**CP0设计文档**

1. **CP0主要功能**

CP0的主要负责功能如下：

* 配置CPU工作状态：通过读/写一个或一些内部存储器来改变根本的CPU特性（MSB -> LSB、LSB -> MSB）
* 高速缓存控制：集成缓存控制器，用于控制、读、写缓存
* 异常控制：异常发生的检测与处理
* 存储管理单元控制：存储区域的合理控制、管理与分配
* 其他：额外功能，如时钟、时间计数器、奇偶校验错误检测等

我们所需实现的功能只有：异常控制、时钟的记录以及时钟中断的产生。

注：高速缓存由于实现较为简单，且没有对于其错误检测的设计，因此无需催高速缓存控制进行设计。

1. **CP0寄存器功能说明**

协处理器CP0有32个32位寄存器，用于实现上述功能。

我们所需实现的寄存器编号：Count(9)、Compare(11)、Status(12)、Cause(13)、EPC（14）

下面对于所需实现的寄存器进行详细说明。

1. **Count寄存器（标号为9）**

31 0

|  |
| --- |
| Count |

图2-1 Count寄存器格式

表2-1 Count寄存器区域描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **区域** | | **描述** | **读/写** | **重置状态** |
| **名称** | **比特** |
| Count | 31:0 | 存储计数器的值 | R/W | 未定义 |

功能：处理器周期计数器

详细描述：初始状态为32’h0，每来一个时钟信号则自增1，当达到32位无符号数上限时（32’hffffffff），清零重新计数。

初始值：32h’0

1. **Compare寄存器（标号11）**

31 0

|  |
| --- |
| Compare |

图2-2 Compare寄存器格式

表2-2 Compare寄存器区域描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **区域** | | **描述** | **读/写** | **重置状态** |
| **名称** | **比特** |
| Compare | 31:0 | 用于与计数值比较的值 | R/W | 未定义 |

功能：定时中断控制

详细描述：与Count寄存器一起完成定时中断功能，当Count寄存器中的计数值与Compare寄存器中的值一样时，会产生定时中断。这个中断会一直保持，直到有数据被写入Compare寄存器。

初始值：32h’0

1. **Status寄存器（标号12）**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 28 | 27 | 26 | 25 | 2423 | 22 | 21 | 20 | 19 | 18 16 | 15 8 | 7 5 | 4 | 3 | 2 | 1 | 0 |
| CU3~CU0 | RP | R | RE | 0 | BEV | TS | SR | NMI | 0 | IM7~IM0 | R | UM | R | ERL | EXL | IE |

图2-3 Status寄存器格式（此次实验使用蓝色部分）

表2-3 Status寄存器区域描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **区域** | | **描述** | **读/写** | **重置状态** |
| **名称** | **比特** |
| CU3~CU0 | 31:28 | CU3~CU0协处理器使能信号 | R/W | 未定义 |
| IM7~IM0 | 15:8 | 中断屏蔽信号 | R/W | 未定义 |
| EXL | 1 | 是否处于异常级 | R/W | 未定义 |
| IE | 0 | 中断使能信号 | R/W | 未定义 |

功能：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

详细描述：实验使用标记的蓝色部分，其余位不进行定义。

CU3~CU0：分别控制协处理器CP3、CP2、CP1、CP0，0表示对应的协处理器不可用，1表示可用，由于本次只有CP0处理器，因此设置为4b’0001即可。

IM7~IM0：表示是否屏蔽相应的中断，0表示屏蔽，1表示不屏蔽，共8个中断源，其中6个为处理器外部硬件中断，2个为软件中断。中断是否能被处理器相应是由Status寄存器与Cause寄存器共同决定（Status.IM[i] & Cause.IP[i] == 1 && Status.IE == 1 则响应相应中断），初始设置为8h’ff即可。

EXL：是否处于异常级，异常发生时设置为1，表示处理器处于异常级，进入内核模式工作，且禁止中断，初始设置为1b’0即可。

IE：中断使能信号，1表示中断使能，0表示中断禁止，初始设置为1b’1即可。

初始值：32h’1000ff01

1. **Cause寄存器（标号13）**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 2928 | 27 | 26 | 2524 | 23 | 22 | 21 16 | 15 10 | 9 8 | 7 | 6 2 | 1 0 |
| BD | R | CE | DC | PCI | 0 | IV | WP | 0 | IP[7:2] | IP[1:0] | 0 | ExcCode | 0 |

图2-4 Cause寄存器格式（此次实验使用蓝色部分）

表2-4 Cause寄存器区域描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **区域** | | **描述** | **读/写** | **重置状态** |
| **名称** | **比特** |
| BD | 31 | 异常指令处于分支延迟槽 | R | 未定义 |
| IP[7:2] | 15:10 | 硬件中断挂起字段 | R | 未定义 |
| IP[1:0] | 9:8 | 软件中断挂起字段 | R/W | 未定义 |
| ExcCode | 6:2 | 异常种类记录 | R | 未定义 |

功能：用于记录最近一次异常发生的原因，控制软件中断请求。

详细描述：实验使用标记的蓝色部分，其余位不进行定义。

BD：当最近一次异常发生的指令处于分支延迟槽，且Status.EXL = 0则置为1，否则保持为0。

IP[7:2]：中断挂起字段，相应位用于指明外部硬件中断是否发生，1为发生，0为没有发生。IP[7]~IP[2]分别对应中断5~0号硬件中断。

IP[1:0]：中断挂起字段，对应软件中断，IP[1]~IP[0]分别对应中断1~0号软件中断。

ExcCode：用于记录发生哪种异常，其编码仅列出本实验所需的部分。

|  |  |  |
| --- | --- | --- |
| **ExcCode编码（十进制）** | **助记符** | **描述** |
| 0 | Int | 中断 |
| 8 | Sys | 系统调用指令Syscall |
| 10 | RI | 执行未定义指令引起异常 |
| 12 | Ov | 整数溢出异常 |
| 13 | Tr | 自陷指令引起异常 |

初始值：32h’0

1. **EPC寄存器（标号14）**

31 0

|  |
| --- |
| EPC |

图2-5 EPC寄存器格式

表2-5 EPC寄存器区域描述

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **区域** | | **描述** | **读/写** | **重置状态** |
| **名称** | **比特** |
| EPC | 31:0 | 存储异常返回地址 | R/W | 未定义 |

功能：异常程序计数器，用于储存异常返回地址

详细描述：一般情况下，存储的是发生异常的指令的地址用于异常处理完后的跳转（PC <- EPC）,若发生的异常指令位于延迟槽中，则其存储的是前一条指令的地址。

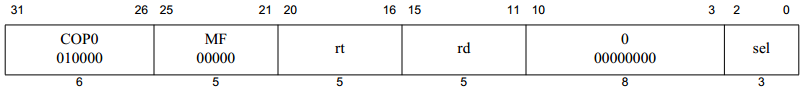
（原因：转移指令->延迟槽指令->转移目标地址的指令，若延迟槽指令发生异常，那么需要进行异常处理并返回继续执行，那么要进行跳转必须返回的是转移指令，否则若返回延迟槽指令那么不会进行跳转，而会继续执行下去，即执行延迟槽后的下一条指令）

初始值：32h’0

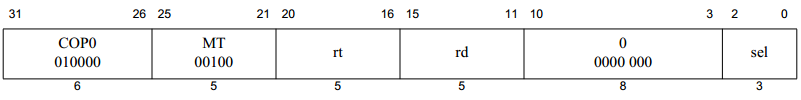
1. **有关指令实现说明**

此实验中直接与CP0有关的指令只有两个：mfc0与mtc0

**mfc0:**



**mtc0:**



此实验中，两个指令的sel的功能均不进行实现，因此只能选择整个32位寄存器并进行读写。

CPR[0, rd, sel]选择CP0哪个寄存器的计算在ID阶段。由于不需要对于sel功能进行实现，因此在ID阶段直接使用译码出来的rd域值，用于对于cp0的寄存器选择。

**读取：**

CP0处理器为时序电路，因此得到了CP0寄存器编号的同时，在ID阶段对CP0中的寄存器进行读取。由于写入是在WR阶段，因此可能会产生数据冒险，即在ID阶段要读CP0的同时，ID/EX，EX/MEM与MEM/WR流水段寄存器可能有对CP0的写入数据，此时需要将数据转发到ID阶段进行选择（注意：此处应当进行转发，但是实际我们实现时考虑用的较少，因此未进行转发，使用时应当在指令中避免出现转发的情况，适当添加nop指令）。由于CP0也有寄存器组，因此一般寄存器需要实现的转发功能在CP0处也需要进行相应的实现。

**写入：**

而在WR阶段对CP0中的寄存器进行写入。

1. **CP0模块接口说明**

CP0框架图（左侧均为输入，右侧均为输出）：

|  |  |  |
| --- | --- | --- |
| reset(1) | CP0 | (32)data\_o |
| clk(1) |  |
|  |  |
| w\_en\_i(1) | (32)status\_o |
| w\_addr\_i(5) | (32)cause\_o |
| data\_i(32) | (32)epc\_o |
|  |  |
| r\_addr\_i(5) | (1)time\_int\_o |
|  |  |
| int\_i(8) |  |
| excepttype\_i(32) |  |
| current\_inst\_addr\_i(32) |  |
| is\_in\_delayslot(1) |  |  |

CP0框架图接口说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 接口名 | 宽度(bit) | 输入/输出 | 作 用 |
| 1 | reset | 1 | 输入 | 复位 |
| 2 | clk | 1 | 输入 | 时钟信号，posedge有效 |
| 3 | w\_en\_i | 1 | 输入 | 写使能 |
| 4 | w\_addr\_i | 5 | 输入 | 写地址 |
| 5 | data\_i | 32 | 输入 | 要写入的数据 |
| 6 | r\_addr\_i | 5 | 输入 | 读地址 |
| 7 | int\_i | 8 | 输入 | 中断信号 |
| 8 | excepttype\_i | 32 | 输入 | 异常编码信号 |
| 9 | current\_inst\_addr\_i | 32 | 输入 | 精确异常处理记录的指令地址 |
| 10 | is\_in\_delayslot | 1 | 输入 | 标记目前指令是否在延迟槽 |
| 11 | data\_o | 32 | 输出 | CP0协处理器数据输出 |
| 12 | status\_o | 32 | 输出 | Status寄存器值，用于异常判断 |
| 13 | cause\_o | 32 | 输出 | Cause寄存器值，用于异常判断 |
| 14 | epc\_o | 32 | 输出 | Epc寄存器值，用于异常返回 |
| 15 | time\_int\_o | 1 | 输出 | 是否有定时中断发生 |

1. **MFC0/MTC0与CP0数据通路说明（蓝色部分）**

E:\学习文档\大三上\计算机组成与设计\流水线\设计文档\cu_cp0_异常交互.emf

MFC0/MTC0数据通路说明：蓝色线部分

MFC0指令，该指令从CP0读出数据写入到通用寄存器，因此从IF/ID流水段寄存器将地址传入CP0中，并进行CP0读取输出到通用寄存器选择端。

MTC0指令，该指令通过译码器得到写入地址以及使能信号，在通用寄存器组中得到写入数据，并在流水线段寄存器中向下传递，直到传递到MEM/WR流水线段寄存器，将值输入到CP0中进行写入。

转发：由于MFC0以及MTC0的数据均在流水线段寄存器中传递，因此可能会产生数据冒险，因此需要进行转发的处理，处理如上图所示。（需注意的是，由于MFC0以及MTC0使用较少，我们未对CP0数据的转发进行实现，因此要使用MFC0与MTC0时需要在前后加入适量的nop指令）

注：红色线部分为异常处理（异常信息）的数据通路，在“中断异常处理说明”部分进行详细描述。