计算机设计与实践
例外和中断处理&下板

2024 · 夏

HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

实验目的

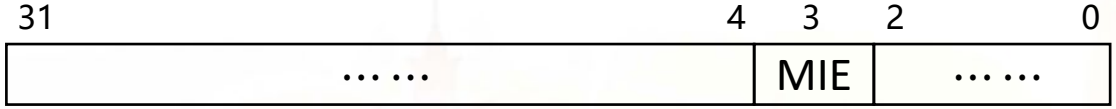
- 理解例外和中断的原理，掌握其实现方法
- 掌握下板调试的基本方法

实验内容

- 为单周期CPU设计实现例外和中断处理功能（不考虑嵌套中断、*选做*）
  - ① 实现CSR寄存器：**MSTATUS**、**MEPC**、**MCAUSE**
  - ② 修改取指数据通路，发生例外和中断时，统一跳转到 **0x1C090000**
  - ③ 扩展IROM模块，增加 **IROM_t** 存放例外和中断处理程序代码
  - ④ 实现 **csrrwi**、**mret**、**ecall** 指令
- 调试、下板并运行汇编实验的题目2

实验原理 – CSR寄存器

- **MSTATUS**



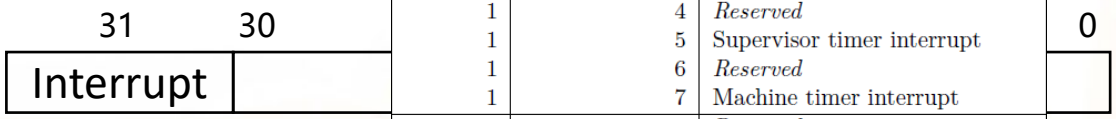
  - 记录特权等级、中断屏蔽等信息
  - CPU默认只有Machine模式，不需考虑特权等级的切换
  - 第3位 **MIE** 控制全局中断：1-开启全局中断，0-关闭全局中断

- **MEPC**

  - 记录被打断的程序控制流地址
  - 指令本身执行出错：记录该指令地址
  - 外部中断、系统调用：记录当前指令**下一条**指令的地址 (不一定是PC+4)

实验原理 – CSR寄存器

- **MCAUSE**
  - 记录例外和中断的具体原因
  - 第31位记录是否发生外部中断：
    - 1-外部中断
    - 0-同步例外/同步异常
  - 第30位~第0位记录具体原因
  - 只需考虑外部中断、ecall



Interrupt	Exception Code	Description
1	0	*Reserved*
1	1	Supervisor software interrupt
1	2	*Reserved*
1	3	Machine software interrupt
1	4	*Reserved*
1	5	Supervisor timer interrupt
1	6	*Reserved*
1	7	Machine timer interrupt
1	8	*Reserved*
1	9	Supervisor external interrupt
1	10	*Reserved*
1	11	Machine external interrupt
1	12–15	*Reserved*
1	≥16	*Designated for platform use*
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
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10	*Reserved*
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	14	*Reserved*
0	15	Store/AMO page fault
0	16–23	*Reserved*
0	24–31	*Designated for custom use*
0	32–47	*Reserved*
0	48–63	*Designated for custom use*
0	≥64	*Reserved*

实验原理 – 例外和中断指令

1. Zicsr扩展指令

RV32/RV64 *Zicsr* Standard Extension

31 20	19 15	14 12	11 7	6 0	
csr	rs1	001	rd	1110011	CSRRW
csr	rs1	010	rd	1110011	CSRRS
csr	rs1	011	rd	1110011	CSRRC
csr	uimm	101	rd	1110011	CSRRWI
csr	uimm	110	rd	1110011	CSRRSI
csr	uimm	111	rd	1110011	CSRRCI

   - RISC-V通过Zicsr扩展指令读写CSR寄存器（实现 `csrrwi` 即可）

2. 例外和中断返回指令 `mret`
   - 从例外和中断处理程序返回到MEPC处的指令

3. 系统调用指令 `ecall`
   - 发起同步例外，通过a7传递系统调用号，a0、a1等传递参数

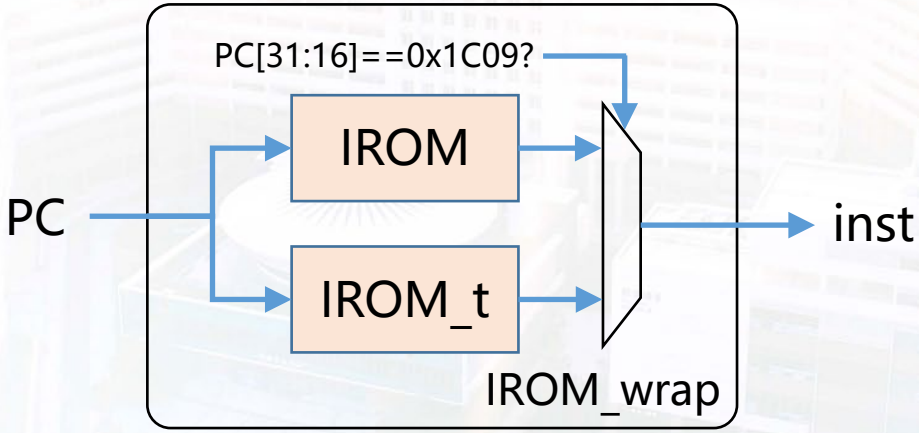
实验原理 – 例外和中断处理过程

1. 发生
   - 例外：执行 `ecall` 指令，**控制器**通过**译码**发出例外信号
   - 中断：外部中断被触发 & 全局中断打开 & 当前执行的并非原子指令
   - MEPC <- Next PC；MCAUSE <- 例外和中断原因；PC <- 0x1C090000
2. 处理
   - 例外和中断处理程序**关闭全局中断** -> 保护现场

     -> 读取**MCAUSE**，根据原因跳转到相应的处理程序
3. 返回
   - 恢复现场 -> **打开全局中断** -> 执行 `mret` 返回

实验步骤

1. 扩展IROM模块
   - 新增IROM_t模块，存放例外和中断处理程序**trap_handle.coe**
   - 将现有的IROM与IROM_t进行封装，通过地址区分从哪输出指令



实验步骤

2. 修改控制器
   - 增加 csrrwi、mret、ecall 指令的译码逻辑
     - csrrwi：译码产生**屏蔽外部中断**的信号
     - mret：译码产生**返回**信号
     - ecall：译码产生**同步例外**信号
3. 修改NPC部件
   - 同步例外或中断时，输出Next PC为0x1C090000
   - 例外和中断返回时，输出Next PC为MEPC

实验步骤

4. 新增CSR模块

- 包含MSTATUS、MEPC、MCAUSE
- CSR寄存器可能通过硬件更新，可能通过软件更新
  - 发生例外和中断时，硬件自动更新
  - 执行csrrwi指令时，读取或修改CSR寄存器的值
  - csrrwi指令根据地址访问特定的CSR寄存器

CSR地址	CSR名称
0x300	MSTATUS
0x341	MEPC
0x342	MCAUSE

- `csrrwi rd, 0x300, 8  # (rd) <- (MSTATUS); (MSTATUS) <- 8`

实验步骤

5. 修改按键开关外设
   - 将**S5**作为外部中断源intr，即将按键开关外设改成只控制S4~S0

6. 修改CPU及SoC顶层模块
   - 将intr信号作为输入添加到SoC顶层模块和CPU顶层模块

下板要求

- 在单周期CPU跑过Trace后再调试下板
- 要求至少为数码管、拨码开关、按键开关、LED设计I/O接口电路
- 要求在SoC上运行实验1编写的汇编程序题目2

下板调试

- 下板跑Trace
  - 使用提供的down_test.zip在开发板上运行Trace验证
- FPGA在线调试
  - ILA (Integrated Logic Analyzer) 抓取电路波形

- 注意代码规范性，可以避开很多问题！

开始实验