

实验五 中断处理实验

1. 实验介绍

除了通常的运算功能之外，任何处理器都需要一些部件来处理中断、配置选项以及需要某种机制来监控诸如片上高速缓存和定时器等功能。在 MIPS CPU 中，异常或者中断发生时的行为以及怎样处理都是由 CP0 控制寄存器和几条特殊指令来定义和控制的。本实验不需要实现 CP0 的全部功能，只关注 CP0 的寄存器读写和对中断异常的处理部分。通过本次试验，你可以了解 CP0 异常处理的实现原理，并学习如何实现 CP0 对异常处理的处理。

2. 实验目标

- 了解 MIPS 架构中 CP0 中断异常的实现原理；
- 使用 Verilog 实现 54 指令 CPU 中的 break、syscall、teq、eret、mfc0、mtc0，实现 CP0 异常处理；
- 可选择进一步实现外部中断。

3. 实验原理

1) CP0 寄存器

- 主要的 CP0 寄存器：

CP0 中有一系列寄存器来完成异常控制、缓存控制等功能，见表 1。

表 1 CP0 中的寄存器描述

标号	寄存器助记符	功能描述
0	Index	TLB 阵列的入口索引
1	Random	产生 TLB 阵列的随机入口索引
2	EntryLo0	偶数虚拟页的入口地址的低位部分
3	EntryLo1	奇数虚拟页的入口地址的低位部分
4	Context	指向内存虚拟页表入口地址的指针
5	PageMask	控制 TLB 入口中可变页面的大小
6	Wired	控制固定的 TLB 入口的数目
7	保留	
8	BadVAddr	记录最近一次地址相关异常的地址
9	Count	处理器计数周期
10	EntryHi	TLB 入口地址的高位部分
11	Compare	定时中断控制
12	Status	处理器状态和控制寄存器，包括决定 CPU 特权等级，使能哪些中断等字段
13	Cause	保存上一次异常原因
14	EPC	保存上一次异常时的程序计数器
15	PRId	处理器标志和版本
16	Config	配置寄存器，用来设置 CPU 的参数

17	LLAddr	加载链接指令要加载的数据存储器地址
18	WatchLo	观测点 watchpoint 地址的低位部分
19	WatchHi	观测点 watchpoint 地址的高位部分
20-22	保留	
23	Debug	调试控制和异常状况
24	DEPC	上一次调试异常的程序计数器
25	保留	
26	ErrCtl	控制 Cache 指令访问数据和 SPRAM
27	保留	
28	TagLo/DataLo	Cache 中 Tag 接口的低位部分
29	保留	
30	ErrorEPC	上一次系统错误时的程序计数器
31	DESAVE	用于调试处理的暂停寄存器

在实验中我们只实现 CP0 的中断、异常处理功能，因此只需要部分寄存器，以下是实验中用到的主要寄存器：

status: 12 号寄存器

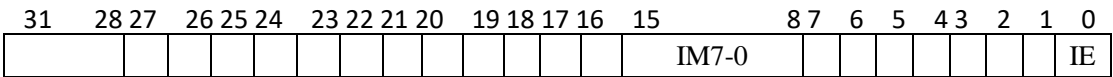


图 1 MIPS status 寄存器

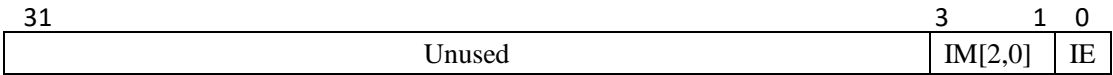


图 2 实现的 status 寄存器

status[0] 'IE' 为中断禁止位，标准的 MIPS CP0 中 status 寄存器的 8 到 15 位为中断屏蔽位，如图 1，本实验中将 status[3:1]定为中断屏蔽位，status[1] 屏蔽 syscall, status[2]屏蔽 break，status[3]屏蔽 teq，如图 2。置 0 时表示禁止或屏蔽中断，我们不实现中断嵌套，所以相应中断异常时把 Status 寄存器的内容左移 5 位关中断，中断返回时再将 Status 寄存器右移 5 位恢复其内容。

cause: 13 号寄存器

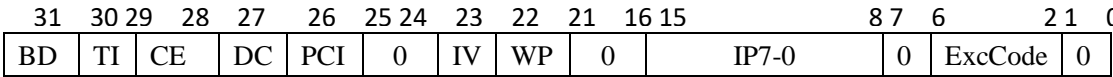


图 3 MIPS cause 寄存器

Cause 寄存器用于存放异常原因，MIPS32 架构的 cause 寄存器如图 3 所示，实验中仅用到 cause[6:2]: 异常类型号 'ExcCode'，1000 为 syscall 异常，1001 为 break，1101 为 teq。

epc: 14 号寄存器

异常发生时 epc 存放当前指令地址作为返回地址。

2) 异常中断控制

在异常、中断控制功能的实现中，我们做如下规定：

- 实现的异常包括断点指令 break 和系统调用 syscall 以及自陷指令 teq;
- 采用查询中断;
- 异常发生时保存当前指令的地址作为返回地址;
- 响应异常时把 Status 寄存器的内容左移 5 位关中断;
- 执行中断处理程序时保存 Status 寄存器内容，中断返回时写回。
- 异常入口地址为 0x4 （注：因本实验实现的 CPU 空间问题，无法做到 MIPS 规范标准中的异常入口地址，若要将 CPU 用到具体项目中时，请自行参考 MIPS 标准对此进行相应修改）

● 接口定义：

以下是一个实现 break、syscall、teq、eret 的 CP0 接口，仅供参考。

```
module CP0(  
    input clk,  
    input rst,  
    input mfc0,  
    input [4:0] addr,  
    output [31:0] CP0_out,  
    input mtc0,  
    input wcau,  
    input wsta,  
    input wepc,  
    input [31:0] data,  
    input exception,  
    input [31:0] pc,  
    input [31:0] cause,  
    output [31:0] status,  
    output [31:0] epc_out  
);
```

● 数据通路图：

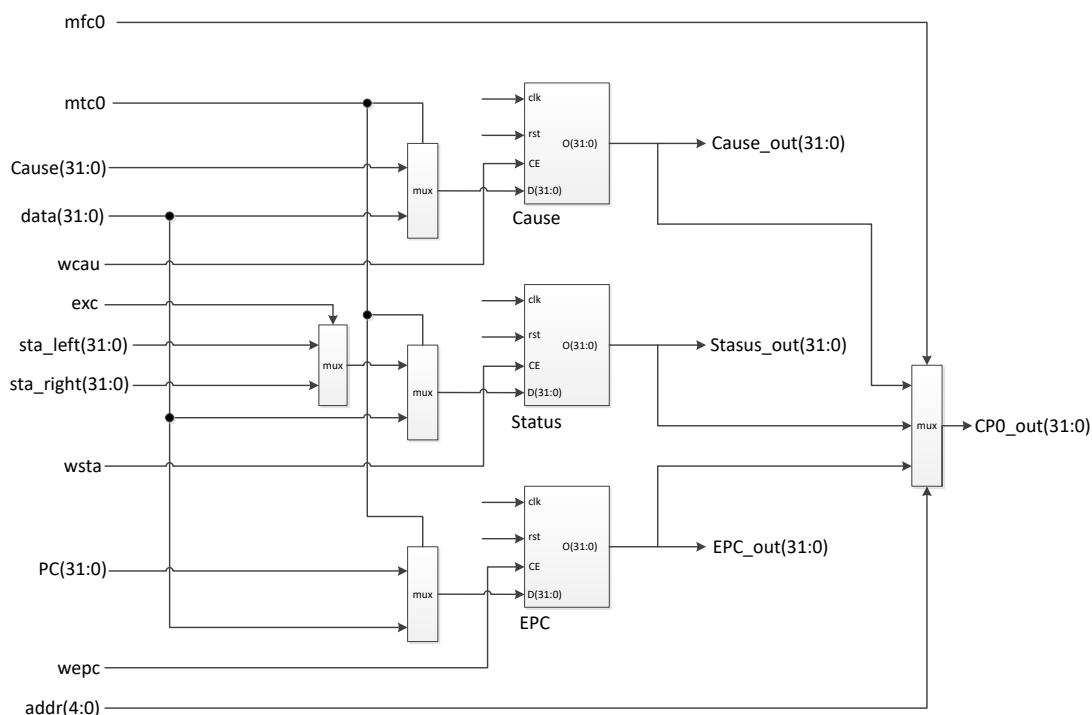


图 4 数据通路图

如图 4 所示为一个实现中断控制的 CP0 数据通路图。

mfc0 读 cp0 指令，mfc0 信号置 1，通过 addr 的值选择 Cause_out, Status_out, EPC_out 这三个寄存器值中的一个作为输出数据 CP0_out。例：mfc0 \$k0, \$13 这条指令，addr 即\$13 对应的机器码，输出到\$k0 中的是\$13 也就是本实验中 cause 寄存器的值。

mtc0 写 cp0 指令，mtc0 置 1，将 CP0 中的寄存器输入数据都选择为 data(31:0)，data 是寄存器堆传入到 CP0 中的数据。同时控制器生成 wcau、wsta、wepc 三个信号，通过 addr 使其中一个为 1，其余为 0，来控制写入哪一个寄存器；

当指令是 syscall、break、teq 时，mtc0 信号为 0，wsta 信号为 1，expression 信号为 1（只有当（中断进制位为 1 && （该指令为中断指令 && 相应中断屏蔽位为 1））这个条件成立时 exception 信号才为 1，例如：break 指令时，status 寄存器中 0 位和 2 位为 1 时 exception 为 1，所以这里需要将 status 的数据输出到控制模块）。此时，让 status 中的数据左移一定位数，使得中断屏蔽位和中断禁止位为 0，该操作即关中断，不再接受其他中断；将 pc，即指令地址存入 EPC 寄存器中，以便之后返回；wcau 的信号为 1，将 cause 存入 cause 寄存器中（数据 cause 在控制器中生成，例如 break 的 cause 为 {28'd0, 1001}）。

eret 指令时，exception 信号为 0，mtc0 信号为 0，wsta 信号为 1。此时，让 status 中的数据右移一定位数，使得中断屏蔽位和中断禁止位恢复之前的状态，重新接受中断请求，将 epc 寄存器中的数据输出到主 CPU 的 PC 中（在写 CP0 时不需要将 epc 寄存器中的地址+4 之后返回给 PC）。

● 异常类型:

在 MIPS32 架构中, 有一些事件要打断程序的正常执行流程, 这些事件有中断 (Interrupt)、陷阱 (Trap)、系统调用 (System Call) 以及其他任何可以打断程序正常执行流程的情况, 统称为异常。异常类型及其优先级如表 2 所示。

表 2 MIPS32 架构中定义的异常类型及其优先级

优先级	异常		描述
1	Reset		硬件复位
2	Soft Reset		在发生致命错误后对系统的复位，是软复位
3	DSS		Debug Single Step 单步调试
4	DINT		Debug Interrupt 调试中断
5	NMI		不可屏蔽中断
6	Machine Check		发生在 TLB 入口多重匹配对
7	Interrupt		发生在 8 个中断之一被检测到时，包括 6 个外部硬件中断、2 个软件中断
8	Deferred Watch		与观测点有关的异常
9	DIB		Debug Hardware Instruction Bread Match，指令硬件断点和正在执行的指令相符合
10	WATCH		取指地址与观测寄存器中的地址相同时发生
11	AdEL		取指地址对齐异常
12	TLB Refill	TLBL	指令 TLB 失靶
13	TLB Invalid		指令 TLB 无效
14	IBE		取指令总线错误
15	DBp		断点，执行了 SDBBP 指令
16	Sys		执行了系统调用指令 syscall
	Bp		执行了 break 指令
	CpU		在协处理器不存在或不可用的情况下执行了协处理器指令
	RI		无效指令
	Ov		算术操作指令 add、addi、sub 运算溢出
	Tr		执行了自陷指令
17	DDBL/DDBS		存储过程中，数据地址断点或数据值断点
18	WATCH		数据地址与观测寄存器中的地址相同时
19	AdEL		加载数据的地址未对齐
	AdES		存储数据的地址未对齐

20	TLB Refill	TLBL TLBS	数据 TLB 失靶
21	TLB Invalid		数据 TLB 无效
22	TLB Mod		对不可写的 TLB 进行了写操作
23	DBE		加载存储总线错误
24	DDBL		加载的数据与硬件断点设置的数据相等

实验中我们仅实现 Sys、Bp、Tr 和一个外部中断 Interrupt。相关的指令包括 syscall、break、teq、eret。

31	26	25		6	5		0
000000	Code					001100	

图 5 syscall 指令的格式

系统调用指令 syscall 的格式如图 5 所示。code 字段在译码过程中没有作用。MIPS32 架构定义了两种工作模式：用户模式和内核模式，在本实验中不对工作模式进行区分，没有对操作进行限制。

自陷指令有 12 条，实验中仅实现一条 teq 指令，TEQ rs, rt 为条件异常指令，若 rs 寄存器的值和 rt 寄存器相等，则引发自陷异常。

Break 为断点异常，实验中仅实现对断点异常跳转和返回处理。

异常返回指令 eret 的作用为从异常处理程序中返回，执行该指令使 EPC 寄存器的值成为新的取指地址，并恢复 status 寄存器的异常屏蔽位。

当有外部中断请求 intr，cpu 要发出中断确认信号 inta，通过读取 cause 寄存器中的 ExcCode 和 IP 内容来进行相应的中断处理，excCode 为 0 表示发生外部中断。

4. 实验步骤

1. 新建工程，参考以上接口定义；
2. 实现 CP0 寄存器读写；
3. 实现 break、syscall、teq 的异常跳转和 eret 的异常返回；
4. 实现外部中断；
5. 对 CP0 模块的测试在下一节介绍。