**实验四报告**

2016K8009907007

黄熠华

一、实验任务（10%）

添加MTC0、MFC0、ERET、SYSCALL指令，硬件增添CP0\_STATUS、CP0\_CAUSE、CP0\_EPC寄存器。运行功能测试通过。

二、实验设计（30%）

首先是设计MTC0、MFC0、ERET、SYSCALL指令，以上指令都需要对CP0寄存器进行修改或读取。如果在写回级对CP0寄存器进行写入，那么就要考虑复杂的数据相关问题。CP0寄存器有32个，并且不同于通用寄存器，可以用wire的内容直接指定读写，而是需要用进行译码判断后决定读写的寄存器目标，这为数据前递更增添了工程上的复杂度。因此本次实验采用了在译码级读写CP0寄存器，将读取的CP0寄存器的值流水至后几级的方法。可以说是将CP0寄存器视作32位，来源有赖于指令种类的控制信号。如此一来，CP0寄存器的读写在同一阶段，便不存在CP0寄存器相关的数据冲突。

CP0\_STATUS、CP0\_CAUSE寄存器并不是每一位都对用户可写的，这点老师上课时强调过，我便按照老师上课时传授的风格实现了这两个寄存器。

需要注意的是，SYSCALL是精确例外，需要保证SYSCALL前的指令都执行，之后的指令都取消。因此不同于跳转指令，SYSCALL指令需要取消掉其延迟槽指令的执行。采取的办法是：在ID\_valid的值允许更新时，判断当前译码指令是否是SYSCALL指令，如果是，那么在下个时钟上升沿将ID\_valid的值赋值为0，这样SYSCALL的下一条指令便失去了作用。同时CP0\_EPC需要赋值为译码级的PC，使得返回后SYSCALL的下一条指令会被重新执行。

三、实验过程（60%）

（一）实验流水账

11.18 20:00 开始写Lab4实验

11.19 03:00 调试结束，仿真通过

（二）错误记录

1、错误1

（1）错误现象

例外发生后，PC又跳至一处insert\_error代码段。

（2）分析定位过程

发现SYSCALL后一条指令是bne指令，这正是跳转至insert\_error代码段的指令。

（3）错误原因

发现忘记取消SYSCALL后的指令。

（4）修正效果

程序在该位置运行正确。

（5）归纳总结（可选）

2、错误2

（1）错误现象

SYSCALL指令写数据与trace不符。

（2）分析定位过程

发现与trace写数据相差了4，便明白写数据用错了。

（3）错误原因

写CP0\_EPC的数据应该为译码级的PC而不是当前取址级的PC。

（4）修正效果

程序正确通过该处。

（5）归纳总结（可选）

四、实验总结（可选）

这次实验阶段相对简单，工作量合理，希望以后能经常遇到。