一：异常

1.1 异常触发

|  |  |
| --- | --- |
| 异常触发流程 | 软件处理/硬件处理 |
| 1.产生的异常信号 | 硬件 |
| 2.对同一时刻同一阶段同一指令产生的所有异常进行判优 | 硬件 |
| 3.检测当前阶段是否处于IF阶段(判断) | 硬件 |
| 4.检测当前阶段是否处于MEM阶段（判断）（3的no分支） | 硬件 |
| 5.使MemWrite无效（4的yes分支） | 硬件 |
| 6.检测前一级流水线寄存器exception abort是否被置位（判断）(4的no分支) | 硬件 |
| 7.将上一级流水线中的异常编号和exception abort位写入下一级流水线寄存器的相应位置（6的yes分支） | 硬件 |
| 8.将本级流水线判优后的异常编号写入下一级流水线寄存器的相应位置（3的yes分支和6的no分支） | 硬件 |
| 9.将exception abort置成有效位，避免该指令在后续流水线阶段发生的异常覆盖已写入的异常编码（8的后续） | 硬件 |

注：下列编号从上到下，从左到右

异常触发过程由硬件完成，如下表1.1

表1.1异常触发过程

异常触发流程图如下图1-1



图1-1 异常触发流程图

1.2 异常检测

异常检测的流程如下表1.2，表1.3，表1.4

表1.2异常检测总体公共处理

|  |  |
| --- | --- |
| 异常检测总体公共流程 | 软件处理/硬件处理 |
| 1.检测MEM/WB流水线寄存器中的exception abort 是否被置位 | 硬件 |
| 清空IF/ID、ID/EXE、EXE/MEM流水线寄存器清空MEM/WB的exception abort清空所有的写控制信号 | 硬件 |
| 3.流水线正常执行 |  |
| 4.判断当前是否为内核模式 | 硬件 |
| 5.进入嵌套异常检测流程 |  |
| 6.进入普通异常检测流程 |  |

表1.3普通异常检测处理

|  |  |
| --- | --- |
| 普通异常检测流程 | 软件处理/硬件处理 |
| 1.将模式置为内核模式 | 硬件 |
| 2 判断是否为错误（判断）（1的后续） | 硬件 |
| 3.将cause、status中相应位置位（2的yes分支） | 硬件 |
| 4.检测Exception Code决定是否将相应地址写入BadVAddr寄存器 | 硬件 |
| 5.检测BD是否被置位（判断）(3的后续) | 硬件 |
| 6.将cause、status中相应位置位（4的后续） | 硬件 |
| 7.保存ErrorEPC为当前指令的地址（5的no分支） | 硬件 |
| 8.保存ErrorEPC为branch或jump的地址（5的yes分支） | 硬件 |
| 9.检测BD是否被置位(6的后续) | 硬件 |
| 10.保存EPC为当前指令地址 | 硬件 |
| 11.保存EPC为branch或jump的地址 | 硬件 |
| 12.判断错误类型 | 硬件 |
| 13.将PC置为统一异常处理入口地址 | 硬件 |
| 14.根据错误类型跳到相应的入口地址 | 硬件 |

表1.4普嵌套异常检测处理

|  |  |
| --- | --- |
| 嵌套异常检测流程 | 软件处理/硬件处理 |
| 1.关闭中断使能位 | 硬件 |
| 2.将当前的相关异常信息保存到CP0的影子寄存器中 | 硬件 |
| 3. 检测BD是否被置位 | 硬件 |
| 4.保存影子寄存器EPC和errorEPC为当前指令地址 | 硬件 |
| 5.保存影子寄存器EPC和errorEPC为branch或jump的地址 | 硬件 |
| 6.设置PC为统一嵌套异常处理入口地址 | 硬件 |

异常检测流程如下图1-2，图1-3，图1-4



图1-2 异常检测的总体公共流程图



图1-3 普通异常检测的流程图



图1-4嵌套异常检测流程

1.3 异常处理

异常处理的过程如下表1.5 ，表1.6

表1.5 普通异常处理

|  |  |
| --- | --- |
| 普通异常处理流程 | 软件处理/硬件处理 |
| 1.将EPC、ErrorEPC保存到系统栈中 | 软件 |
| 2.读取CP0中的相关寄存器 | 软件 |
| 3.检查异常类型 | 软件 |
| 4.通过异常类型计算对应异常处理入口地址 | 软件 |
| 5.跳转到相应的异常处理地址 | 软件 |
| 6.进行相应的异常处理 | 软件 |
| 7.跳转到统一异常返回程序 | 软件 |

表1.6 嵌套异常处理

|  |  |
| --- | --- |
| 嵌套异常处理流程 | 软件处理/硬件处理 |
| 1.保存所有相关寄存器 | 软件 |
| 2.将影子寄存器中的值写入cp0中相应的寄存器 | 软件 |
| 3.根据异常编号计算相应的处理程序入口地址 | 软件 |
| 4.进行相应的处理 | 软件 |
| 5.调用恢复程序 | 软件 |

异常处理流程图如下图1-5，图1-6



图1-5普通异常处理流程



图1-6嵌套异常处理流程

二：中断

2.1 中断触发

|  |  |
| --- | --- |
| 中断触发流程 | 软件处理/硬件处理 |
| 1.产生中断信号 | 硬件 |
| 2.将相应的准备位置位 | 硬件 |
| 3.通过准备位修改相应的IP位 | 硬件 |
| 4.将IP域与IM域相与 | 硬件 |
| 5.进行中断优先级译码 | 硬件 |
| 6.将译码结果与IE位相与通过向量计算中处理程序入口地址 | 硬件 |
| 7.检测相与结果是否为0 | 硬件 |
| 8.进入中断检测流程将向量中断入口地址保存专用寄存器中 | 硬件 |

中断触发的处理如下表2.1所示

表2.1 中断触发处理

中断触发流程如下图2.1



图2.1 中断触发流程图

2.2 中断检测

中断检测的处理如下表2.2，表2.3，表2.4

表2.2 中断检测总流程

|  |  |
| --- | --- |
| 中断检测总流程 | 软件处理/硬件处理 |
| 1.判断MEM/WB寄存器中Exception abort是否被置位 | 硬件 |
| 2.清空IF/ID、ID/EXE、EXE/MEM寄存器清空所有的写控制信号 | 硬件 |
| 3.判断当前模式是否为内核模式 | 硬件 |
| 4.进入普通中断检测流程 |  |
| 5.进入嵌套中断检测流程 |  |
| 6.关闭中断使能位 | 硬件 |

表2.3普通中断检测流程

|  |  |
| --- | --- |
| 普通中断检测流程 | 软件处理/硬件处理 |
| 1.将处理器设为内核模式 | 硬件 |
| 2.判断WB阶段指令是否处于延迟槽 | 硬件 |
| 3.将EPC置为WB阶段指令的地址 | 硬件 |
| 4.将EPC设为Branch或Jump指令的地址 | 硬件 |
| 5.将pc置为向量中断入口地址进行跳转 | 硬件 |

表2.4嵌套中断检测流程

|  |  |
| --- | --- |
| 嵌套中断检测流程 | 软件处理/硬件处理 |
| 1.判断WB阶段的指令是否处于延迟槽中 | 硬件 |
| 2.将相应的影子寄存器设为Branch或Jump指令的地址 | 硬件 |
| 3.将相应的影子寄存器置为WB阶段指令的地址 | 硬件 |
| 4.将pc置为嵌套中断入口地址并进行跳转 | 硬件 |
| 5.将pc置为向量中断入口地址进行跳转 | 硬件 |

中断检测流程如下图2-2，图2-3，图2-4



图2-2 中断检测流程总图



图2-3 普通中断检测流程图



图2-4 嵌套中断检测流程图

2.3 中断处理

中断处理如下表2.5，表2.6

|  |  |
| --- | --- |
| 普通中断处理流程 | 软件处理/硬件处理 |
| 1.将EPC、ErrorEPC、Status中的EXL、ERL保存到系统栈 | 软件 |
| 2.根据当前中断类型修改IM域 | 软件 |
| 3.将本次中断请求准备位清除 | 软件 |
| 4.打开中断使能位 | 软件 |
| 5.进行相应的中断处理 | 软件 |
| 6.跳转到统一返回处理入口地址 | 软件 |

表2.5 普通中断处理

表2.6 嵌套中断处理

|  |  |
| --- | --- |
| 嵌套中断处理流程 | 软件处理/硬件处理 |
| 1.将CP0中相关寄存器保存到系统栈中 | 软件 |
| 2.将影子寄存器EPC转移到EPC | 软件 |
| 3.跳转到向量中断入口地址 | 软件 |
| 4.进行相应的中断处理 | 软件 |
| 5.跳转到统一返回处理入口地址 | 软件 |

中断处理流程如下图2-5和图2-6



图2-5 普通中断处理流程



图2-6 嵌套中断处理流程