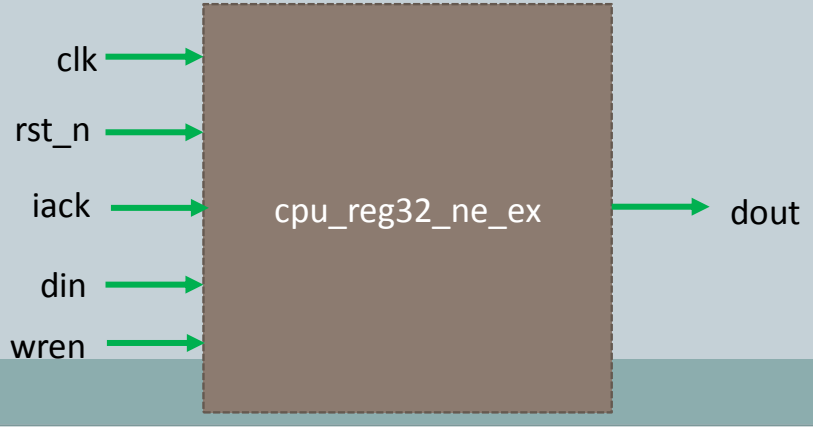
- 操作與資料夾路徑如下:

  MIPS_cpu_test_IF_ID_flush

- 因為必須修改cpu_reg32_ne_ex.v，所以我們必須要Add Existing File 把cpu_reg32_ne_ex.v添加到Project中

Project - D:/Wang/Sean/CPU_Sean/LAB7/CHALLENGE/ADV.MIPS_cpu_test/MIPS_cpu_test/MIPS_cpu_test

Name	Status	Type	Order	Modified
cpu_reg32_ne_ex.v	✓	Verilog	0	10/23/2013 12:12:30 ...
timescale.v	✓	Verilog	1	12/28/2010 09:33:18 ...



進階題說明

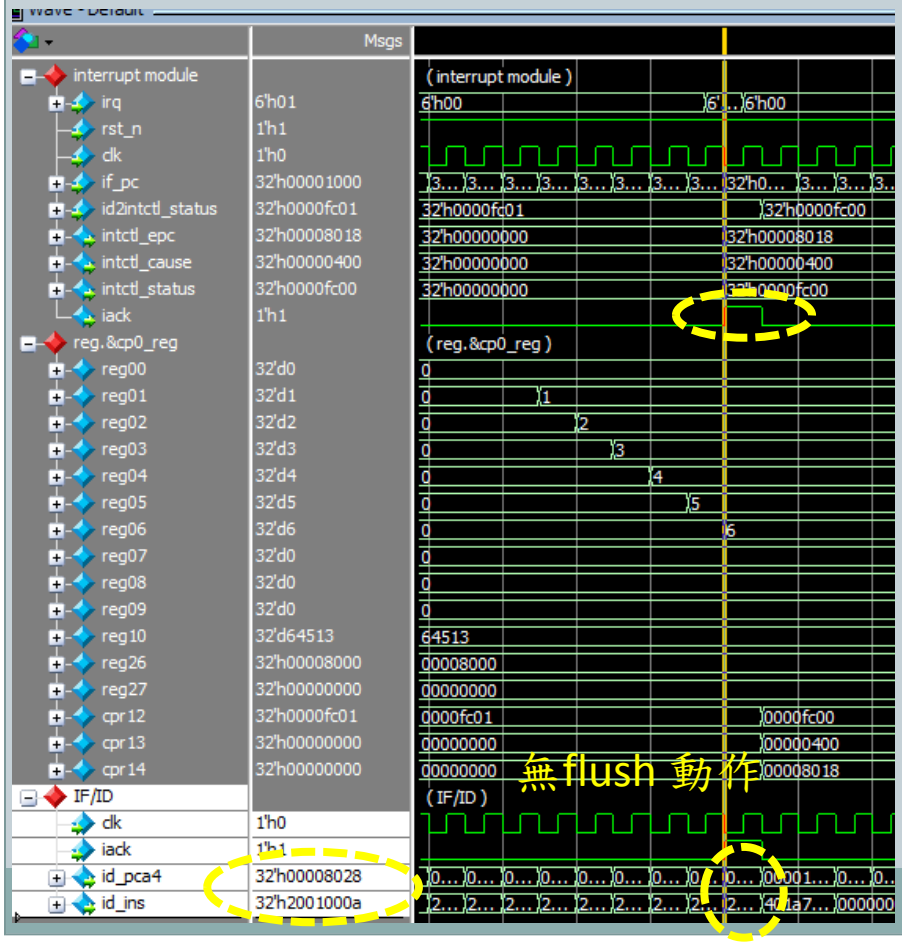
- 請修改cpu_reg32_ne_ex.v，使其有flush IF/ID register的功能
- 下圖已有註解當作提示:

```
module cpu_reg32_ne_ex(clk, iack, rst_n, din, wren, dout);

        input           clk;    // System clk
        input           rst_n;  // System Reset
        input   [31:0]  din;    // Data input
        input           wren; // Enable Register

        input           iack;
        output  [31:0]  dout;   // Data Output


        reg     [31:0]  dout;

        always@(negedge clk or negedge rst_n or posedge ??)     //當正緣觸發?? or 負緣觸發rst_n時 or 正緣觸發clk時
        begin
                //若rst_n ==0 或 ?? == 1 時，設dout = 0(flush動作)
                if(~rst_n)
                    begin
                    dout<=32'b0;
                    end
                else
                    begin
                        if(wren) begin dout<=din;  $monitor("%0dns :\$monitor: wren=%b ",$stime,wren);   end
                        else    dout<=dout;
                    end

        end


endmodule
```

Wave

➢ 欲觀察的驗證的波型圖檔在MIPS_cpu_test_interrupt_module/IF_ID_flush.do

◆ error



◆ correct