

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机组成 |
| 姓 名： | 应周骏 |
| 学 院： | 计算机科学与技术学院 |
| 系： | 计算机科学与技术系 |
| 专 业： | 计算机科学与技术 |
| 学 号： | 3200103894 |
| 指导教师： | 马德 |

2022 年 5月 21 日

**浙江大学实验报告**

课程名称： 计算机组成 实验类型： 综合

实验项目名称： CPU设计之中断

学生姓名： 应周骏 专业：计算机科学与技术 学号：3200103894

同组学生姓名： 无 指导老师： 马德

实验地点： 东4-509实验日期：2022 年 4 月 25 日

**一、实验目的和要求**

1. 深入理解CPU结构

3. 学习如何提高CPU使用效率

3. 学习CPU中断工作原理

4. 设计中断测试程序

**二、实验内容和原理**

**目标：**

熟悉RISC-V 中断的原理，了解引起CPU中断产生的原因及其处理方法，扩展包含中断的CPU。

**内容：**

扩展实验CPU中断功能并设计测试。

**原理：**

1. 中断概念

中断是指程序执行过程中，当发生某个事件时，中止CPU上现行程序的运行，引出处理该事件的程序执行的过程，此过程都需要打断处理器正常的工作，为此，才提出了“中断”的概念。

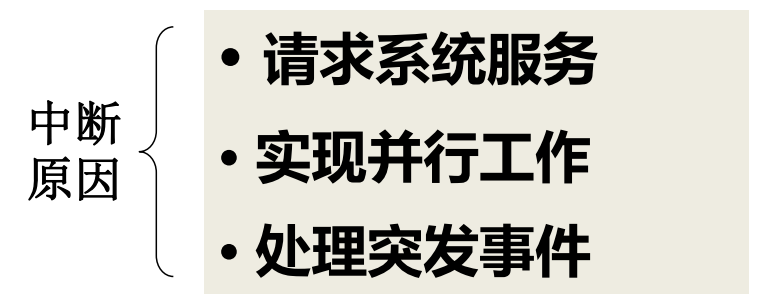


图2.1 中断原因

在中断过程中，有如下几个概念：

·中断源：引起中断的事件称为中断源；

·中断请求：中断源向CPU提出处理的请求；

·断点：发生中断时被打断程序的暂停点；

·中断响应：CPU暂停现行程序而转为响应中断请求的过程；

·中断处理程序：处理中断源的程序；

·中断处理：CPU执行有关的中断处理程序；

·中断返回：返回断点的过程；

2. 中断的分类与作用

按照中断信号的来源，可把中断分为外中断和内中断两类：

·外中断（又称中断）：指来自处理器和主存之外的中断；

·内中断（又称异常）：指来自处理器和主存内部的中断；

中断处理程序：保护CPU现场、处理发生的中断事件、恢复正常操作。

3. 中断（异常）处理过程

当CPU收到中断或者异常的信号时，它会暂停执行当前的程序或任务，通过一定的机制跳转到负责处理这个信号的相关处理程序中，在完成对这个信号的处理后再跳回到刚才被打断的程序或任务中。

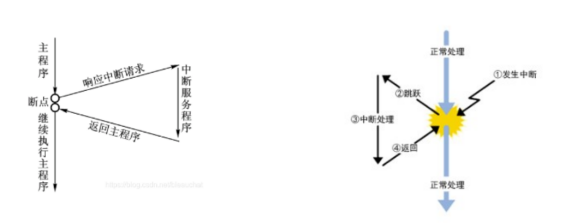


图2.2 中断处理图解

4. 简化中断设计

①外部中断（Int）触发中断或非法指令（illegial）触发异常或ecall系统调用；

②响应mtvec寄存器定义的PC值分别针对Int为0x0c；ecall为0x08；illegical为0x04；

 ③mepc寄存器值更新为下一条指令的PC值；

④执行异常服务程序；

⑤执行mret指令，返回mepc保存的PC处继续程序流；

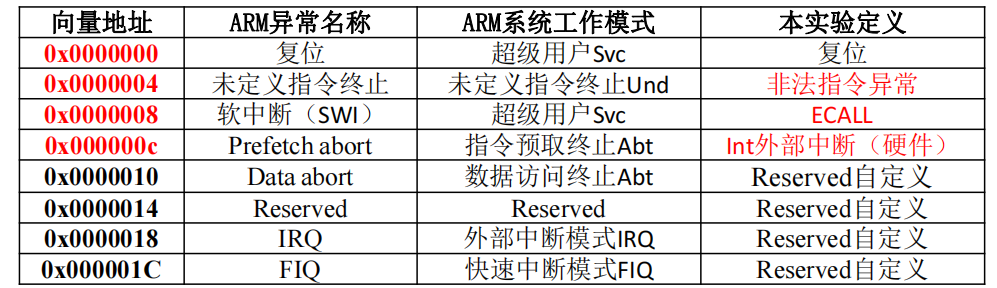


图2.3 ARM中断向量表

**三、实验过程和数据记录**

**1. 工程文件建立**

新建工程文件，命名为“OxExp04\_Interrupt”，“OExp02-IP2SOC\_inter”。

**2. SCPU模块更新**

在实验四扩展指令的“SCPU. v”文件基础上，修改部分代码。

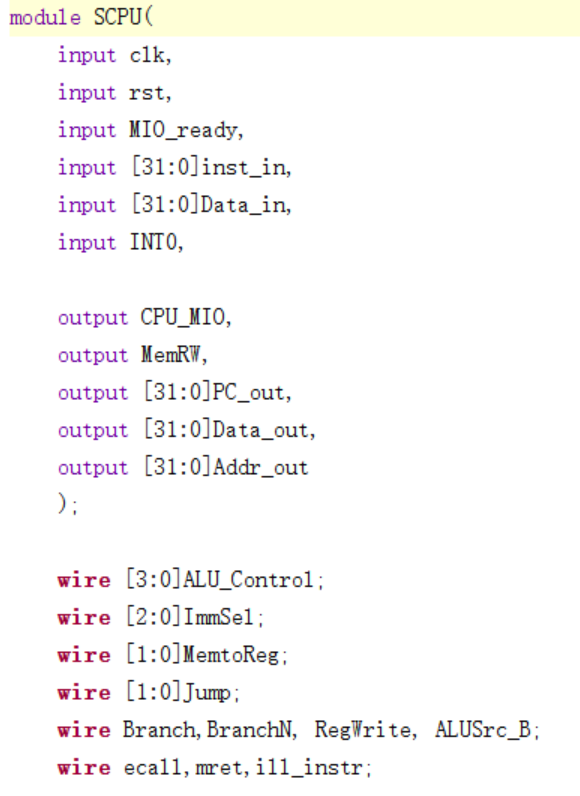


图3.2.1 SCPU接口

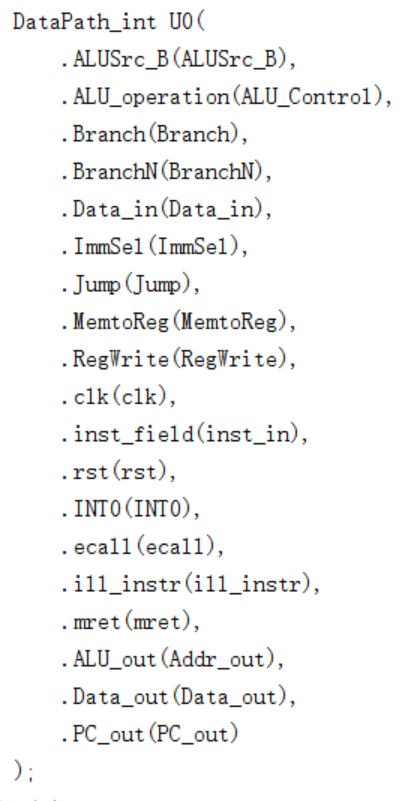
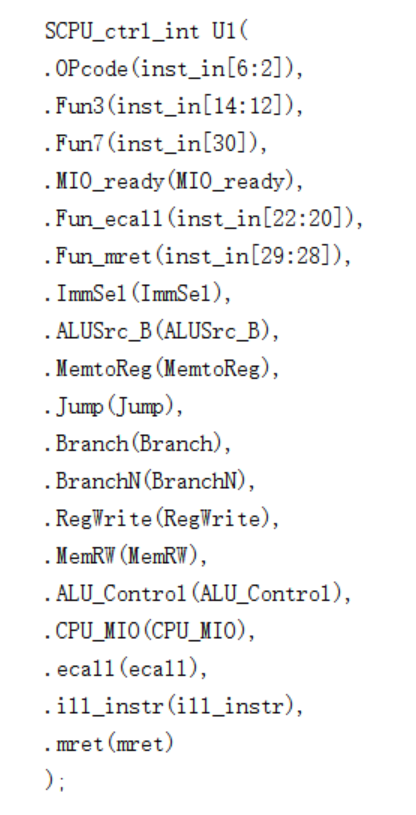


图 3.2.2 连接数据通路和控制通路

**3. 数据通路模块更新**

在实验4的“Datapath.v”基础上，添加中断信号及中断模块。

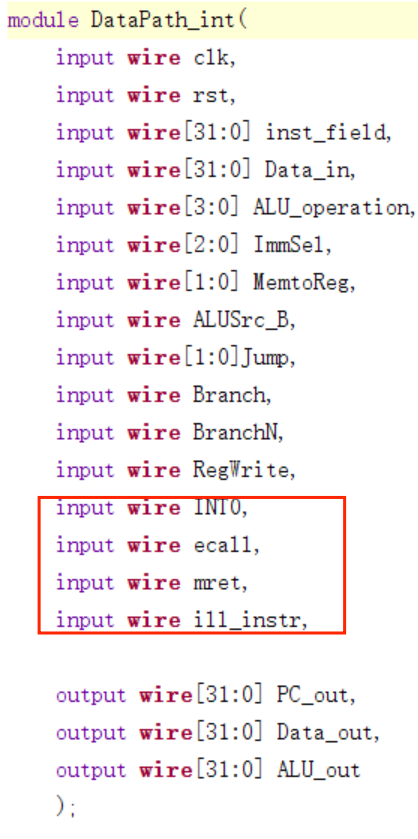


图3.3.1 数据通路接口

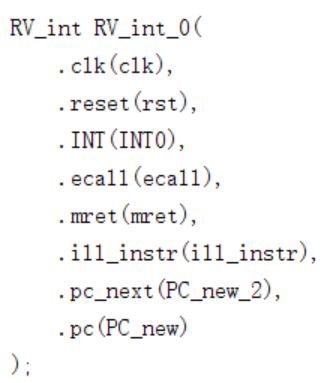


图3.3.2 添加中断控制模块

**4. RV\_int模块设计**

新建“RV\_int.v”模块，输入对应代码，以实现对中断的PC控制。

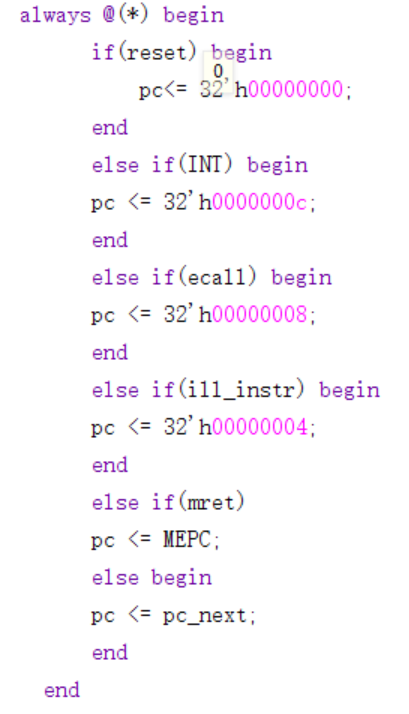
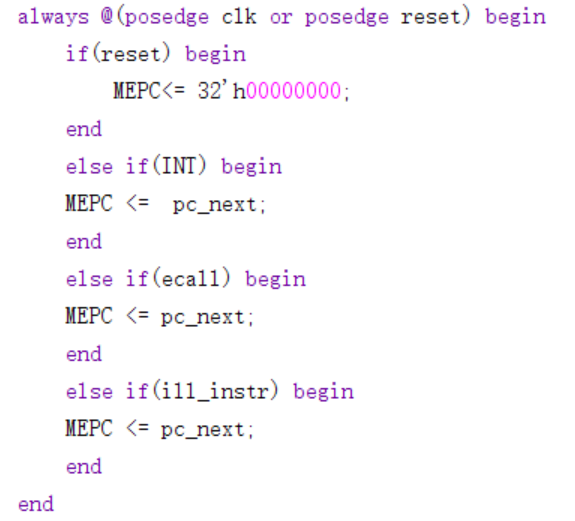
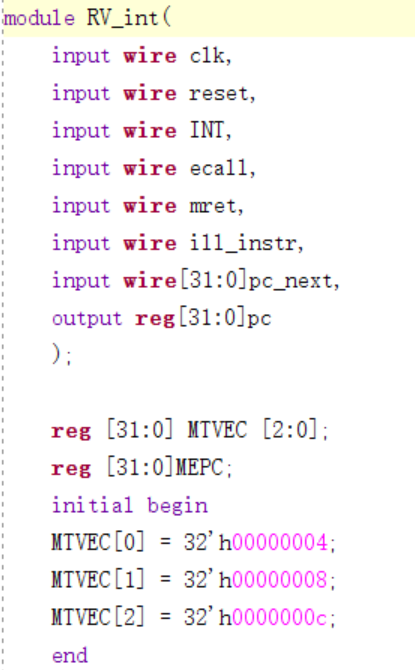


图3.4.1 中断模块代码

对该文件设计对应仿真代码，代码如下。

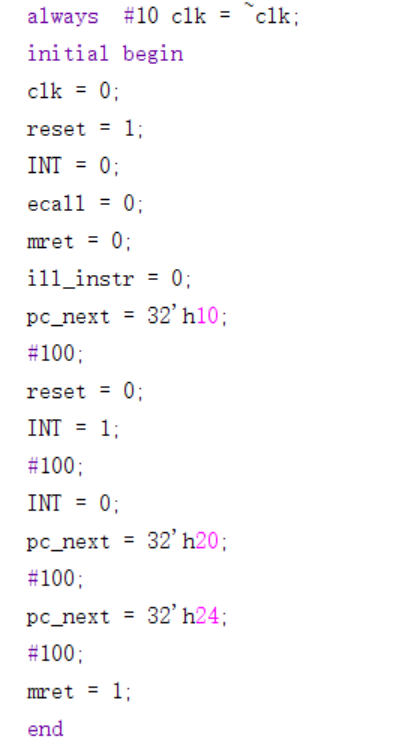


图3.4.2 中断模块仿真代码

仿真结果如下。

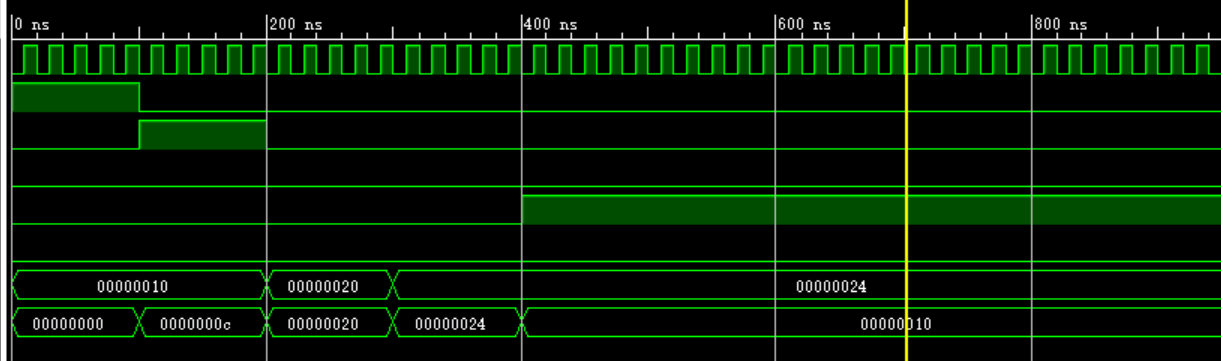


图3.4.3 中断模块仿真结果

**5. 控制通路模块设计**

在实验4的“SCPU\_ctrl.v”基础上，添加中断信号。

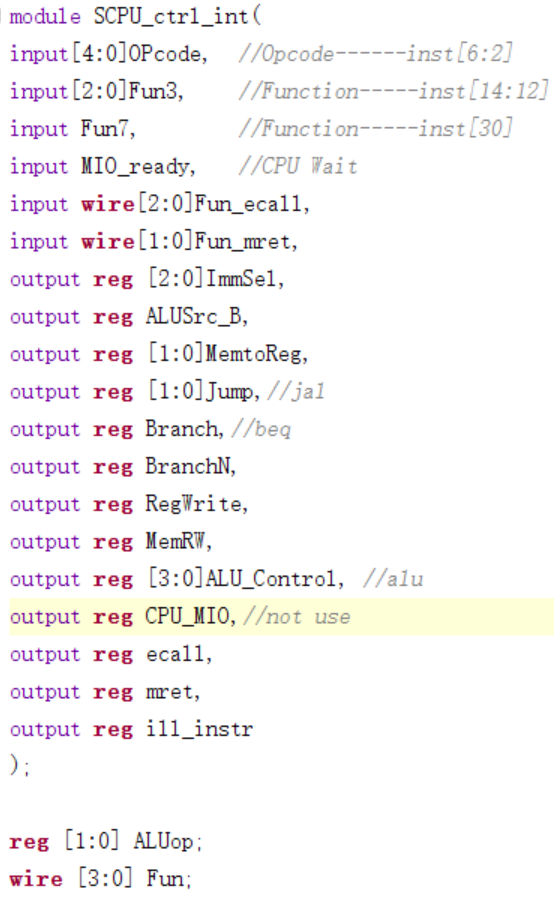


图3.5.1 控制通路接口

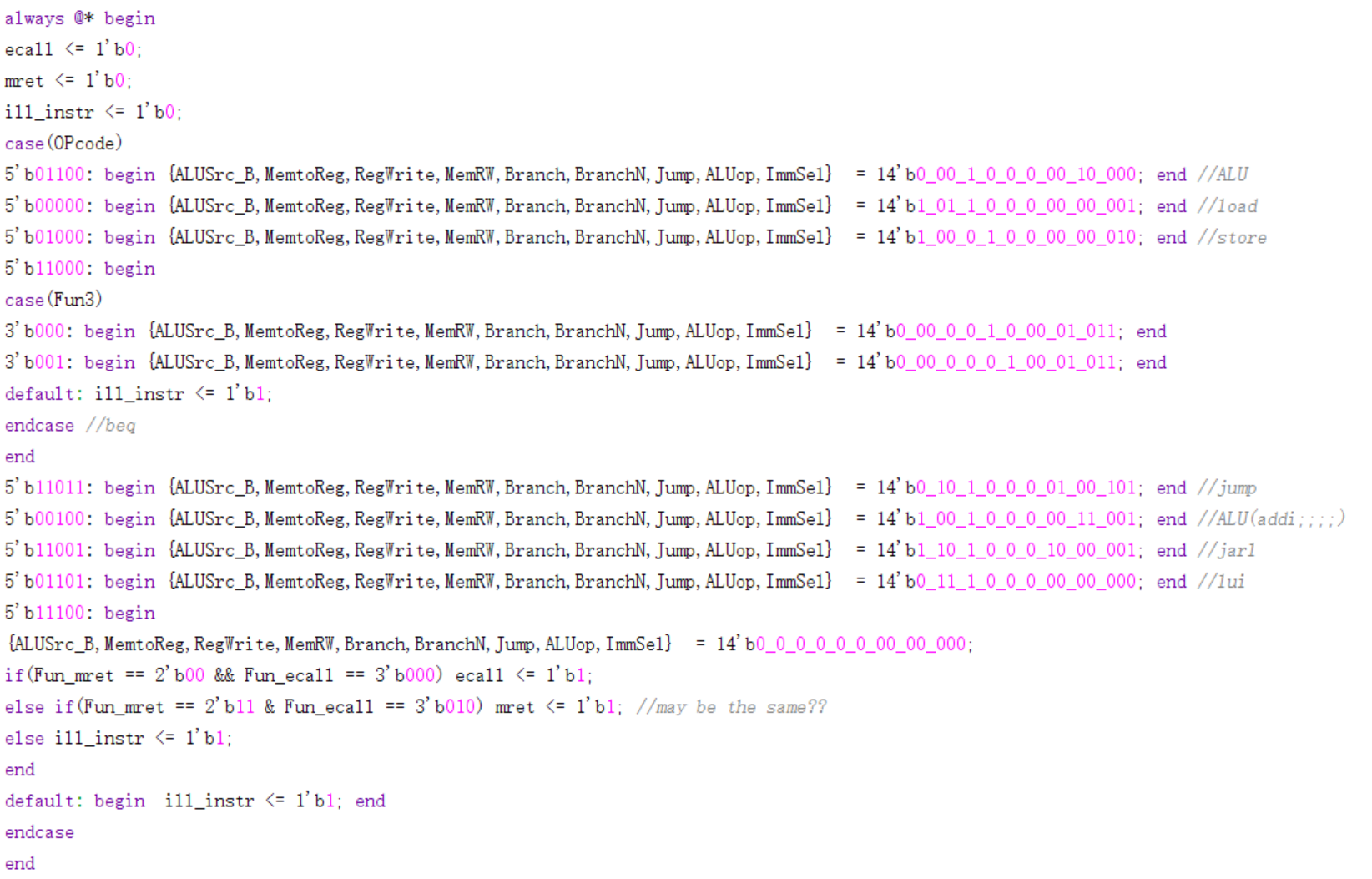


图3.5.2 控制通路信号生成

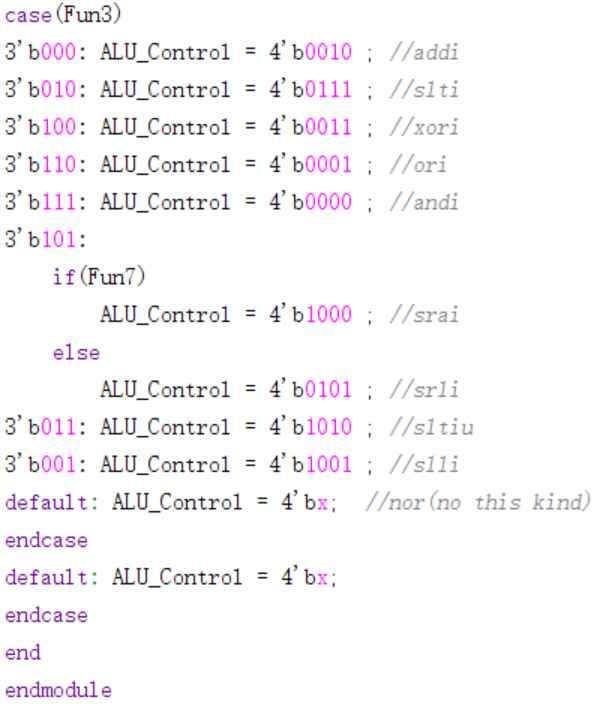
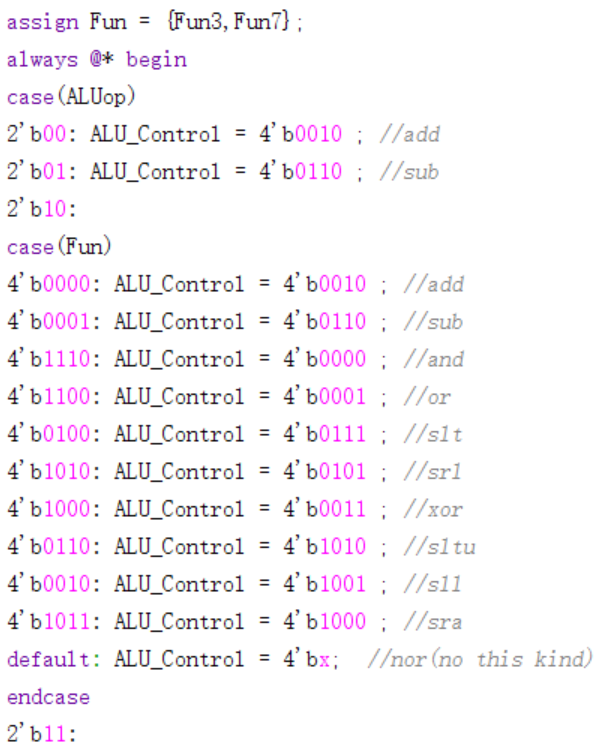


图3.5.3 控制通路ALU两级译码

**6. 测试模块接口调整**

在实验二测试框架基础上，在接口处加入按钮控制的外部中断。

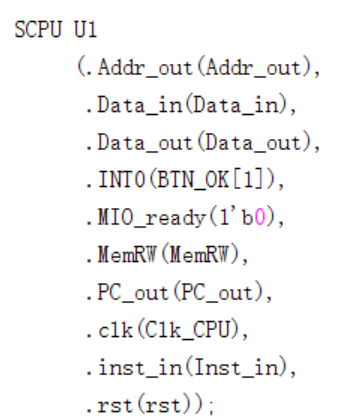


图3.6.1 外部中断接口

**7. 物理验证**

对上述扩展指令生成bit文件，在SWORD实验板上进行验证。

**四、实验结果分析**

**1. RV\_int功能分析**

该模块主要响应各类中断信号的PC变化，通过时序逻辑控制MEPC寄存器，并通过组合逻辑实现响应中断对PC的影响。

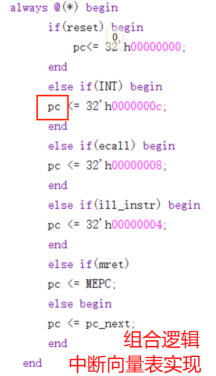
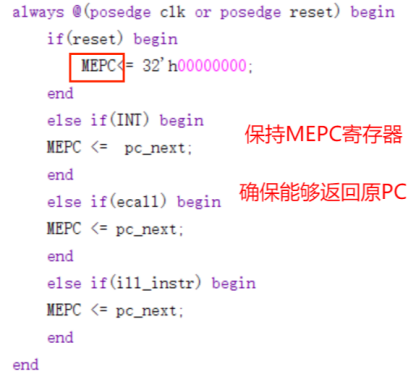
****

图4.1 RV\_int功能

**2. 中断信号生成**

控制通路中，增加对中断信号的响应，包括无效指令和ecall/外部中断。



图4.2 控制通路产生中断信号

**3. 物理验证结果**

测试程序为三种中断情况，运行能够进入中断。

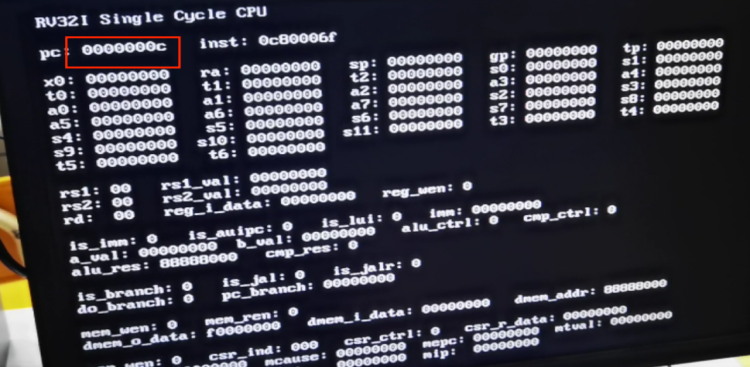


图4.3.1 按钮导致的外部中断

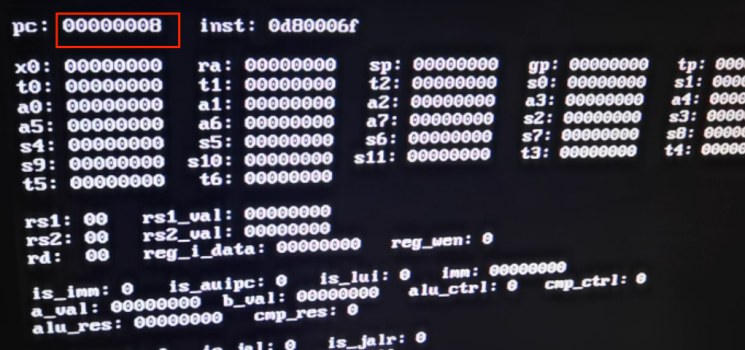


图4.3.2 ecall型中断

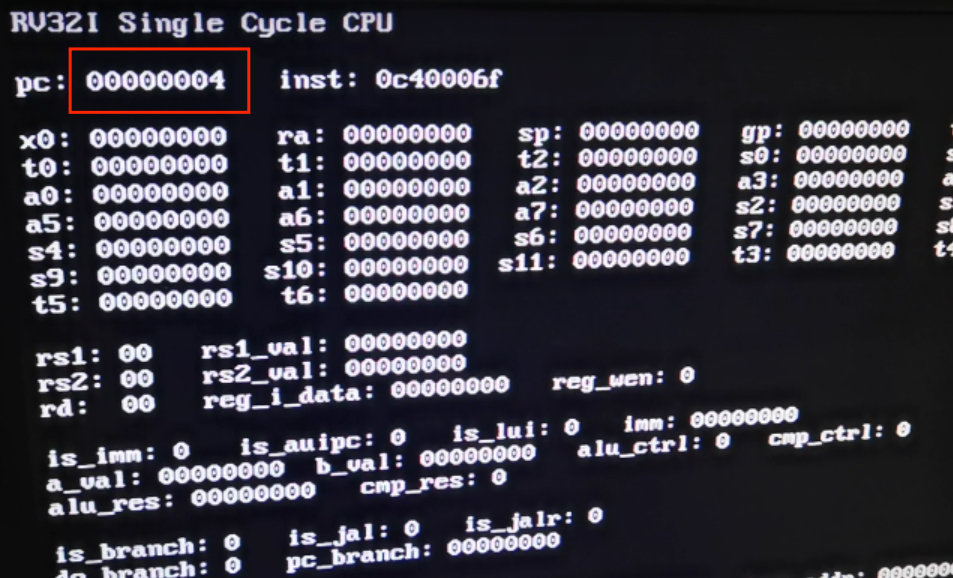


图4.3.3 非法指令中断

**五、讨论与心得**

1. 通过本次实验，我掌握了中断的概念，基本了解了CPU执行中断的流程。

2. 本次实验中，在中断信号控制过程中，我起初没有厘清中断信号的格式，导致在指令译码时，产生了对非法指令等处理有误的情况，后来通过进一步校对指令格式以及各指令概念，最终实现了中断信号的准确生成。

3. 在设计RV\_int的过程中，最初由于时序逻辑与组合逻辑的混淆，导致在实际测试时，出现了中断后PC停止、PC时钟加快等一系列问题，这也提示我此后实验中，要注意Verilog语言设计时的逻辑选择。