

# Lab04-4

## CPU设计—中断

Ma De (马德)

[made@zju.edu.cn](mailto:made@zju.edu.cn)

2022

College of Computer Science, Zhejiang University

# Course Outline

---

- 一、实验目的
- 二、实验环境
- 三、实验目标及任务

# 实验目的

---

1. 深入理解CPU结构
3. 学习如何提高CPU使用效率
3. 学习CPU中断工作原理
4. 设计中断测试程序

# 实验环境

---

## □ 实验设备

1. 计算机（Intel Core i5以上，4GB内存以上）系统
2. Sword2.0/Sword4.0开发板
3. Xilinx Vivado14.7及以上开发工具

## □ 材料

无

# 实验目标及任务

---

- **目标：** 熟悉RISC-V 中断的原理，了解引起CPU中断产生的原因及其处理方法，扩展包含中断的CPU
- **任务一：** 扩展实验CPU中断功能
  - 修改设计数据通路和控制器
    - 修改或替换Exp04-3的数据通路及控制器
    - 兼容Exp04-3数据通路增加中断通路
    - 增加中断控制
  - 扩展CPU中断功能
    - 非法指令中断；
    - 外部中断；
    - ecall
- **任务二：** 设计CPU中断测试方案并完成测试

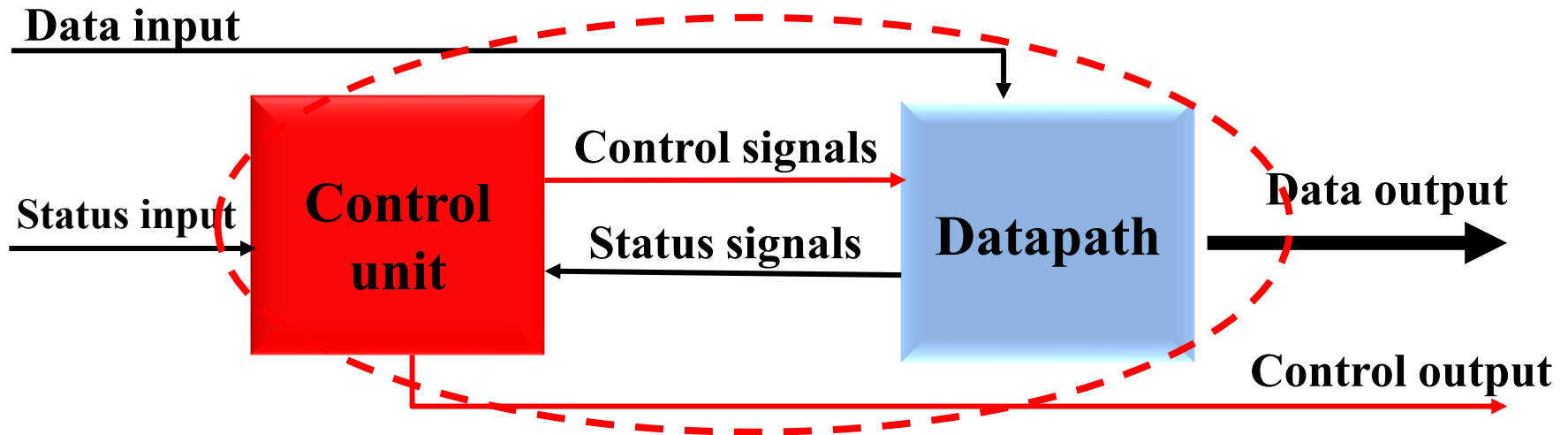
---

# CPU中断的原理介绍

# CPU organization

## □ Digital circuit

- General circuits that controls logical event with logical gates -  
**-Hardware**



## □ Computer organization

- Special circuits that processes logical action with instructions  
**-Software**

# 中断概念

---

- **中断**是指程序执行过程中，当发生某个事件时，中止CPU上现行程序的运行，引出处理该事件的程序执行的过程，此过程都需要打断处理器正常的工作，为此，才提出了“中断”的概念。

中断  
原因

- 请求系统服务
- 实现并行工作
- 处理突发事件



# 中断概念

---

- **中断源**:引起中断的事件称为中断源;
- **中断请求**:中断源向CPU提出处理的请求;
- **断点**:发生中断时被打断程序的暂停点;
- **中断响应**:CPU暂停现行程序而转为响应中断请求的过程;
- **中断处理程序**:处理中断源的程序;
- **中断处理**:CPU执行有关的中断处理程序;
- **中断返回**:返回断点的过程;

# 中断概念

□ 按照中断信号的来源，可把中断分为外中断和内中断两类：

- 外中断(又称中断)：指来自处理器和主存之外的中断；
- 内中断(又称异常)：指来自处理器和主存内部的中断；

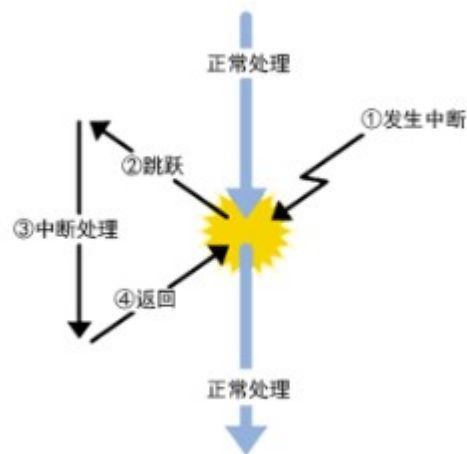
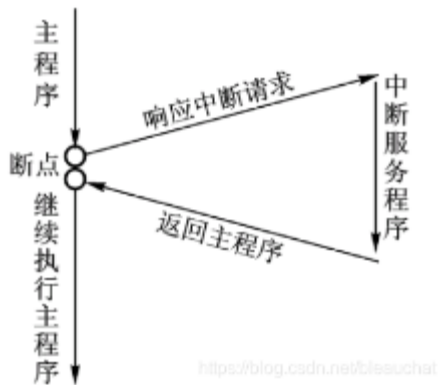
□ 中断处理程序主要工作：

- 保护CPU现场
- 处理发生的中断事件
- 恢复正常操作

狭义的中断  
和异常均可  
归于广义的  
异常范畴

# 中断（异常）处理过程

- 当CPU收到中断或者异常的信号时，它会暂停执行当前的程序或任务，通过一定的机制跳转到负责处理这个信号的相关处理程序中，在完成对这个信号的处理后再跳回到刚才被打断的程序或任务中。



# RISC-V中断结构

---

## □ RISC-V架构工作模式

- 机器模式(Machine Mode)
- 用户模式(User Mode)
- 监督模式(Supervisor Mode)

## □ 不同的模式下均可产生异常以及中断

## □ RISC-V架构中机器模式是必须具备的模式，因此必须具备机器模式的异常处理机制（本实验只针对机器模式下的异常（中断））

# RISC-V中断处理---进入异常

## □ RISC-V处理器检测到异常，开始进行异常处理：

- 停止执行当前的程序流，转而从CSR寄存器mtvec定义的PC地址开始执行；
- 更新机器模式异常原因寄存器mcause
- 更新机器模式异常PC寄存器mepc
- 更新机器模式异常值寄存器mtval
- 更新机器模式状态寄存器mstatus

# 异常入口基址寄存器-mtvec

- ❑ RISC-V处理器进入异常后，跳入的PC地址由mtvec寄存器指定：



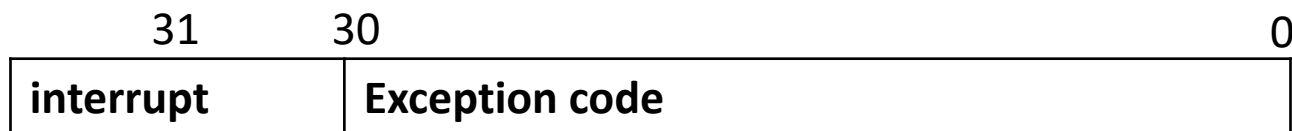
CAUSE表示  
中断对应的  
异常编号

- ❑ MODE = 0;异常响应时，处理器跳转到BASE值指示的PC地址
- ❑ MODE = 1;异常响应时，处理器跳转到BASE值指示的PC地址
- ❑ MODE = 1;中断响应时，处理器跳转到BASE+4\*CAUSE值指示的PC地址

本实验为简化设计，仿照ARM采用固定向量表，请见课件P26页

# 异常原因寄存器-mcause

- RISC-V处理器进入异常后，异常的引发原因由mcause寄存器指定：



INT	EC	Description
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned

INT	EC	Description
0	7	Store/AMO access fault
0	10..8	Not supported
0	11	Ecall from M-mode
0	>=12	Reserved
1	2..0	Reserved
1	3	Machine Software Interrupt
1	6..4	Reserved
1	7	Machine Timer Interrupt
1	10..8	Reserved
1	11	Machine External Interrupt
1	>=12	Reserved

# 异常PC寄存器-mepc

---

- ❑ RISC-V处理器进入异常后，异常的返回地址由mepc寄存器保存
- ❑ 在进入异常时，硬件将自动更新mepc寄存器的值为当前遇到异常的指令PC值(即当前程序的停止执行点)
- ❑ Mepc寄存器作为异常的返回地址，在异常结束后，能够使用它保存的PC值回到之前被停止执行的程序点



# 异常值寄存器-mtval

---

- ❑ RISC-V处理器进入异常后，异常的存储器访问地址或指令编码由mtval寄存器保存
- ❑ 如果是存储器访问造成的异常，如遭遇硬件断点、取指令、读写存储器造成异常，则将存储器访问的地址更新到mtval寄存器中
- ❑ 如果是非法指令造成的异常，则将改指令的指令编码更新到mtval寄存器中

# 异常状态寄存器-mstatus

- RISC-V处理器进入异常后，异常的各种状态由mstatus寄存器指示

Bits	Name	Attributes	Description
2..0	RSV	RZ	Reserved
3	MIE	RW	Global interrupt enable
6..4	RSV	RZ	Reserved
7	MPIE	RW	Previous global interrupt enable
10..8	RSV	RZ	Reserved
12..11	MPP	QRO	Previous privilege mode (hardwired to 11)
31..13	RSV	RZ	Reserved

- MIE = 1;表示机器模式下所有中断全局打开
- MIE = 0;表示机器模式下所有中断全局关闭

# 中断相关指令

## 异常返回

### MRET

PC  $\leq$  MEPC; (MStatus寄存器有变化)

31	27	26	25	24	20	19	15	14	12	11	7	6	0	
0001000					00010		00000	000		00000		1110011		R sret
0011000					00010		00000	000		00000		1110011		R mret
0001000					00101		00000	000		00000		1110011		R wfi

## 环境调用

### ecall

MEPC = ecall指令本身的PC值

## 断点

### ebreak

MEPC = ebreak指令本身的PC值

000000000000	00000	000	00000	1110011	I ecall
000000000001	00000	000	00000	1110011	I ebreak

# RISC-V中断结构---退出异常

- ❑ 当异常程序处理完成后，最终要从异常服务程序中退出，并返回主程序。RISCV中定义了一组退出指令MRET，SRET，和URET，对于机器模式，对应MRET。
- ❑ 在机器模式下退出异常时候，软件须使用MRET。RISCV架构规定，处理器执行完MRET指令后，硬件行为如下：
  - ❑ 停止执行当前程序流，转而从csr寄存器mepc定义的pc地址开始执行。
  - ❑ 执行mret指令不仅会让处理器跳转到上述的pc地址开始执行，还会让硬件同时更新csr寄存器机器模式状态寄存器mstatus。mstatus寄存器MIE域被更新为当前MPIE的值。MPIE 域的值则更新为1。

# RISC-V中断结构---异常服务程序

---

- ❑ 处理器进入异常后，即开始从`mtvec`寄存器定义的PC地址执行新的程序。
- ❑ 所执行的新的程序即为异常服务程序，并且程序还可以通过查询`mcause`中的异常编号决定跳转到更具体的异常服务程序。

# 典型处理器中断结构

## □ Intel x86中断结构

- 中断向量：000~3FF，占内存最底1KB空间
  - 每个向量由二个16位生成20位中断地址
  - 共256个中断向量，向量编号n=0~255
  - 分硬中断和软中断，响应过程类同，触发方式不同
  - 硬中断响应由控制芯片8259产生中断号n(接口原理课深入学习)

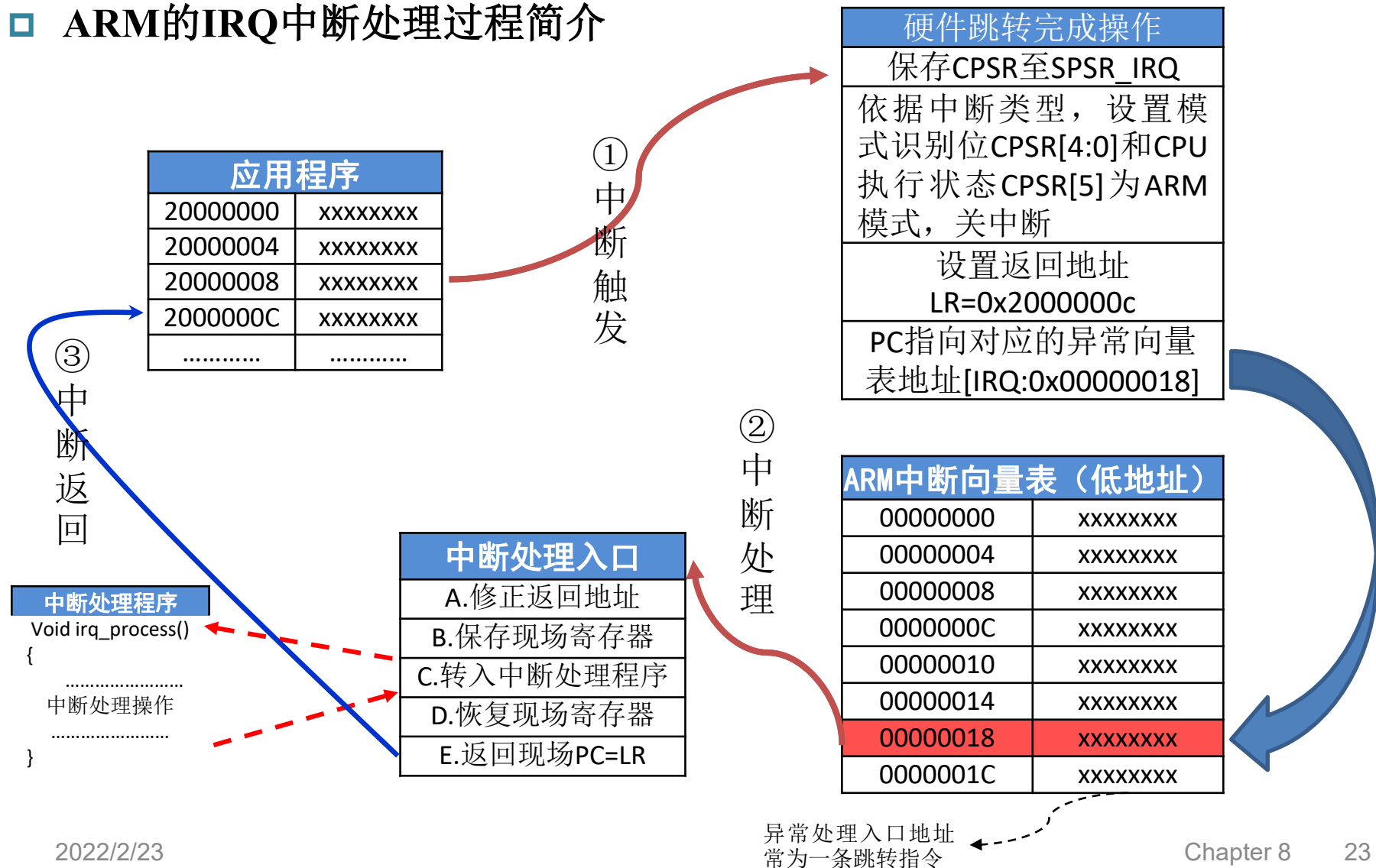
## □ ARM中断结构

- 固定向量方式(嵌入式课程深入学习)

异常类型	偏移地址(低)	偏移地址(高)	
复位	00000000	FFFF0000	
未定义指令	00000004	FFFF0004	
软中断	00000008	FFFF0008	
预取指令终止	0000000C	FFFF000C	
数据终止	00000010	FFFF0010	
保留	00000014	FFFF0014	
中断请求(IRQ)	00000018	FFFF0018	
快速中断请求(FIQ)	0000001C	FFFF001C	

# 典型处理器中断结构---ARM中断

## ARM的IRQ中断处理过程简介



---

## 任务一：扩展实验七CPU中断功能

- ▣ 修改设计数据通路和控制器
- ▣ 扩展CPU中断功能



# 简化中断设计：

## □ ARM中断向量表

向量地址	ARM异常名称	ARM系统工作模式	本实验定义
<b>0x0000000</b>	复位	超级用户Svc	复位
<b>0x0000004</b>	未定义指令终止	未定义指令终止Und	非法指令异常
<b>0x0000008</b>	软中断（SWI）	超级用户Svc	ECALL
<b>0x000000c</b>	Prefetch abort	指令预取终止Abt	Int外部中断（硬件）
<b>0x0000010</b>	Data abort	数据访问终止Abt	Reserved自定义
<b>0x0000014</b>	Reserved	Reserved	Reserved自定义
<b>0x0000018</b>	IRQ	外部中断模式IRQ	Reserved自定义
<b>0x000001C</b>	FIQ	快速中断模式FIQ	Reserved自定义

## □ 简化中断设计

### ■ 参考ARM中断向量

- 实现非法指令异常和外部中断以及ECALL
- 设计寄存器MEPC

# 简化中断设计：

- 1.外部中断（Int）触发中断或非法指令（illegal）触发异常或ecall系统调用
- 2.响应mtvec寄存器定义的PC值分别针对Int为0x0c；ecall为0x08；illegal为0x04
- 3.mepc寄存器值更新为下一条指令的PC值
- 4.执行异常服务程序
- 5.执行mret指令，返回mepc保存的PC处继续程序流

mtvec在此设计中仅为reg类型变量，功能是存储中断向量

mepc为受clk控制的寄存器，功能是暂存返回PC值

# 设计方案参考：DataPath

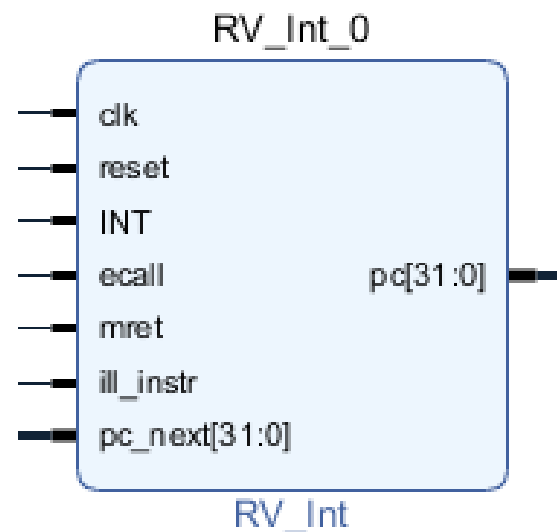
## ◎ DataPath修改

⌚ CPU复位时，MEPC=PC=0x00000000

⌚ 修改PC模块增加

- ⊙ mtvec寄存器型变量，中断和异常触发PC转向中断地址
  - ◆ 相当于硬件触发Jal，用mret返回
- ⊙ mepc寄存器，中断和异常返回PC的地址
- ⊙ 增加控制信号INT、mret、ecall、ill\_instr
  - ◆ INT宽度根据扩展的外中断数量设定

**注意：INT是电平信号，不要重复响应**



# 设计方案参考：控制器

---

## ◎ 控制器修改

### ㊦ 简洁模式

- ⊙ 增加mret、ecall指令以及非法指令的处理
- ⊙ 中断请求信号触发PC转向，在Datapath模块中修改

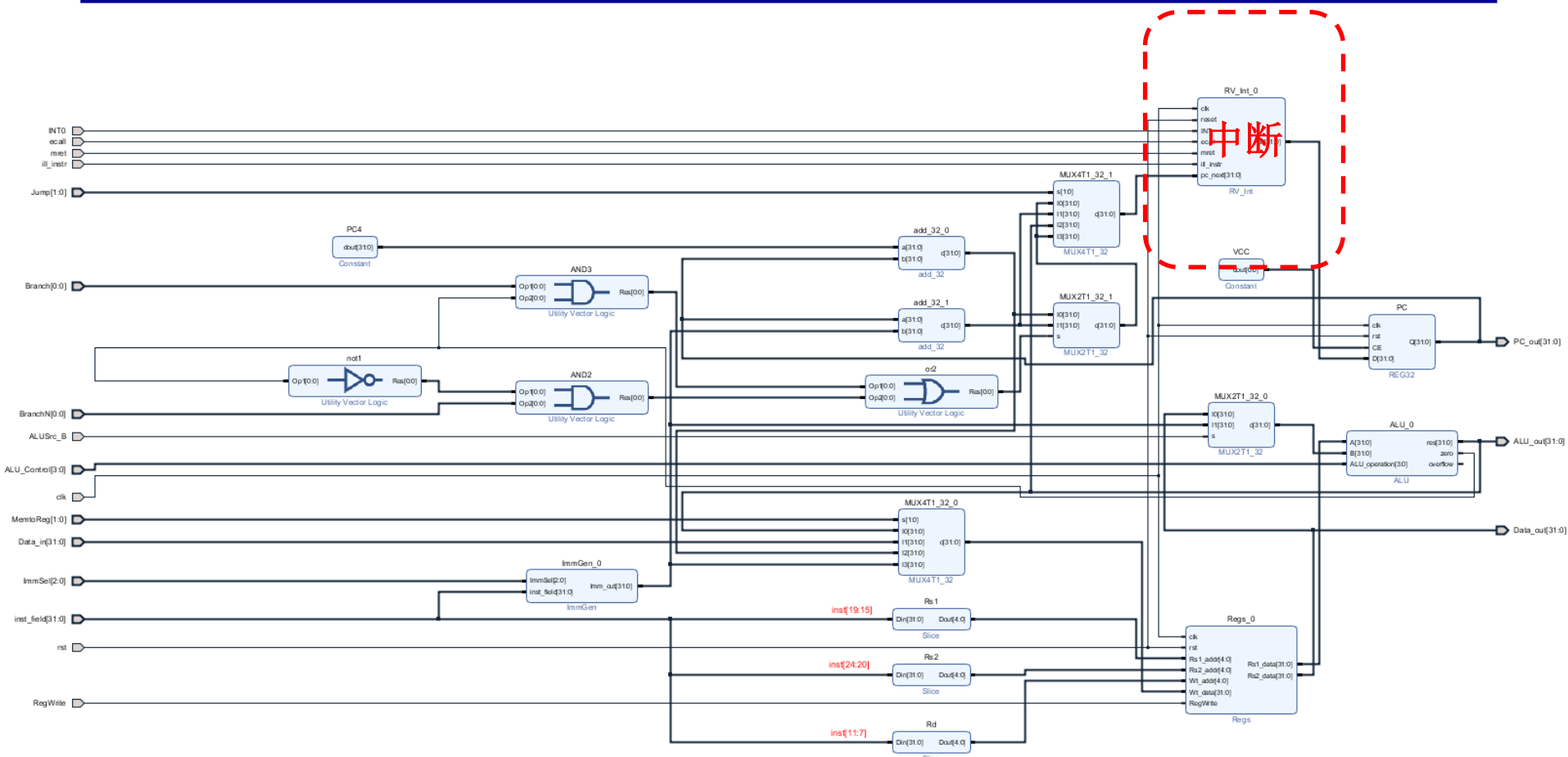
## ◎ 中断调试

### ㊦ 首先时序仿真（仿真平台参见lab04-2）

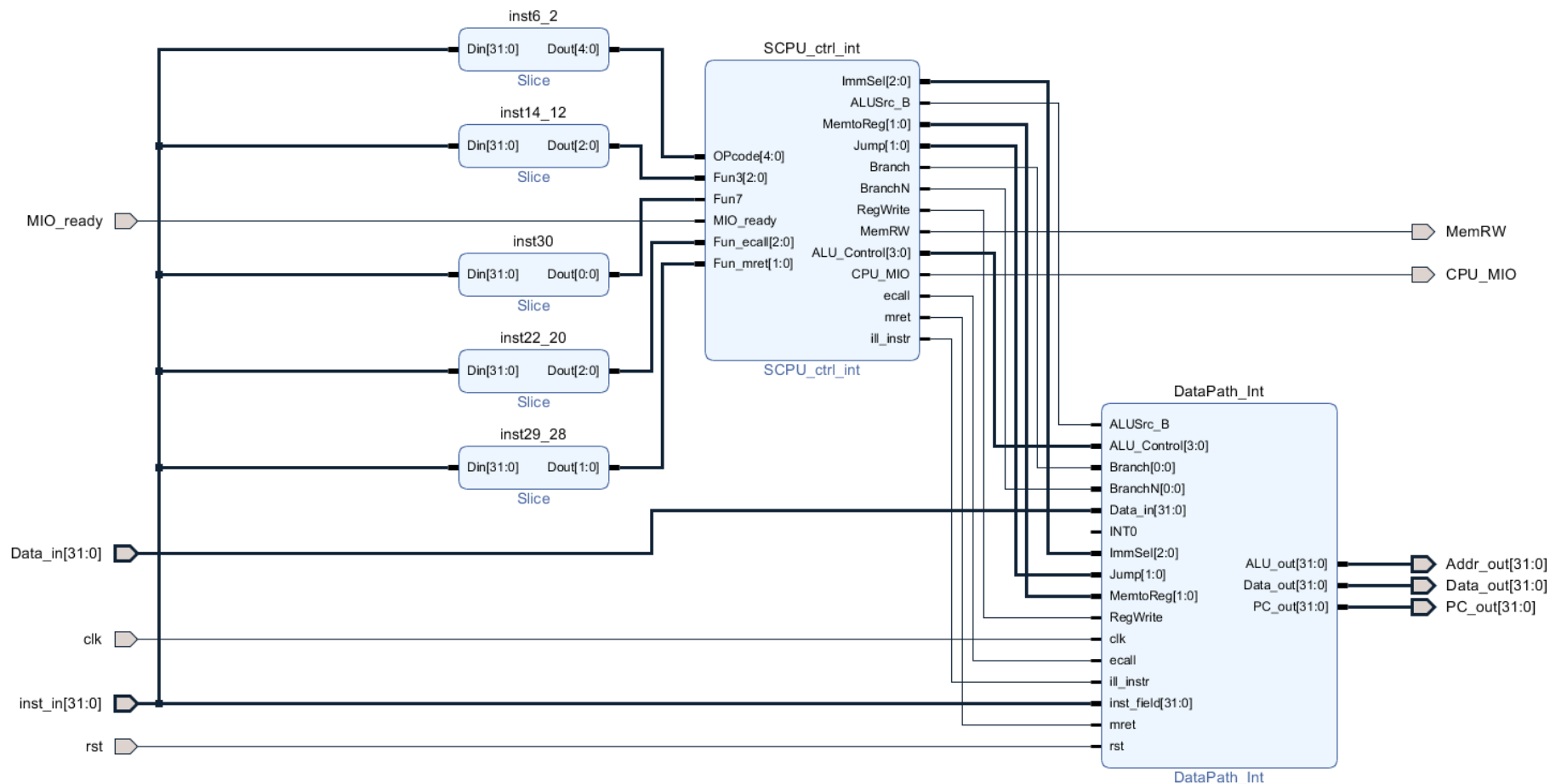
### ㊦ 物理验证

- ⊙ 执行非法指令或ecall指令或采用BTN[1]外部触发中断
- ⊙ 观察PC由顺序执行流转向中断向量表，进而执行相应中断服务程序，最后返回断点继续顺序执行

# 增加简单中断后的DataPath



# 增加中断后的CPU模块



- 
- 任务二：设计CPU中断测试方案并完成测试

# CPU调试与测试

---

- 使用**DEMO**程序目测控制器功能(方法同lab04-3)
  - DEMO程序也可自己另外编写
  
- 用汇编语言设计测试程序
  - 测试ALU指令(R-格式译码、Function译码)
  - 测试LW指令(I-格式译码)
  - 测试SW指令(S-格式译码)
  - 测试分支指令(B-格式译码)
  - 测试转移指令(J-格式译码)
  - 测试中断模式（非法指令、ecall译码、eret译码、Int外部触发）



# CPU调试与测试

## □ 调试

- SCPU\_ctrl\_int模块仿真
  - 设计测试激励代码仿真测试\*
- Data\_path\_int模块仿真
  - 设计测试激励代码仿真测试\*
- CPU功能仿真（仿真测试平台参见lab04-2）
- **请尽量先仿真正确再进行SOC集成**

若含有提供的  
EDF格式IP则无  
法仿真

也可直接调  
用.v形式的子  
模块

## □ 集成替换

- 仿真正确后逐个替换Exp04-3的相应模块
- 使用DEMO程序（或另外编写）目测控制器正常运行
  - DEMO程序与前面实验不一样

```
memory_initialization_radix=16;
memory_initialization_vector=
0200006F, 0C40006F, 0D80006F, 0C80006F, 00000033, 00000033, 00000033, 00000033, 00007293, 00007313,
88888137, FE62DAE3, 00832183, 0032A223, 00402083, 01C02383, 00338863, 555550B7, 0070A0B3, FE0098E3,
007282B3, 00230333, 00531463, 40000033, 40530433, 405304B3, 0080006F, 00007033, 0072F533, 00000073,
00157593, 00B51463, 00006033, 00A5E5B3, 0015E513, 00558463, 00004033, 00A5C633, 00164613, 00B61463,
00000013, 0012D293, 00060463, 40000033, 00129293, 00B28463, 00000013, 001026B3, 00503733, F5DFF06F,
00168693, 00168693, 30200073, 40C70733, 40C70733, 30200073, 00128793, 00178793, 30200073;
```

# 设计测试记录表格

---

- CPU指令测试结果记录
  - 自行设计记录表格

# 思考题

---

- 设计SCPU\_ctrl\_int时，所增加的特权指令与RV32I基本指令译码有何区别？
- 指令集规定的中断寄存器mepc、mtvec均为时钟控制寄存器，本实验在设计RV\_int时若也都设计为寄存器，能实现PC的正常跳转吗？

---

 **END**