

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机体系结构 |
| 姓 名： | 王嘉豪 |
| 学 院： | 计算机科学与技术学院 |
| 系： | 计算机系 |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3190105304 |
| 指导教师： | 陈文智 |

2021年10月25日

**浙江大学实验报告**

课程名称： 计算机体系结构 实验类型： 综合

实验项目名称： Pipelined CPU supporting exception & interrupt

学生姓名： 王嘉豪 专业： 计算机科学与技术 学号： 3190105304

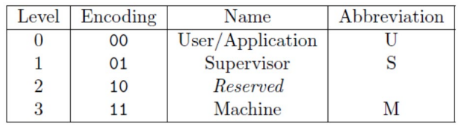
同组学生姓名： 汪辉 指导老师： 陈文智

实验地点： 曹西301 实验日期： 2021 年 10 月 20 日

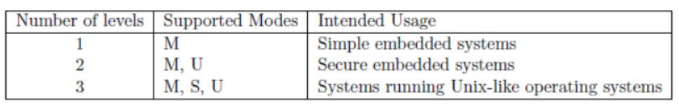
1. 实验目的和要求
2. 理解CPU中断和异常的原理以及执行过程。
3. 掌握支持中断和异常处理的流水线CPU的设计方法。
4. 掌握验证支持中断和异常处理的流水线CPU的方法。
5. 实验内容和原理
6. 实验内容
   1. 设计中断和异常处理模块。
   2. 使用程序验证设计的流水线CPU，观察程序运行。
7. 实验原理

2.1. RISC-V的三种特权模式

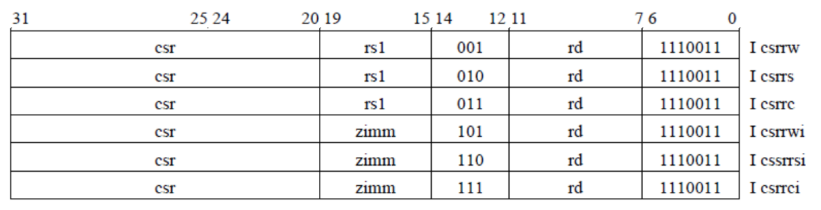
RISC-V 有三个特权模式：U (user) 模式、S (supervisor) 模式和 M (machine) 模式。



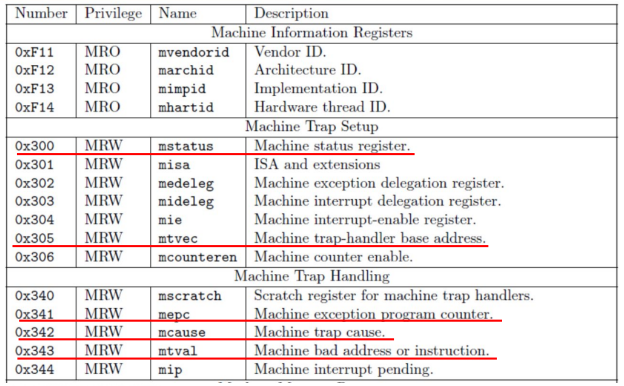
支持不同特权模式的CPU用于的系统也有差异。



2.2. 与中断和异常处理有关的CSR寄存器的读写指令



2.3. 相关CSR寄存器用途及编号



2.4. 在M模式运行期间可能发生的五种同步例外

a. 访问错误异常：当物理内存的地址不支持访问类型时发生。

b. 断点异常：在执行ebreak指令或地址或数据与调试触发器匹配时发生。

c. 环境调用异常：在执行ecall指令时发生

d. 非法指令异常：在译码阶段发现无效操作码时发生

e. 非对齐地址异常：在有效地址不能被访问大小整除时发生。 2.5. 发生异常/中断时，硬件自动经历如下的状态转换

a. 异常指令的PC被保存在mepc中，PC被设置为mtvec。mepc指向导致异常的指令；对于中断，它指向中断处理后应该恢复执行的位置。

b. 根据异常来源设置mcause，并将mtval设置为出错的地址或者其它适用于特定异常的信息字。

c. 把控制状态寄存器mstatus中的MIE位置零以禁用中断，并把先前的 MIE值保留到MPIE中。

d. 发生异常之前的权限模式保留在mstatus的MPP域中，再把权限模式改为M。（本实验只有机器模式，因此无需实现这一步）

1. 实验过程和数据记录

本实验的主要工作是编写ExceptionUnit.v，以下对相关代码进行分析。

1. 相关常量

    parameter mie\_addr = 12'h304 ;

    parameter mip\_addr = 12'h344 ;

    parameter mepc\_addr = 12'h341 ;

    parameter mtvec\_addr = 12'h305 ;

    parameter mcause\_addr = 12'h342 ;

    parameter mstatus\_addr = 12'h300 ;

这些变量是在异常阶段操作的几个CSR寄存器的地址。

2. 相关控制信号

    assign reg\_FD\_flush = flush\_FD\_reg | exception | mret ;

    assign reg\_DE\_flush = flush\_DE\_reg | exception | mret ;

    assign reg\_EM\_flush = flush\_EM\_reg | exception | mret ;

    assign reg\_MW\_flush = flush\_MW\_reg | exception | mret ;

    assign RegWrite\_cancel = exception ;

相关各阶段的寄存器在异常时要清空。

3. 核心代码  
 上升沿进行以下操作：刷新、重定向和更新写信号，如有需要即跳转。

always @(posedge clk) begin

    if ( flush\_reg )

            begin

                        flush\_FD\_reg <= 0 ;

                        flush\_DE\_reg <= 0 ;

                        flush\_EM\_reg <= 0 ;

                        flush\_MW\_reg <= 0 ;

                        flush\_reg <= 0 ;

            end

    if ( exception )

            begin

                flush\_reg <= 1 ;

                flush\_FD\_reg <= 1 ;

                flush\_DE\_reg <= 1 ;

                flush\_EM\_reg <= 1 ;

                flush\_MW\_reg <= 1 ;

            end

    redirect\_reg <= exception ; // redirecting after one cycle when exception

    write\_signal <= next\_signal ; // status of signal for writing CSRs during exception

    turn\_on <= next\_turn ; // status of turn

end

如果发生了异常，则需要多花费一个周期进入trap。在异常发生后的一个周期进行重定向，并且进行写信号、跳转信号的更改。

在下降沿，若发生异常，向CSR寄存器进行写操作，并进行跳转。

always @\* begin

    if ( ~clk ) begin

            if ( turn\_on ) begin

                case (write\_signal )

                    2'b00:begin //write reg mepc

                        csr\_waddr <= mepc\_addr ;

                        csr\_wdata <= epc\_reg ;

                    end

                    2'b01:begin //write reg mstatus

                        csr\_waddr <= mstatus\_addr ;

                        csr\_wdata <= mstatus\_reg ;

                    end

                    2'b10:begin //write reg mcause

                        csr\_waddr <= mcause\_addr ;

                        csr\_wdata <= exception\_reg ;

                    end

                    2'b11:begin

                        next\_turn <=  0;

                    end

                endcase

                    next\_signal <= write\_signal+1 ;

            end

    end

            csr\_wsc = csr\_rw\_in ? csr\_wsc\_mode\_in : 2'b00 ;

            csr\_w = csr\_rw\_in | turn\_on | mret;

            if ( csr\_rw\_in & ~turn\_on )

                begin

                    csr\_waddr = csr\_rw\_addr\_in ;

                    csr\_raddr = csr\_rw\_addr\_in ;

                    if (~csr\_w\_imm\_mux)   csr\_wdata = csr\_w\_data\_reg ;

                        else    csr\_wdata = {27'b0,csr\_w\_data\_imm} ;

                end

            else if ( exception & ~turn\_on )

                begin

                    next\_turn = 1 ;

                    csr\_raddr = mtvec\_addr ;

                    epc\_reg = epc\_cur ;

                    exception\_reg = exception\_code ;

                    mstatus\_reg = {mstatus[31:8],mstatus[3],mstatus[6:4],1'b0,mstatus[2:0]} ;

                end

            else if ( mret & ~turn\_on )

                begin

                    csr\_raddr = mepc\_addr ;

                    csr\_waddr = mstatus\_addr ;

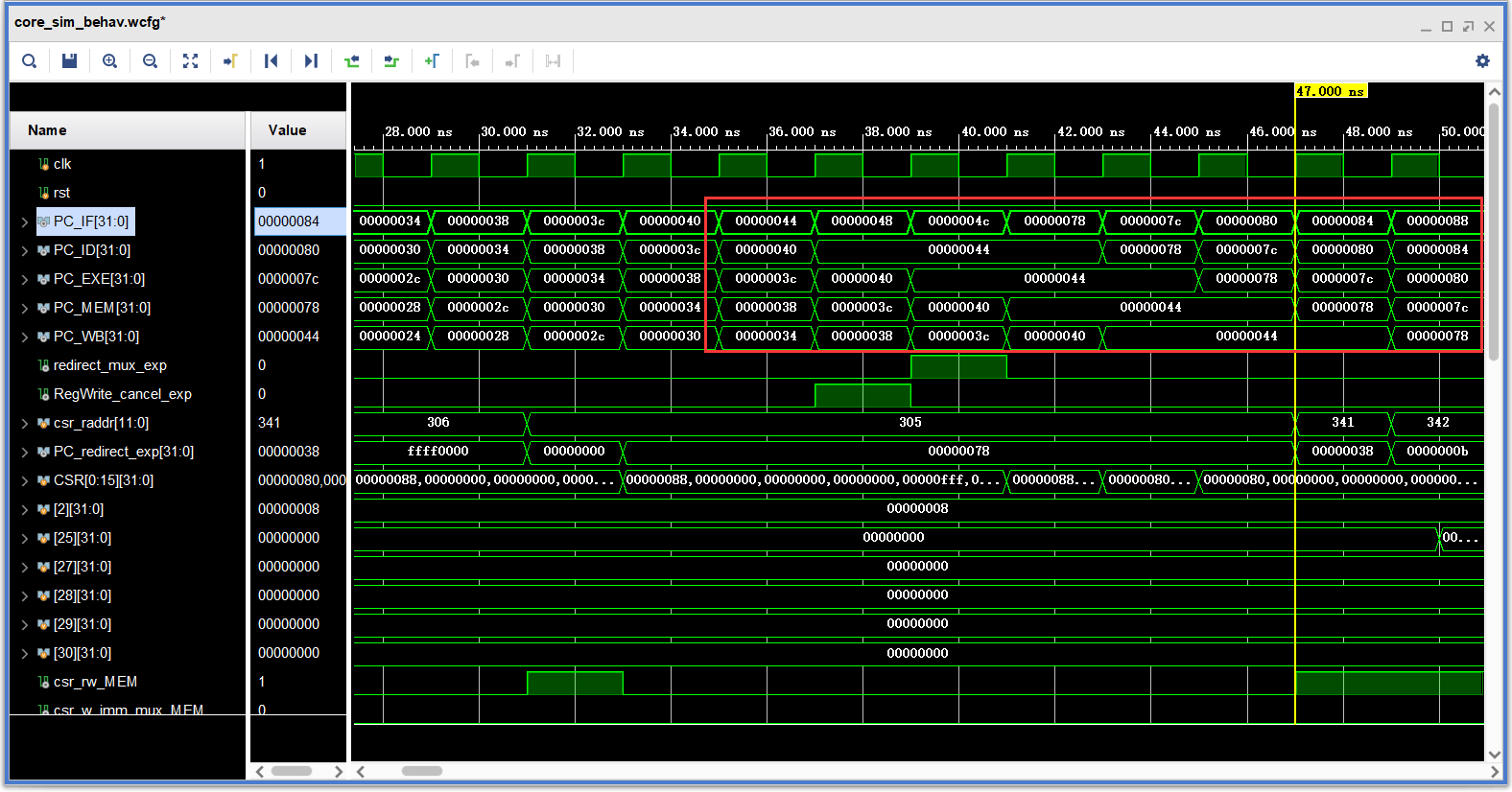
                    csr\_wdata = {mstatus[31:4],mstatus[7],mstatus[2:0]};

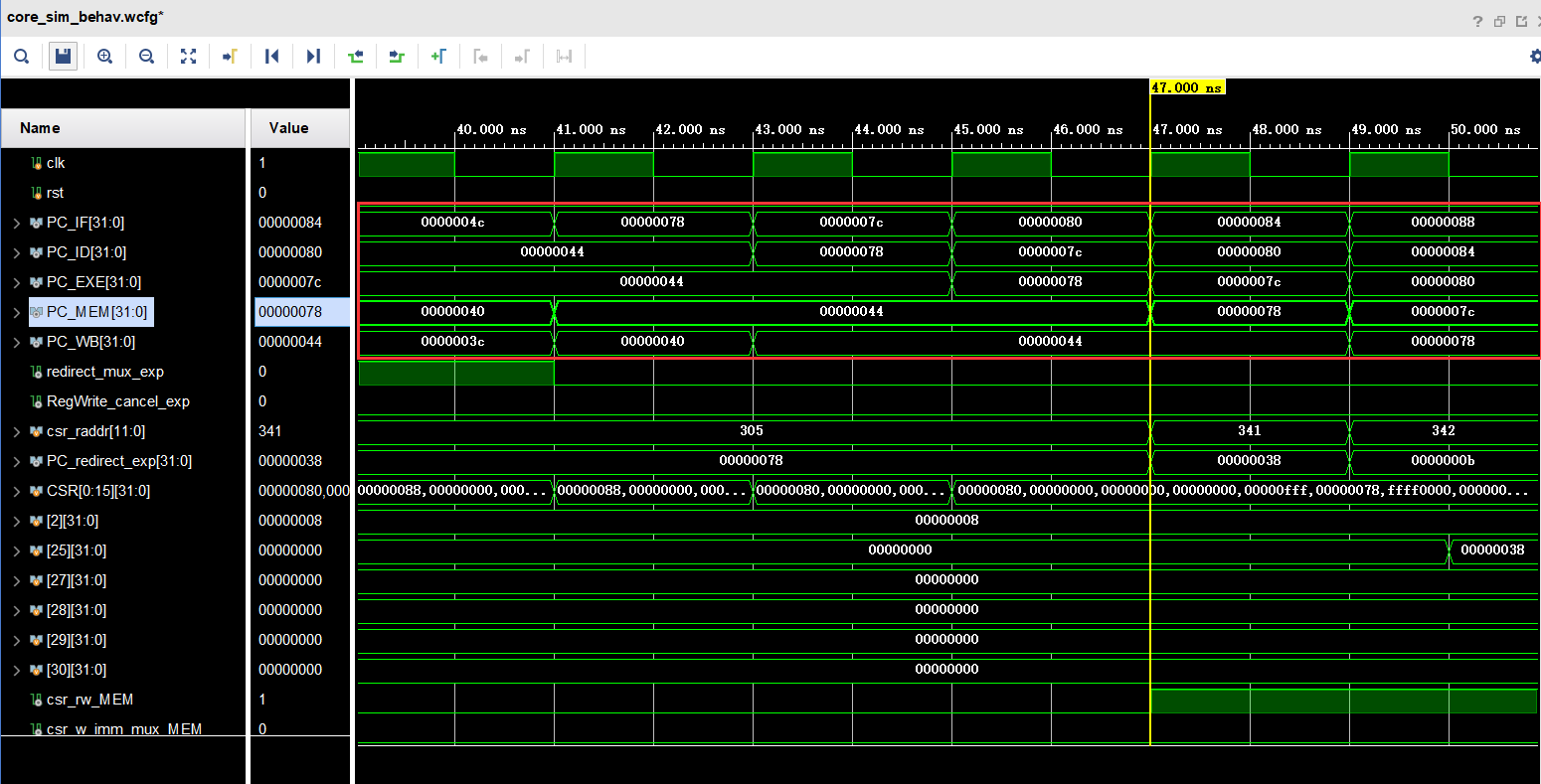
                end

end

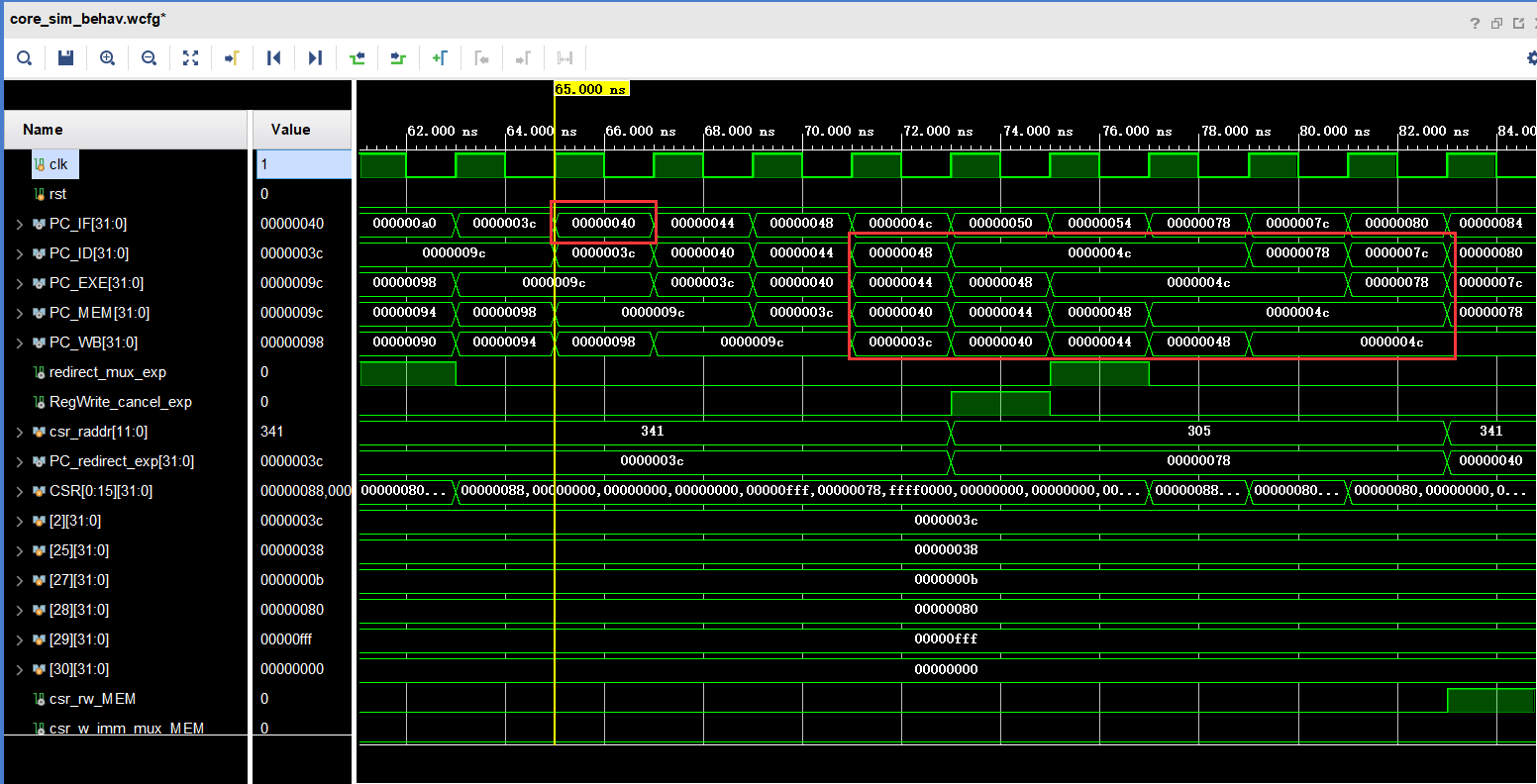
1. 实验结果分析
2. 仿真波形

ecall指令地址为38，自IF后五个周期跳转至78执行异常处理。

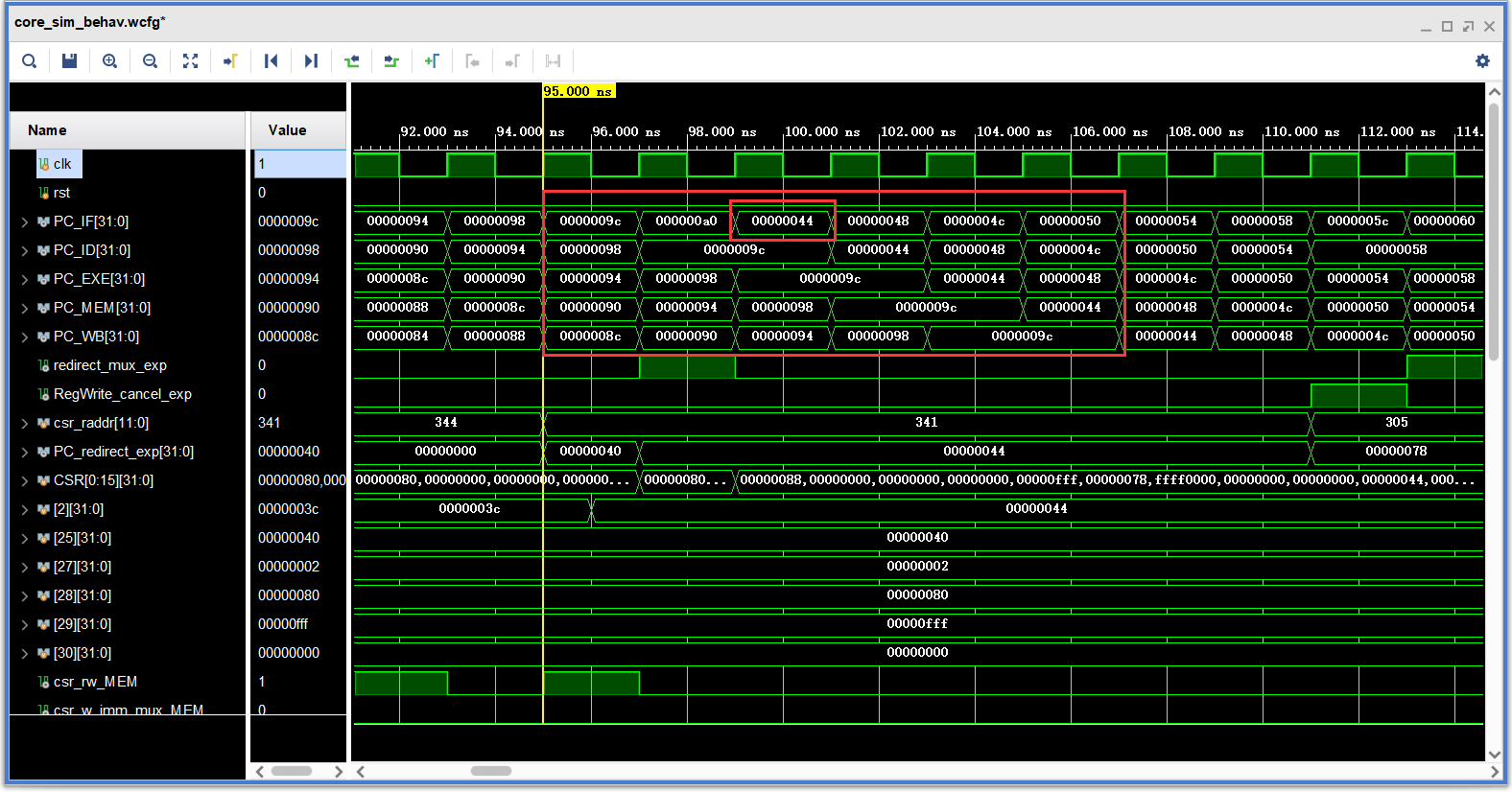




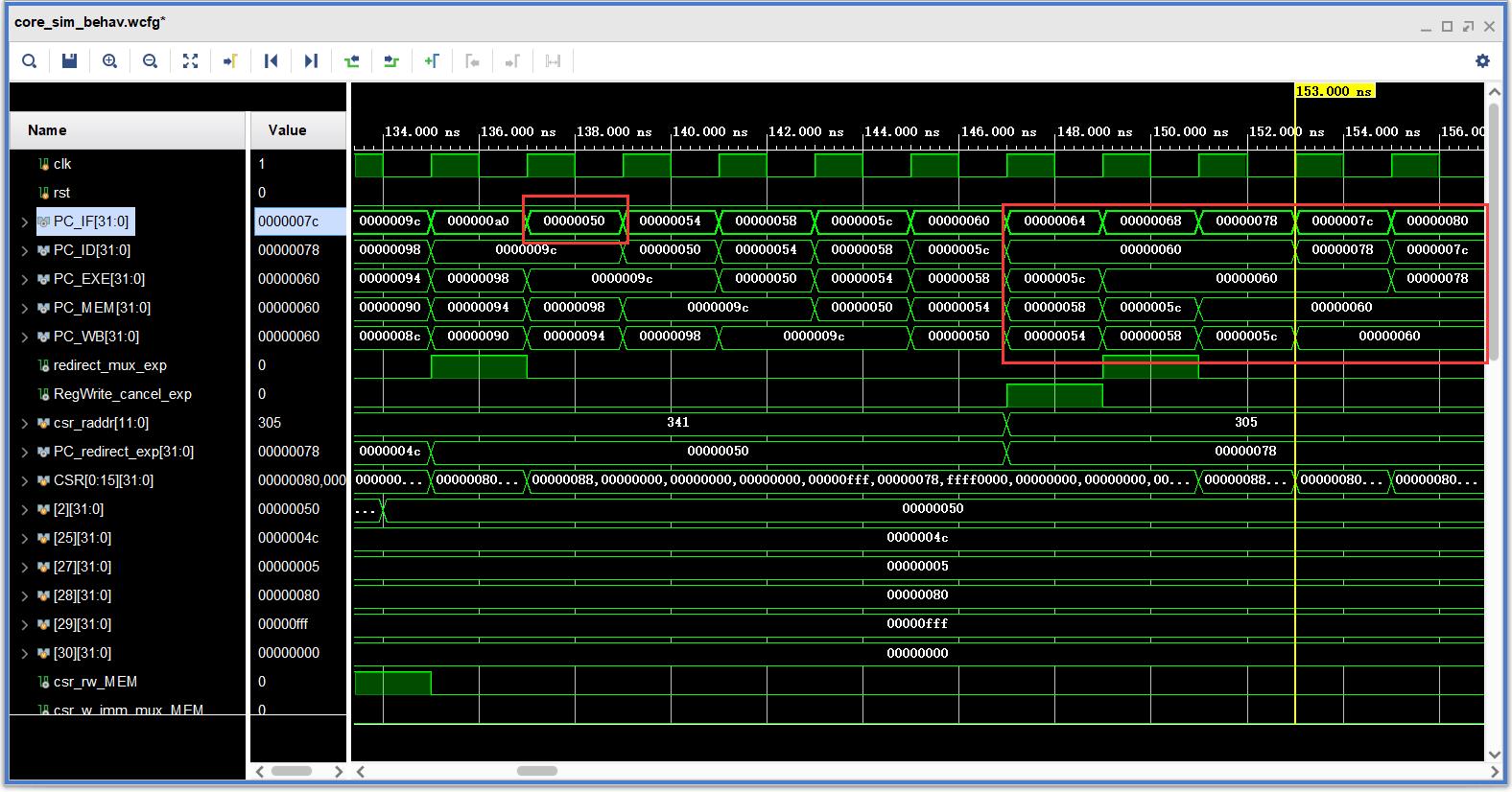
非法指令地址为40，而后进行异常处理。



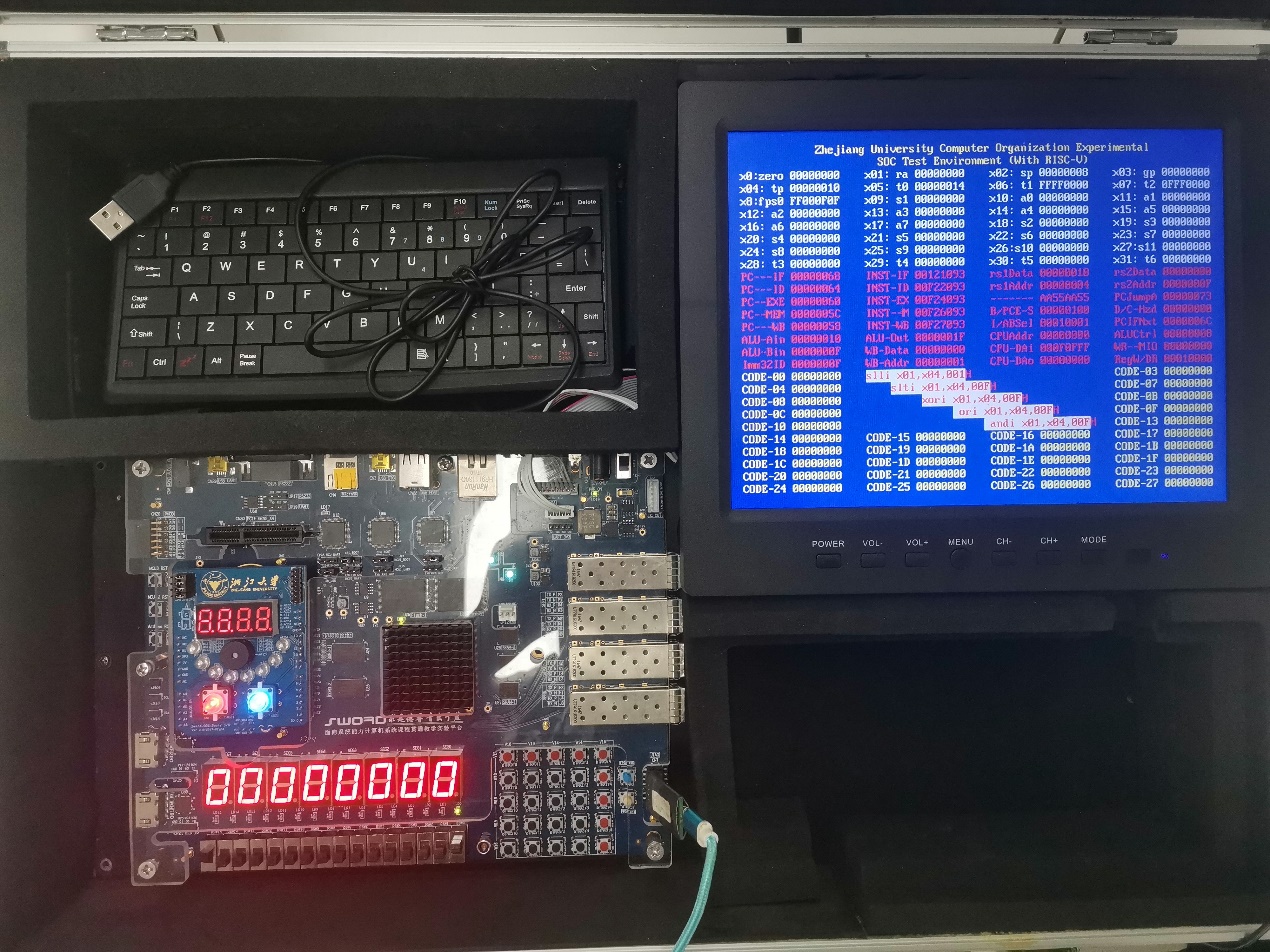
lw指令访问地址4c失败，进行异常处理。



sw指令访问地址54失败，进行异常处理。



1. 实验箱记录



1. 讨论与心得

本实验是继续完善流水线CPU的功能，要求增加中断异常处理功能。因为我计算机组成课基础不够好，即使助教老师给出的工程代码非常详细，但还是感觉比较困难，而且在实验过程中，我和我的队友碰到了几个很少见的问题，向助教老师求助之后再进行了长时间的修改和调整才得到了正确的结果，总的来说还是要先对相关知识有完整的了解，再进行代码的阅读和理解，最后进行代码的编写以及debug。

实验中，我和汪辉使用了三个周期进行异常的处理，使用上升沿和下降沿进行判断等，这样的结果在仿真中是正确的，但上板后遇到了一些问题，还是多亏了助教老师提醒，不然我们还难以发现原因，对于比较复杂甚至有点“玄学”的硬件实验，还是要与同学多进行交流，积极与助教老师沟通。

这次实验总体来说不算轻松，希望之后的实验能够做的更加得心应手！感谢我的队友汪辉，感谢老师们的悉心教导。