实验 4-1 报告

学号:2016K8009929060
姓名:王晨赳

一、实验任务（10%）

本次实验需要添加 CPU 对系统调用指令 syscall 的支持，为此需增加 mtc0、mfc0、eret 指令，同时需设置 cp0_epc、cp0_status、cp0_cause 寄存器。通过 69 个点的行为仿真测试后，再上板进行检验。

二、实验设计（30%）

实验需要添加 syscall 指令，即 CPU 需支持系统调用例外。为此需要添加 mtc0、mfc0、eret 指令及 cp0_epc、cp0_status、cp0_cause 寄存器。处理器对例外的一般处理流程为，在例外提交时，如果 status_exl 为 0，且发生例外的指令不在分支延迟槽中，那么将这条指令的 PC 写到 cp0_epc，否则将这条指令前一条指令的 PC 写到 cp0_epc，同时将 cause_bd 置 1。例外提交时还会把 status_exl 置 1，之后跳转到通用例外处理入口 0xbfc00380，开始执行例外处理程序。要实现精确例外，即发生例外的指令之前的指令都执行完了，发生例外的指令之后的指令不会修改处理器的状态，处理器的状态包括寄存器和内存的值等。因为我之前的 CPU 在 MEM 阶段向寄存器堆发请求写回，所以发生的例外统一在 MEM 阶段提交。例外的标记则在发生例外的那一流水级进行，如 syscall 就在 ID 阶段标记，然后流到 MEM 阶段提交。为避免修改处理器状态，可以在可能修改处理器状态的控制信号的赋值逻辑后与上例外标记的取反。比如系统调用例外标记 exl_sys，在控制信号后与上~exl_sys 即可。eret 指令起到例外返回作用，负责置 status_exl 为 0，并跳转到 cp0_epc 里的 PC 处执行。对于 mtc0 和 mfc0 指令需要考虑数据相关问题。对于 cp0 寄存器需要考虑各个数据域可写还是只读的问题，以及何时向哪几位写入数据。

三、实验过程（60%）

（一）实验流水账

2018.11.18 20:00~22:00 构思，写代码

2018.1119 14:00~17:00; 18:00~23:00 写代码，调试

（二）错误记录

1、错误 1

（1）错误现象

Debug 的 PC 与 reference 的 PC 不符。

（2）分析定位过程

看在该时刻的寄存器堆写信号是否拉高。

（3）错误原因

发生例外的指令之后的指令修改了处理器状态。

（4）修正效果

将例外之后指令的寄存器堆写使能、内存写使能置为 0，问题解决。

（5）归纳总结（可选）

说说你觉得这个错误是哪种类型的，今后如何提前规避。

2、错误 2

（1）错误现象

Syscall 指令发生例外后，例外返回还是返回到 syscall 指令。

（2）分析定位过程

对照汇编代码，已经有指令将 cp0_epc 的值加 4 了，之后看波形里的 cp0_epc 未加 4。最后观察上下几条指令的相关性。

（3）错误原因

mtc0、mfc0 指令可能会有数据相关的问题。

（4）修正效果

对这 mtc0 复用之前对数据相关的处理，以及安排 mfc0 读 cp0 寄存器的时机，问题解决。

3、错误 3

（1）错误现象

hi,lo 寄存器值出错。

（2）分析定位过程

对照汇编代码，发现是在 syscall 指令后跟了一条 divu 指令。

（3）错误原因

发生例外的指令之后的指令修改了处理器状态。

（4）修正效果

将例外之后指令除法器、乘法器使能置为 0，问题解决。