实验 2-1 报告

学号：2016K8009929011
姓名：段江飞

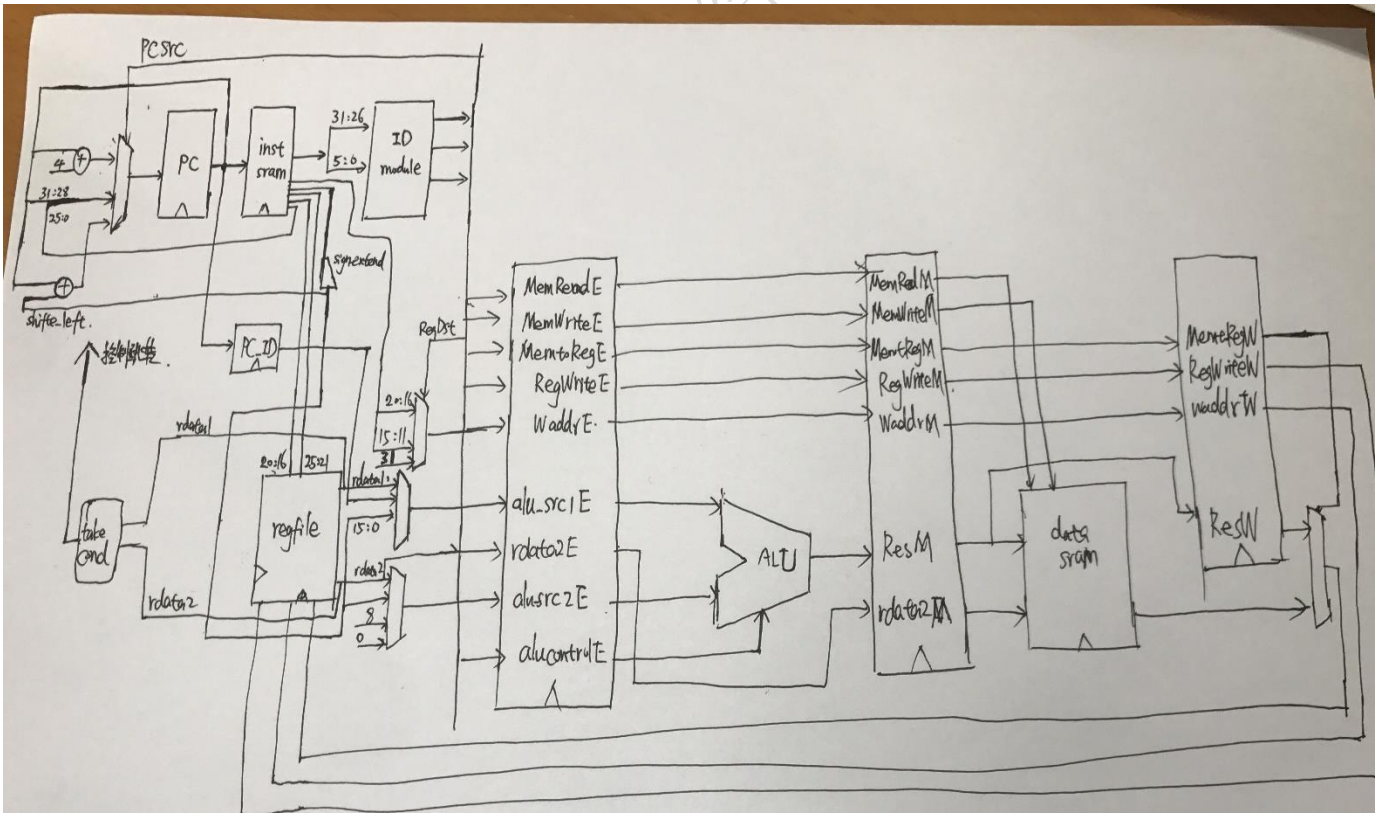
一、实验任务（10%）

本实验要设计并完成一个静态五级流水 CPU，五个阶段分别是：取指--译码--执行--访存--写回， CPU 除了实现 19 条指令外，也要实现 mips 分支延迟槽，即分支延迟槽里的指令（不管什么指令）也要执行，CPU 的设计要考虑结构相关和控制相关，避免流水线堵塞。

设计的 CPU 要通过 23 个功能点（19 条指令+4 个分支延迟槽测试）的测试，仿真验证在控制台打印 23 个功能点 PASS 和最终的测试结束的 PASS，同时在上板验证时，数码管的值要从 0 累加到 23，高低 8 位要一直相等，LED 灯的变化要和讲义中一致。

二、实验设计（30%）

如下静态五级流水 CPU 结构图



图一 CPU 结构图

（一）取指

1、取指令

连接到 inst_sram 的信号是地址和使能信号，地址信号就是 PC，使能信号则是第一级的 validin，表示第一级输入有效，也就是 PC 有效，validin 和 resetn 信号连接。

2、计算下一条指令 PC

流水阻塞时，PC 不发生变化。流水阻塞时，下一条指令的 PC 如下计算：

对非跳转指令，下一条指令地址就是 PC+4，在下一个时钟上升沿写入 PC。

对跳转指令，下一条指令地址仍然是 PC+4，在下一个时钟上升沿写入 PC，但是再下一条指令地址，根据读出指令和读出寄存器堆的数据判断跳转是否发生，若发生，则计算出跳转目标地址写入 PC，否则，继续写入 PC+4。

3、传递到下一流水信号

流水线不阻塞时，在时钟上升沿到来时，将当前 PC 传到下一级，写入 PC_ID。

（二）译码

1、译码模块

由于 inst_sram 中读出数据会延迟一拍，而且流水阻塞时，PC 不会发生变化，所以不需要指令寄存器。

译码模块输入就是读出的指令，输出是内存、寄存器堆、ALU 等的控制信号。

2、寄存器堆模块

此模块接受两个读数据地址，返回两个读数据，接受一个写数据地址、一个写使能信号和一个写入数据。

3、其他功能

流水线阻塞时，继续进行原来的操作，流水线不阻塞时，进行传递信号。

译码阶段将根据译码产生的控制信号选择将要带到下一级的 ALU 操作数和寄存器写地址。

直接译码产生的在当前阶段用不到的信号将带到下一流水级。

当前指令的 PC_ID 信号继续带到下一级。

（三）执行

1、ALU 模块

ALU 模块的输入为两个操作数和 ALU 控制信号，输出为计算结果和一些其他的信息。

2、其他功能

流水线不阻塞时，将该级没有用到的信号传递到下一级，包括访存信号，写回寄存器堆的信号等。

（四）访存

1、访存

将访存的使能信号、写使能信号和 MemReadM、MemWriteM 连接，访存地址为上一级 ALU 计算结果 ResM，写入数据为一级一级带下来的 rdata2M。

2、其他功能

流水线阻塞时，不往下传递，流水线不阻塞时，将和寄存器堆相关信号继续向下传递。

（五）写回

1、写回信号

将写回的地址和写使能信号和寄存器堆的接口连接，写回数据根据该级的 MemtoRegW 信号在 ResW 和 data_sram_rdata 之间进行选择，因为 data_sram 读出数据要慢一拍，所以不再进行存储，之间选择即可。

（六）实现功能

静态五级流水能正常运行，通过 19 个指令功能点测试。

实现分支延迟槽，通过 4 个分支延迟槽的功能点测试。

（七）验证

1、仿真验证

在 vivado 上仿真，首先编译提供的测试代码，然后给龙芯 CPU 重新定制 instram，跑测试文件生成比对文件。之后，在自己的 CPU 上进行仿真验证，知道打印 23 个功能点 PASS，通过验证。

2、上板验证

将实验箱连接到 vivado，编程后看实验箱的数码管和 LED 灯的变化，数码管的高低 8 位一直同时累加到 0x17，LED 灯的颜色变化和讲义一致，则通过上板验证。

三、实验过程（60%）

（一）实验流水账

1、9 月 20 日晚上 6 点到 11 点

理解阅读五级流水资料，并画出五级流水的结构和数据通路，同时思考如何实现分支延迟槽。

2、9 月 21 日晚上 7 点到 9 月 22 日早上 1 点

根据数据通路写 CPU，设计分支延迟槽，并跑了龙芯的 CPU。最后，尝试去创建工程但是失败了。

3、9 月 22 日晚上 6 点到 2 点

测试 CPU，发现有数据相关，并添加了数据旁路，修正了很多弱智的 bug，还有一些没想到的 bug，并通过了仿真验证和上板验证。

4、9 月 24 号晚上 7 点到 9 点

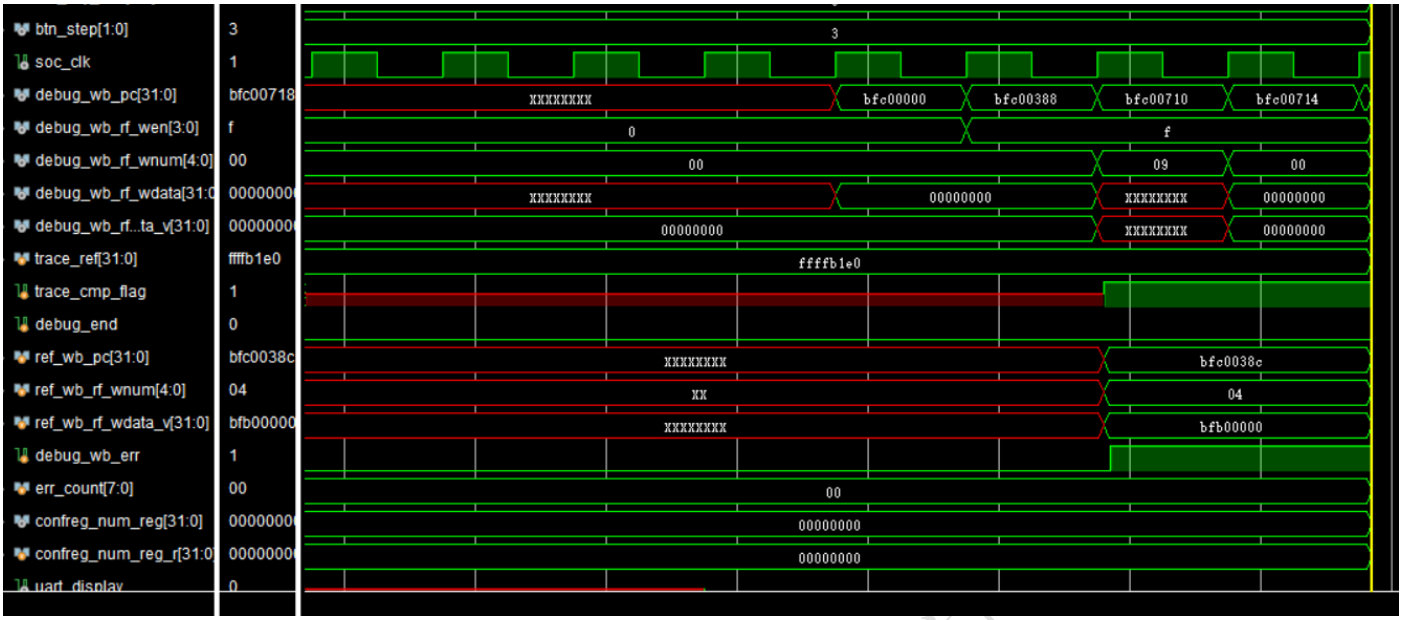
完善实验报告。

（二）错误记录

1、错误 1

（1）错误现象

指令流水混乱，PC 乱跳转，且没有分支延迟槽。



图一

（2）分析定位过程

首先根据比对信息，和波形图，猜测是因为分支延迟槽的问题，分支延迟槽莫名其妙的消失，在结构的设计中应该不会发生这个问题，但是分支延迟槽确实消失了，我在波形图中看获得的下一个 PC 和跳转信号，发现跳转信号在 resetn 刚复位，就已经满足了跳转的条件，就直接写入了跳转的目标地址，而没有经历分支延迟槽。再继续追踪，为什么跳转信号会在刚开始就满足，发现 soc 的时钟分频什么的，在 resetn 为低电平的时候就有上升沿，导致取指发生，然后在 resetn 为低电平的过程中，取指和译码就完成了。

（3）错误原因

流水级的控制信号出错，第一级的当前流水级信号有效不应该一直为 1，应该是和 resetn 同时变化，在 resetn 低电平时为。

（4）修正效果

分支延迟槽出现，并且流水正常运行下去，然后停在了下一个错误的地方。

2、错误 2

（1）错误现象

从寄存器堆读数的时候，读出的数是未知值。

（2）分析定位过程

根据反汇编定位错误指令，根据波形图和指令发现先后的指令出现数据相关，读出来的数据时未知值，但是之前应该写入的为 0，出现数据相关。

（3）错误原因

数据相关，从寄存器堆读数的时候，读数的地址是即将要写入的地址。

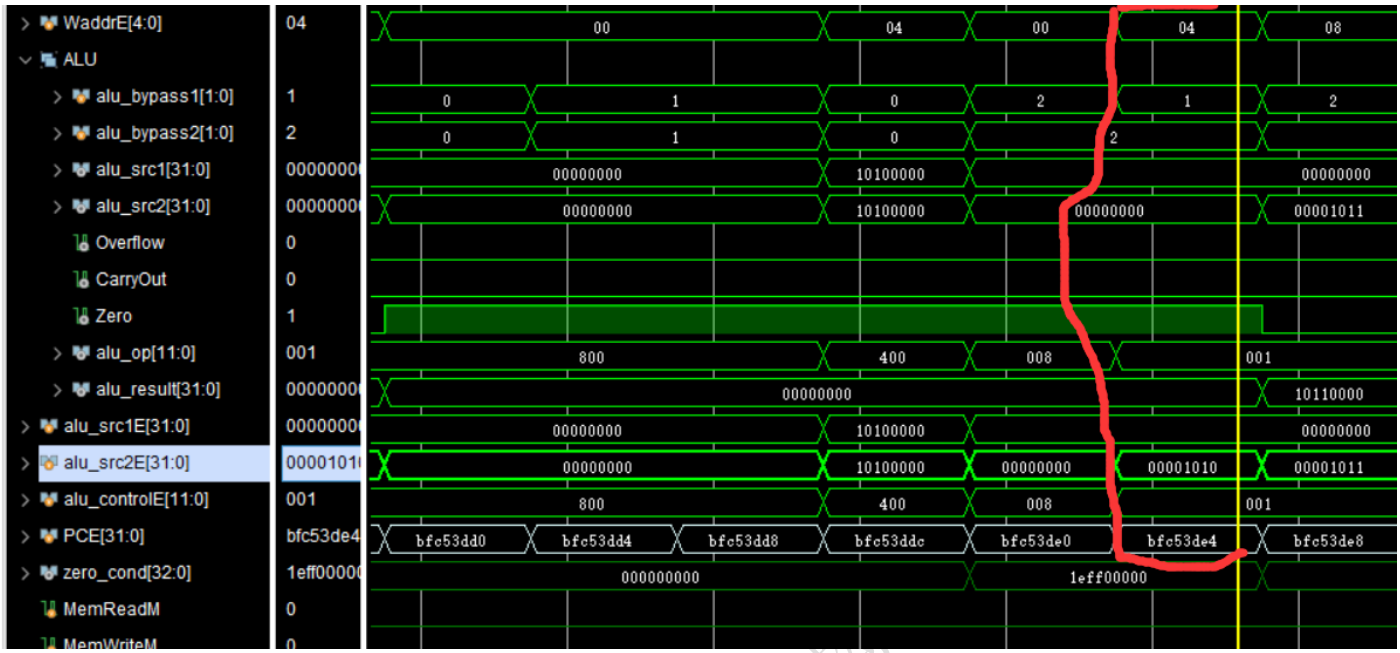
（4）修正效果

利用解决数据相关的旁路技术，成功的解决了数据相关。

3、错误 3

（1）错误现象

写入寄存器堆的结果错误。



图三

（2）分析定位过程

根据反汇编定位错误指令，执行的是高位加载，而且发现在流水线的寄存器里传递的值是对的，但是进入 ALU 的值是错误的，根据 ALU 赋值逻辑，找到错误。

（3）错误原因

构建数据旁路的时候判断条件少了。

（4）修正效果

修正判断条件，纠正了错误。

4、错误 4

（1）错误现象

写入寄存器堆的结果错误。错误波形图见图四。

（2）分析定位过程

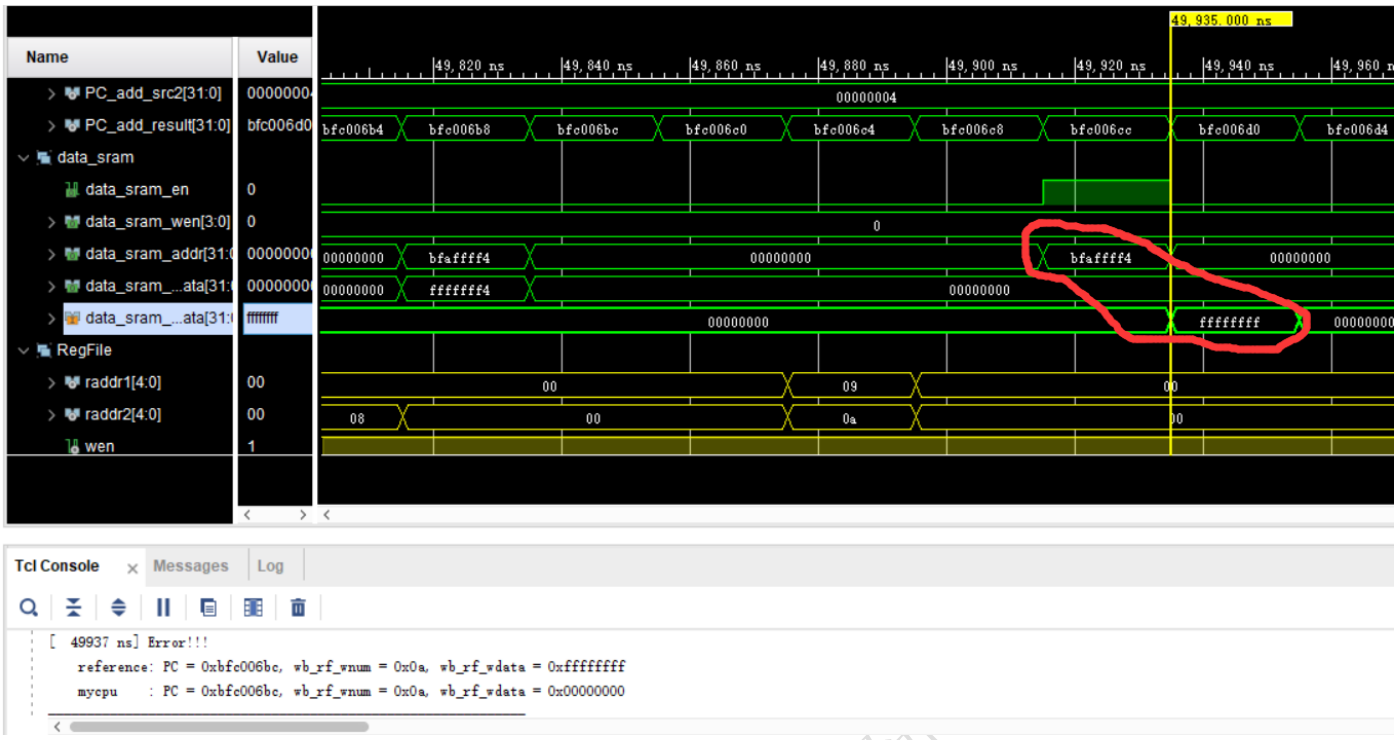
根据反汇编定位错误指令，执行的是 load word 指令，首先验证读内存数的地址和读出的数据是否正确，发现错误。

（3）错误原因

读内存的地址和读出的数据都正确，但是设计的时候忘记了读数据的返回会延迟一拍，导致流水的时候写入 ResW 的数据错误。

（4）修正效果

修改了写回的逻辑，在写回的时候，如果是 load 指令，就不在由 ResW 写回，直接由读出的数据写回。



图四

5、错误 5

（1）错误现象

一直运行，不打印功能点 PASS，也不结束。

（2）分析定位过程

第一个功能点 PASS 都没打印，先去找第一个功能点 PASS 是在哪个位置，根据龙芯的波形图对照，找到打印第一个功能点的地方，发现数码管累加的地方并没有累加，然后找到附近的反汇编指令，废了好大劲，发现附近独特的指令只有 sw，对比数据之后，找到错误。

（3）错误原因

sw 指令写入内存的数是 rdata2，但是计算的时候，我在译码阶段把 ALU 的操作数按照 alu_src_sel 选择之后，保存了操作数，没有保存 rdata2，写入也是将 ALU 的第二个操作数写入，导致写入数据错误。

（4）修正效果

新添加一个寄存器，存储 rdata2，成功的修正了错误。

四、实验总结

五级流水好难啊，写了这么久才写好！！！

想要一下子全部想明白几乎是不可能的，还有好多 bug 只有在测试的过程中才能发现。

分支延迟槽按照胡老师研究生那本书的 CPU 结构，很好处理啊。