7.5 MIPS CPU中断

同濟大學
TONGJI UNIVERSITY

目录 | CONTENT

7.1.3 CP0

- 在MIPS体系结构中，最多支持4个协处理器(Co-Processor)。其中，协处理器CP0是体系结构中必须实现的。MMU、异常处理、乘除法等功能，都依赖于协处理器CP0来实现。
- MIPS的CP0包含32个寄存器。本课程仅讨论常用的一些寄存器，如表7.1.2所示。

表7.1.2 CP0常用寄存器

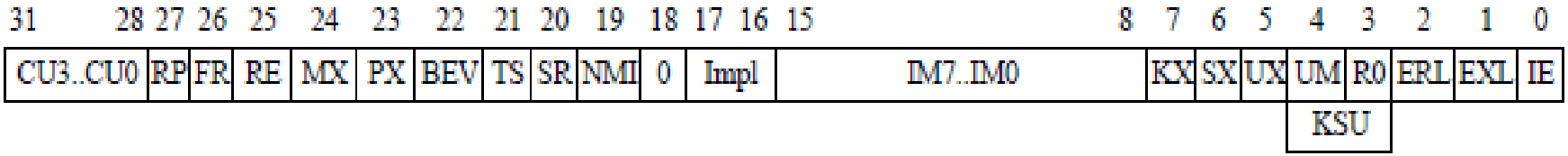
标号	寄存器助记符	功能描述
0	Index	TLB阵列的入口索引
1	Random	产生TLB阵列的随机入口索引
2	EntryLo0	偶数虚拟页的入口地址的低位部分
3	EntryLo1	奇数虚拟页的入口地址的低位部分
4	Context	指向内存虚拟页表入口地址的指针
5	PageMask	控制TLB入口中可变页面的大小
6	Wired	控制固定的TLB入口的数目
7	保留	
8	BadVAddr	记录最近一次地址相关异常的地址

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

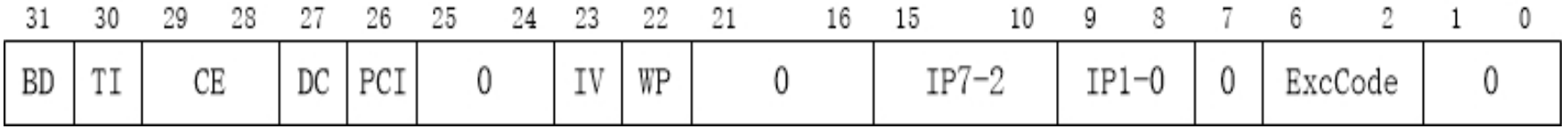
Status

- 这个寄存器标识了处理器的状态，其中status[0]为中断禁止位，status[15:8]为中断屏蔽位。



Cause

- 在处理器异常发生时，这个寄存器标识出了异常的原因。其中，最重要的是从Bit2到Bit6，5个Bit的Excetion Code位。它们标识出了引起异常的原因



EPC

- 这个寄存器的作用很简单，就是保存异常发生时的指令地址。从这个地方可以找到异常发生的指令，再结合BadVAddr, sp, ra等寄存器，就可以推导出异常时的程序调用关系，从而定位问题的根因。

WatchLo/WatchHi

- 这一对寄存器可以用来设定“内存硬件断点”，也就是对指定点的内存进行监测。当访问的内存地址和这两个寄存器中地址一致时，会发生一个异常。

CP0主要操作

- **mfc0 rt, rd　　将CP0中的rd寄存器内容传输到rt通用寄存器；**
- **mtc0 rt, rd　　将rt通用寄存器中内容传输到CP0中寄存器rd；**

CP0冒险现象

- MIPS体系结构是一个无互锁，高度流水的五级pipeline架构，这就意味着，前一条指令如果尚未执行完，后一条指令有可能已经进入了取指令/译码阶段。这样，就有可能发生所谓的CP0冒险（CP0 Hazard）现象。简单地说，就是mfc和mtc指令的执行速度是比较慢的，因此，开始执行完下一条指令时，有可能CP0寄存器的值尚未最后传输到指定的目标通用寄存器中。此时，如果读取该通用寄存器，有可能并未得到正确的值。 这就是所谓的CP0冒险现象。

- 为了避免CP0冒险，我们在编程时需要在CP0操作指令的后面加上一条与前一条指令的目的通用寄存器无关的指令，也就是所谓的延迟槽（delay slot）。如果对性能不敏感，可以考虑用一条nop空操作指令填充延迟槽。

7.1.4 MIPS CPU中断机制

MIPS异常

- 在MIPS中，中断、陷阱、系统调用和任何可以中断程序正常执行流的情况都称之为异常。
- 精确异常：在引发异常的指令执行时，后面一条指令已经完成了读取和译码的预备工作，当异常产生时，这些预备工作被废弃，CPU从异常中返回时，再重新做读取和译码的工作。
- **MIPS对异常的处理是给异常分配一些类型，然后由软件给它们定义一些优先级，然后由同一个入口进入异常分配程序，在分配程序中根据类型及优先级确定该执行哪个对应的函数。**
- CP0中的EPC寄存器用于指向异常发生时指令跳转前的执行位置，一般是被中断指令地址。当异常时，是返回这个地址继续执行。但如果被中断指令在分支延迟槽中，则会硬件自动处理使EPC往回指一条指令，即分支指令。在重新执行分支指令时，分支延迟槽中的指令会被再执行一次。

- MIPS异常处理步骤

①设置EPC，指向返回的位置

②设置Status寄存器，EXL位迫使CPU进入内核模式（高特权级）并且禁用中断

③设置Cause寄存器，使得软件能看到异常原因。地址异常时，也要设置BadVAddr寄存器，存储管理系统异常还要设置一些MMU寄存器。

④CPU从异常入口点取指令执行。

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

- MIPS异常处理例程

① 保存现场：在异常处理例程入口，需要保护被中断的程序的现场，存储寄存器的状态，保证关键状态不被覆盖。

② 处理异常：根据Cause：ExcCode确定发生了什么类型的异常，完成想要做的任何事情。

③ 准备返回：恢复现场，修改status寄存器，设置成安全模式（内核态，禁止异常），也就是异常发生后的模式。

④ 从异常返回：用指令“eret”，既清除SR：EXL位，也将控制权返回给存储在EPC中的地址。

□MIPS中断

- CPU核外部的事件，即从一些真正的“硬件连线”过来的输入信号（外中断或硬中断），Cause：ExcCode编码为00000中断。这些中断使CPU转向某外部事件，MIPS CPU有8个独立的中断位（在Cause：IP7~2和IP1~0段），其中，6个（IP7~2）为外部中断，2个（IP1~0）为内部中断（可由软件访问），片上的时钟计数 / 定时器都会连接到一个硬件中断位上。

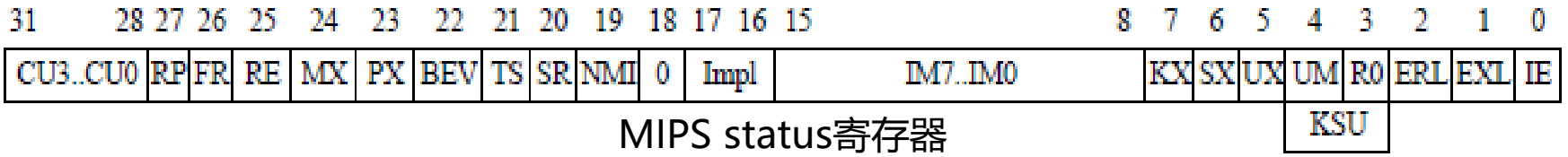
31	30	29 28	27	26	25 24	23	22	21 16	15 10	9 8	7	6 2	1 0
BD	TI	CE	DC	PCI	0	IV	WP	0	IP7-2	IP1-0	0	ExcCode	0

MIPS cause寄存器

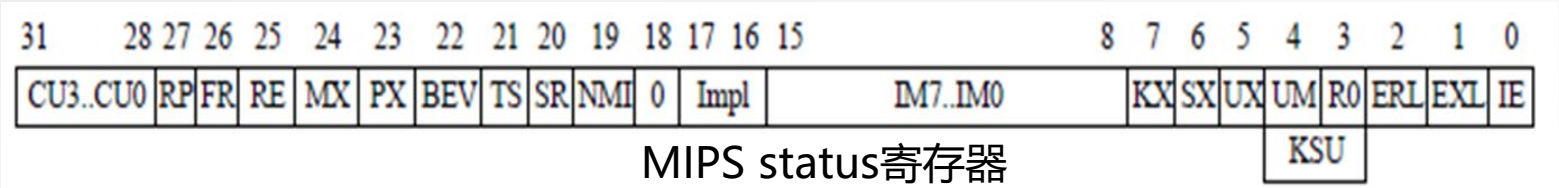
- MIPS CPU对中断的支持涉及CP0的两个重要的寄存器：状态寄存器（SR）和原因寄存器（Cause）。

① 使能全局中断（IE：Interrupt Enable）：要想使能中断，则全局中断位SR：IE必须置1，开中断，它是一个全局开关。

② 中断使能屏蔽（IM：Interrupt Mask）：SR[15~8]为中断屏蔽位，对应IM[7~0]，这8个bit位决定了哪些中断源有请求时可以触发一个异常，实际上是对中断信号的使能开关。8个中断源中的6个（IM[7~2]）对应Cause：IP7~2，用于外部硬件设备中断，其它2个（IM[1~0]）对应Cause：IP1~0，为软件中断屏蔽位。所谓中断源就是产生硬中断信号的PIC外接设备或者软中断。



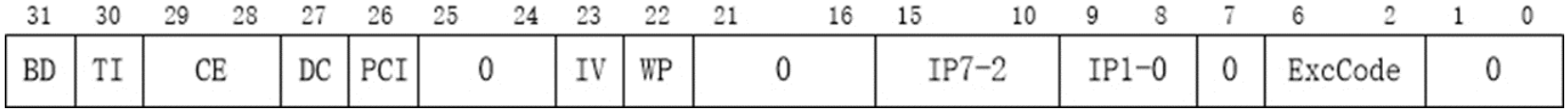
MIPS status寄存器



MIPS status寄存器

③ 异常级别（EXL：Exception Level）：异常发生后，CPU立即设置SR：EXL，进入异常模式。异常模式强制CPU进入内核特权级模式并屏蔽中断，而不会理会SR其它位的值。EXL位在已设置的情况下，还没有真正准备好调用主内核的例程。在这种状态下，系统不能处理其他异常。保持EXL足够的时间保存现场，使软件决定CPU新的特权级别和中断屏蔽位应该如何设置。

④ 异常类型（ExcCode：Exception Code）：PIC每个输入引脚上的有效（IM位为1，未被屏蔽）输入每个周期都会被采样，如果被使能，则引起一个异常。异常处理程序检查到Cause：ExcCode=0，则说明发生的异常是中断，此时将进入通用中断处理程序。



MIPS cause寄存器

⑤ 中断请求寄存器（IP：Interrupt Pending）：Cause[15~8]为中断挂起状态位，用于指示哪些设备发生了中断，具体来说就是识别PIC的哪个接入引脚对应的设备发来了中断信号。IP7~2随着CPU硬件输入引脚上的信号而变化，而IP1~0为软件中断位，可读可写并存储最后写入的值。当SR：IM[7~0]某些位使能，且硬中断或软中断触发时，Cause：IP[7~0]对应位将被置位，一般通过查询Cause：IP[7~2]的pending位，确定哪个设备发生了中断。

⑥ 中断处理步骤（Interrupt Handle Procedure）中断是异常的一种，所以中断处理只是异常处理的一条分流。经过上一层异常处理例程处理后，其处理步骤如下：

a) 将Cause：IP与SR：IM进行逻辑与运算，获得一个或多个使能的中断请求。

b) 选择一个使能的中断来处理，优先处理最高优先级的中断。

c) 存储SR：IM中的中断屏蔽位，改变SR：IM，以保证禁止当前中断以及所有优先级小于等于本中断的中断在处理期间产生。

d) 对于嵌套异常，则此时需要保护现场。

e) 修改CPU到合适的状态以适应中断处理程序的高层部分，这时通常允许一些嵌套的中断或异常。设置全局中断使能SR：IE位，以允许处理高优先级的中断。还需要改变CPU特权级域（SR：KSU），使得CPU处于内核态，清除SR：EXL以离开异常模式，并把这些改动反映到状态寄存器中。

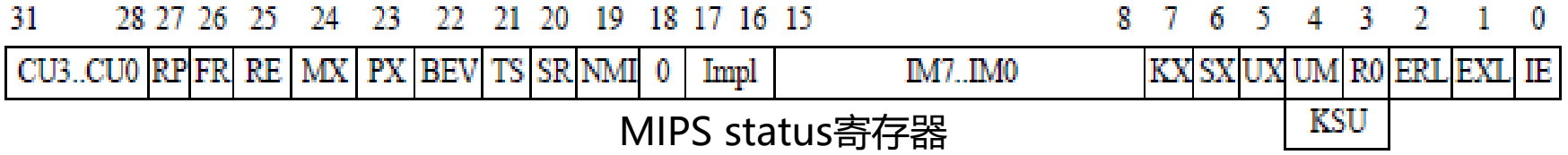
f) 执行中断处理程，完成要做的事情。

g) 恢复现场，恢复相关寄存器，返回被中断指令（返回被中断程序）。

CP0实验

实验涉及的 CP0寄存器

标号	寄存器助记符	功能描述
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



MIPS status寄存器

Status

- 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8.5.1，本实验中将status[10:8]定为中断屏蔽位，status[8] 屏蔽systcall, status[9]屏蔽 break, status[10]屏蔽teq。

31	30	29 28	27	26	25 24	23	22	21 16	15 10	9 8	7	6 2	1 0
BD	TI	CE	DC	PCI	0	IV	WP	0	IP7-2	IP1-0	0	ExcCode	0

MIPS cause寄存器

Cause

- Cause寄存器用于存放异常原因，实验中仅用到cause[6:2]：异常类型号 ‘ExcCode’ ，01000为systcall异常，01001为break，01101为teq

EPC

- 异常发生时 epc 存放当前指令地址作为返回地址。

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

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。
- 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、 mfc0 、mtc0。

BREAK

格式：BREAK

目的：引起一个断点异常

描述：当一个断点异常发生时，将立刻并且不可控制的转入异常处理。

操作：

SignalException(Breakpoint)

注：CP0中的cause寄存器要有硬件通路支持写入正确的异常类型码，同时要有硬件通路支持CP0中的EPC寄存器写入PC+4（异常返回地址，即break指令的下一条指令地址）

SYSCALL

格式：SYSCALL

目的：引发一个系统调用异常

描述：当一个系统调用发生时，将立刻并且不可控制的转入异常处理。

操作：

SignalException(SystemCall)

注：CP0中的cause寄存器要有硬件通路支持写入正确的异常类型码，同时要有硬件通路支持CP0中的EPC寄存器写入PC+4（异常返回地址，即break指令的下一条指令地址）

TEQ

格式：TEQ rs, rt

目的：比较寄存器值根据条件引发异常

描述：比较寄存器rs和rt的值，若相等则引发一个自陷异常

操作：

```
if GPR[rs] = GPR[rt] then
SignalException(Trap)
endif
```

注：CP0中的cause寄存器要有硬件通路支持写入正确的异常类型码，同时要有硬件通路支持CP0中的EPC寄存器写入PC+4（异常返回地址，即break指令的下一条指令地址）

ERET

格式：ERET

目的：从异常中断返回

描述：在所有中断处理过程结束后返回到中断指令。ERET不执行下一条指令。

操作：

```
if StatusERL = 1 then
temp ← ErrorEPC
StatusERL ← 0
else
temp← EPC
StatusEXL ← 0
endif
...
...
if IsMIPS16Implemented() then
PC ← temp31..1 || 0
ISAMode ← temp0
else
PC ← temp
endif
LLbit ← 0
```

将CP0的EPC寄存器中的地址送PC寄存器，从异常返回主程序

MFC0

格式： MFC0 rt, rd （实现该格式，将CP0的rd寄存器内容送CPU的rt寄存器）

MFC0 rt, rd, sel

目的：把一个数据从特殊寄存器移到通用寄存器

描述：rt ←CPR[0,rd,sel]

由rd和sel选择协处理器0中的特殊寄存器，把它的内容转移到通 用寄存器rt中。

操作：

data ← CPR[0,rd,sel]

GPR[rt] ←data

MTC0

格式：MTC0 rt, rd （实现该格式，将CPU的rt寄存器内容送CP0的rd寄存器）

MTC0 rt, rd, sel

目的：把一个数据从通用寄存器移到特殊寄存器

描述：CPR[r0, rd, sel] ← rt

由rd和sel选择协处理器0中的特殊寄存器，把通用寄存器rt中的内容转移到特殊寄存器中。

操作：

data ← GPR[rt]

CPR[0,rd,sel] ← data

000000	Code	001100

syscall指令的格式

① 系统调用指令syscall的格式如上图所示。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。

② 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。

③ Break为断点异常，实验中仅实现对断点异常跳转和返回处理。

④ 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。

⑤ 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。

接口定义

```
module CP0(
input clk,
input rst,
input mfc0,          // CPU指令Mfc0
input mtc0,          // CPU指令Mtc0
input [31:0]pc,
input [4:0] Rd,        // 指定CP0寄存器
input [31:0] wdata,    // 数据从GP寄存器到CP0寄存器
input exception,
input eret,          // 指令ERET (Exception Return)
input [4:0]cause,
input intr,
output [31:0] rdata,    //数据从CP0寄存器到GP寄存器
output [31:0] status,
output reg timer_int,
output [31:0]exc_addr   // 异常起始地址
);
```

谢谢聆听

Thank You