**中国科学技术大学计算机学院**

**《计算机组成原理实验》报告**



实验题目：\_\_\_\_流水线CPU设计\_\_\_\_

学生姓名：\_\_\_\_\_\_\_\_徐奥\_\_\_\_\_\_\_\_\_\_

学生学号：\_\_\_\_\_\_PB20061343\_\_\_\_\_\_

完成日期：\_\_\_\_2022年4月27日\_\_\_

计算机实验教学中心制

2020年09月

【实验目标】

1. 理解流水线CPU的结构和工作原理
2. 掌握流水线CPU的设计和调试方法，特别是流水线中的数据相关和控制相关的处理
3. 熟练掌握数据通路和控制器的设计和描述方法

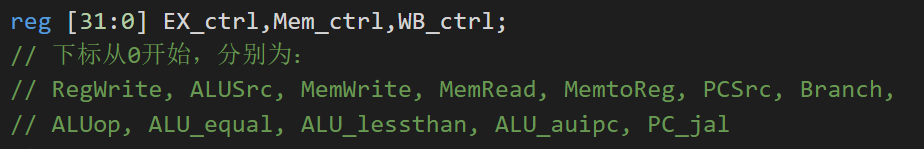
【实验内容】

**题目1**.设计无数据和控制相关处理的流水线CPU

在单周期CPU的基础上进行设计。相较于单周期CPU，流水线CPU每个时钟周期执行一个流水段，将结果存储到当前流水段与下一流水段之间的寄存器，而当前流水段的数据来源于上一个流水段和本流水段之间的寄存器。简单而言，每个流水段在执行时，只需要考虑将前一个寄存器的值读出来，根据控制信号对数据做相应的处理，将结果存到下一组段间寄存器。

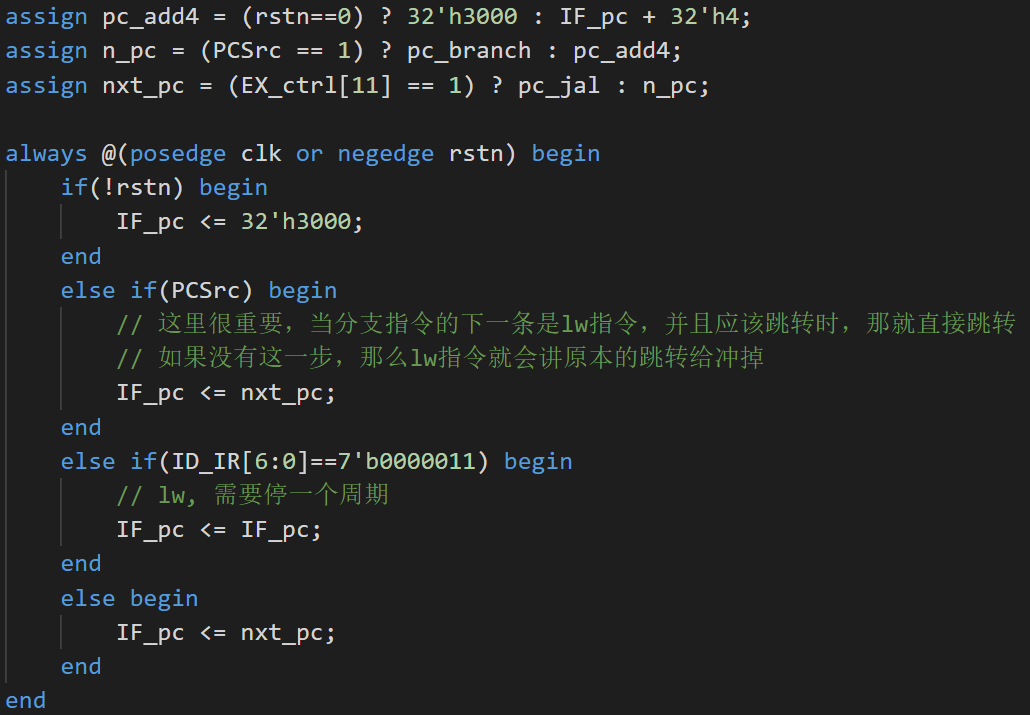
流水线共分为五段，IF、ID、EX、Mem、WB，每两个流水段之间加寄存器，传递下一流水段需要的数据和控制信号。

首先，我选择将所有控制信号逐次向后传，使用一个32位寄存器型变量进行存储。控制信号分别为：

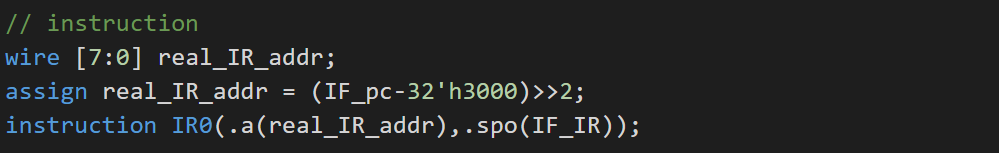


**图1**

**IF段**，完成更新PC和取IR。指令存储器在具体实现中使用ROM。核心代码如下：



**图2**

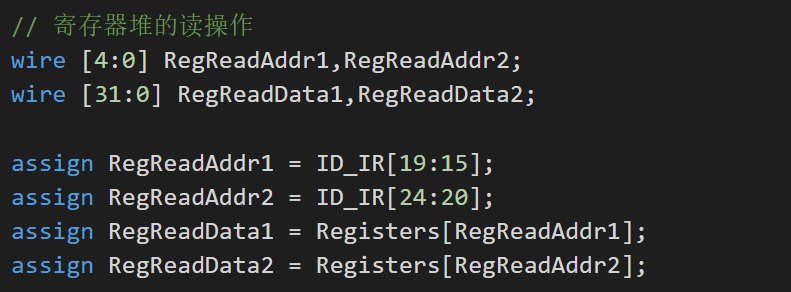


**图3**

然后传PC和IR到下一流水段。

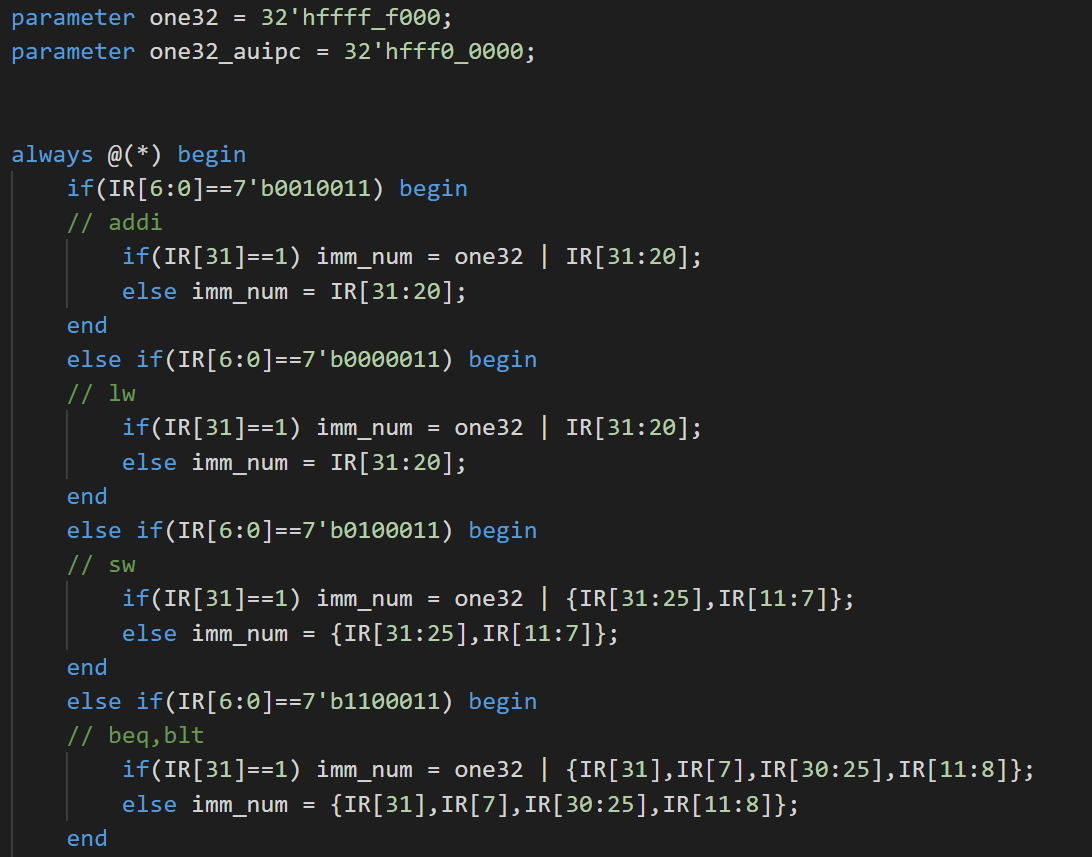
**ID段**，完成寄存器读操作，计算立即数，根据IR求出部分控制信号，还有部分控制信号需要在EX段执行以后才能获取，比如两操作数的大小关系。

寄存器读操作：

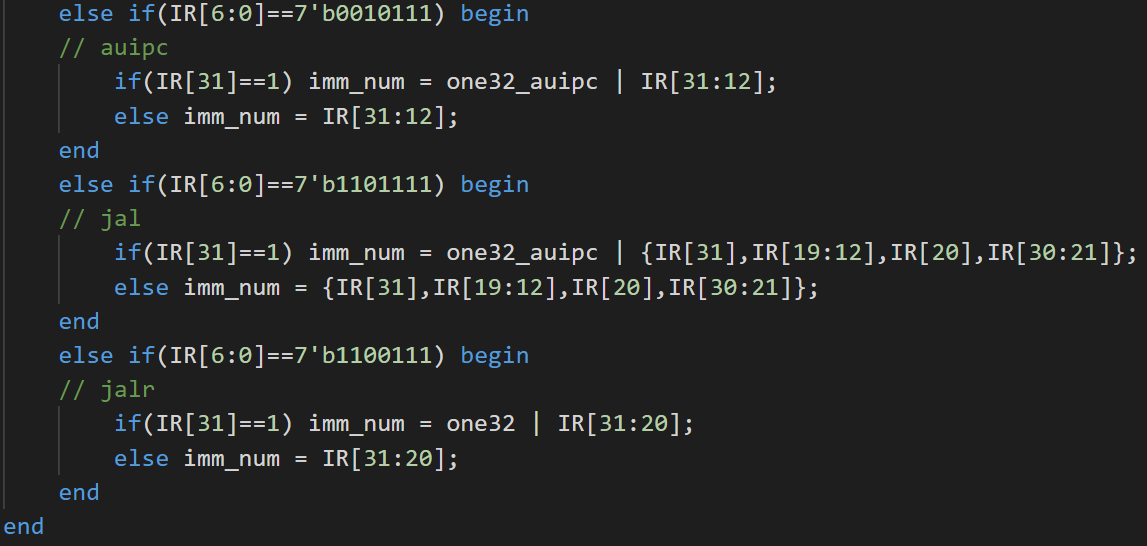


**图4**

计算立即数的模块:

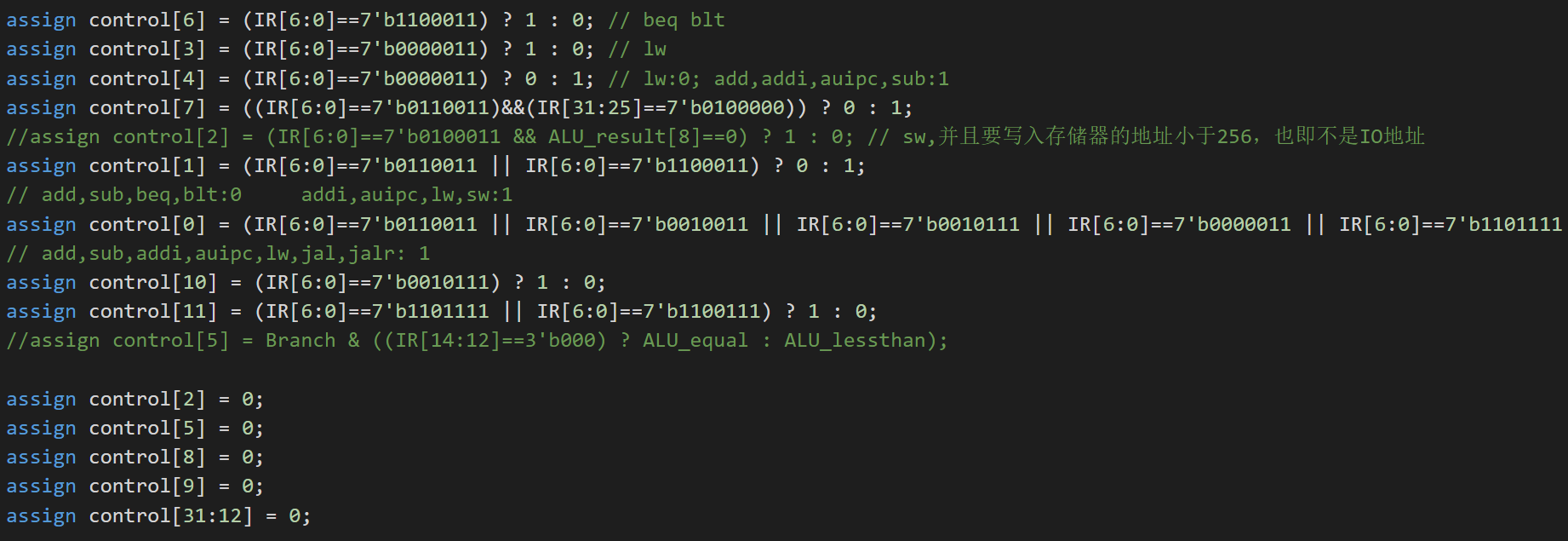


**图5**



**图6**

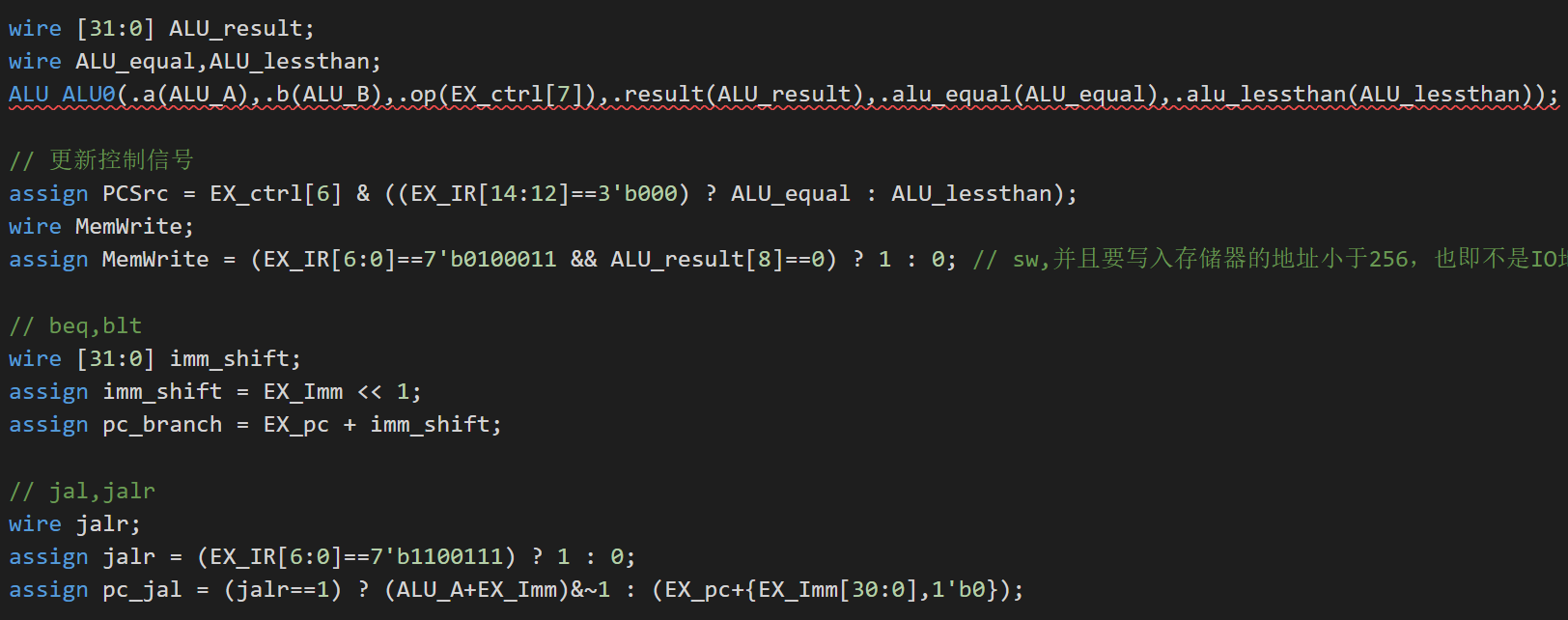
计算控制信号的模块:



**图7**

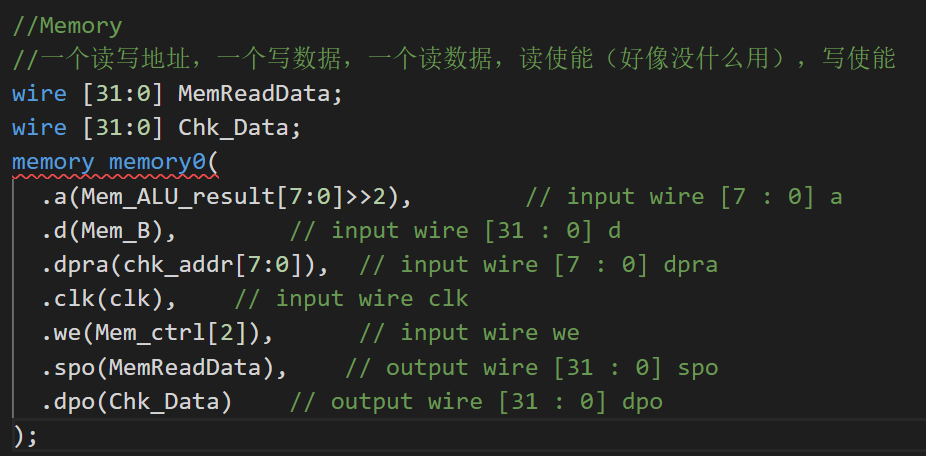
然后，传PC、IR、A、B、Imm、control到下一流水段。

**EX段**，进行ALU的操作，如果是条件跳转指令，那么将根据ALU的结果决定是否跳转，如果是jal、jalr指令，则直接修改nxt\_pc。



然后传control、ALU\_result、B、IR到下一流水段。

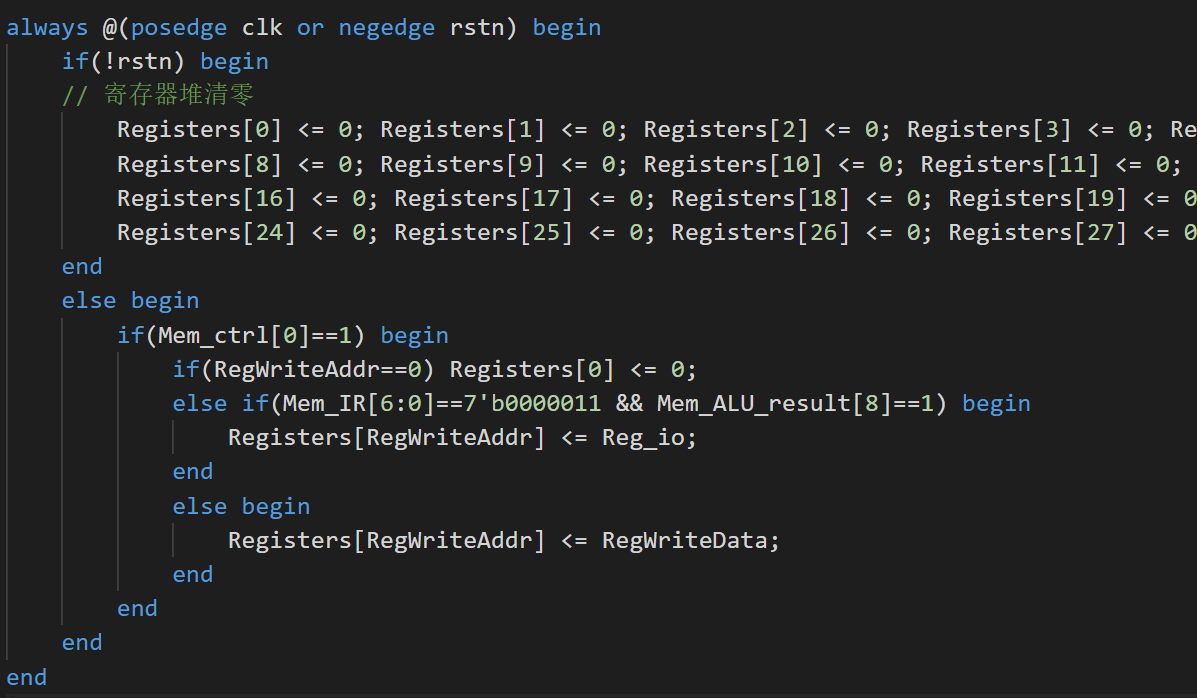
**Mem段**，完成Memory的读写操作。



**图9**

然后传control、MemReadData、ALU\_result和IR到下一流水段。

**WB段**，寄存器的写回操作。但实际上写入寄存器的数据准备工作是在Mem阶段完成，写入是在进入WB的时钟上升沿完成。



**图10**

然后是处理IO操作。MMIO 的起始地址是 32'h0100，这样的好处就在于可以避免地址太大，而需要额外加指令。其他操作与单周期CPU区别不大，差别主要有两处：单周期CPU的写数据到外设，是在一条指令执行完后可以看到外设的结果，而流水线CPU是在那条指令Mem段结束时看到结果；CPU的IO输入数据的准备是在指令执行的前一个周期，而流水线CPU的IO输入数据的准备是在对应指令的Mem段，也即WB流水段的上一段。

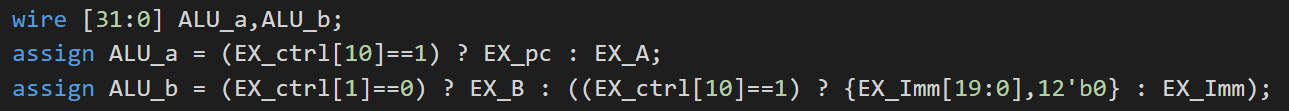
最后，处理Debug总线，这里与单周期基本相同，唯一的更改是Debug总线返回的pc是WB阶段的pc，这样设置的好处就是可以根据返回的pc值设置cpu连续运行的断点。

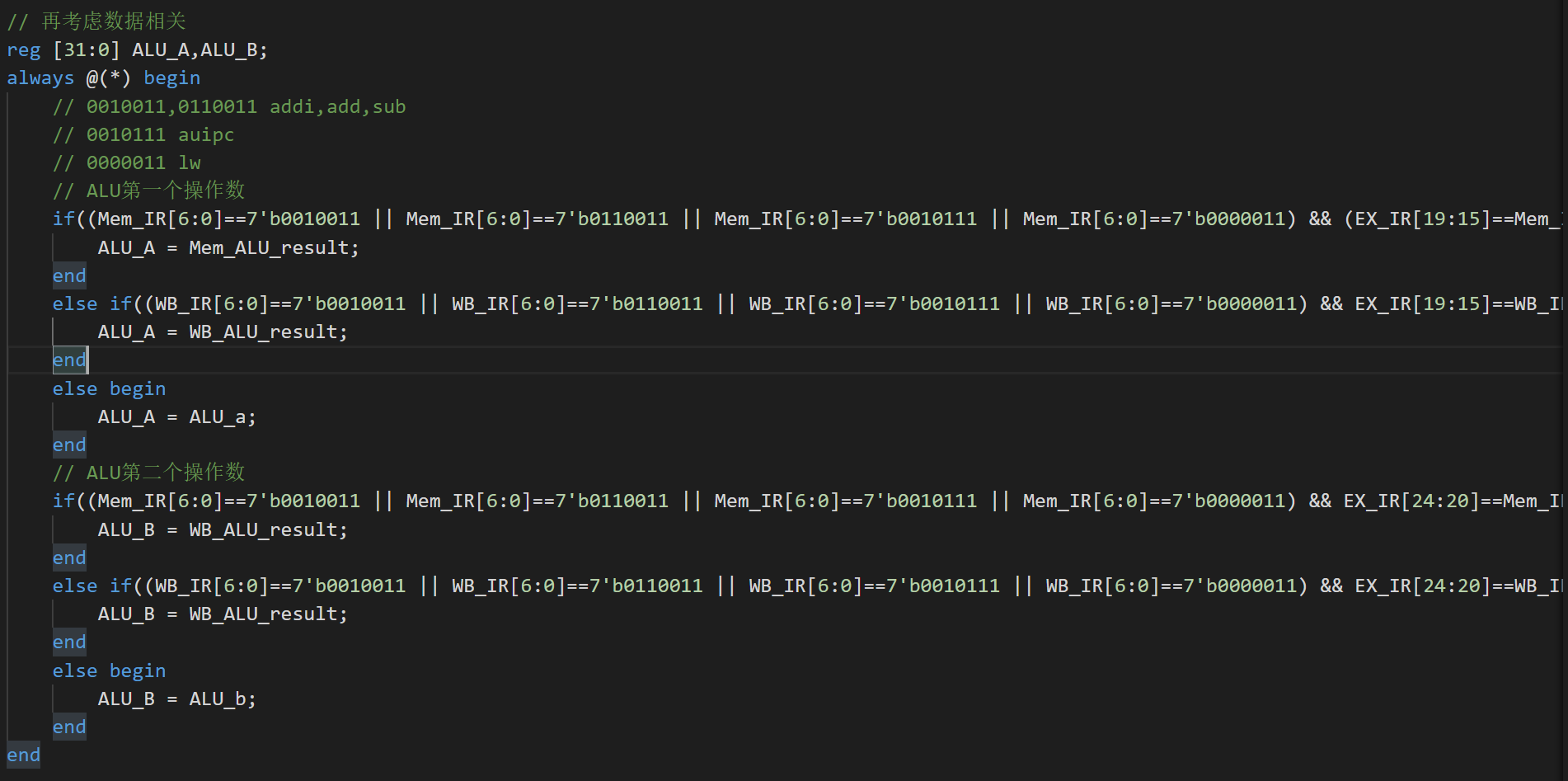
还需要处理结构相关，共两个方面：存储器相关处理：哈佛结构（指令和数据存储器分开）；寄存器堆相关处理：同一寄存器读写时，写优先（Write First）。

**题目2**.设计仅有数据相关处理的流水线CPU.

此次实现的CPU只会出现写后读的数据相关，也即之前指令的写回寄存器操作还没完成，之后的指令就读取了这个寄存器的值。对此的处理有两个：对于lw指令，在其后停止一个周期；对于其他写后读的操作，使用数据定向技术，将执行结果提前传递至之前流水线。

主要涉及的写操作有addi,add,sub,auipc,lw，这些操作均会对寄存器有写入。而读取寄存器是在EX阶段，这是就需要特别判断待读取的寄存器地址是否是其之前两条指令的写回寄存器地址，如果是，那就直接将之前指令的执行结果传递过来。核心代码如下：

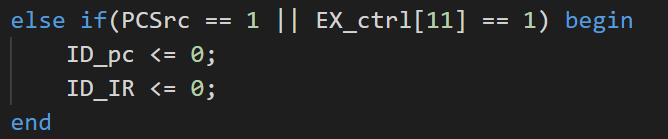




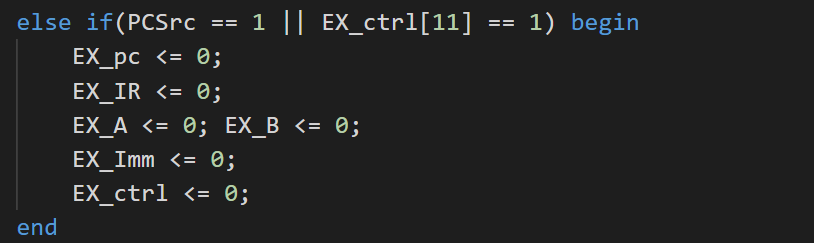
**图12**

**题目3**.设计完整的有数据和控制相关处理的流水线CPU.

也即在题目2的基础上增加了控制相关。当执行无条件跳转以及条件分支条件满足时，需要更改pc值到跳转的目标地址。跳转指令执行到EX段时，会更新nxt\_pc，但此时顺序排在跳转指令后的两条指令已经进入流水线，需要进行清除。具体操作是清除段间寄存器。代码如下：

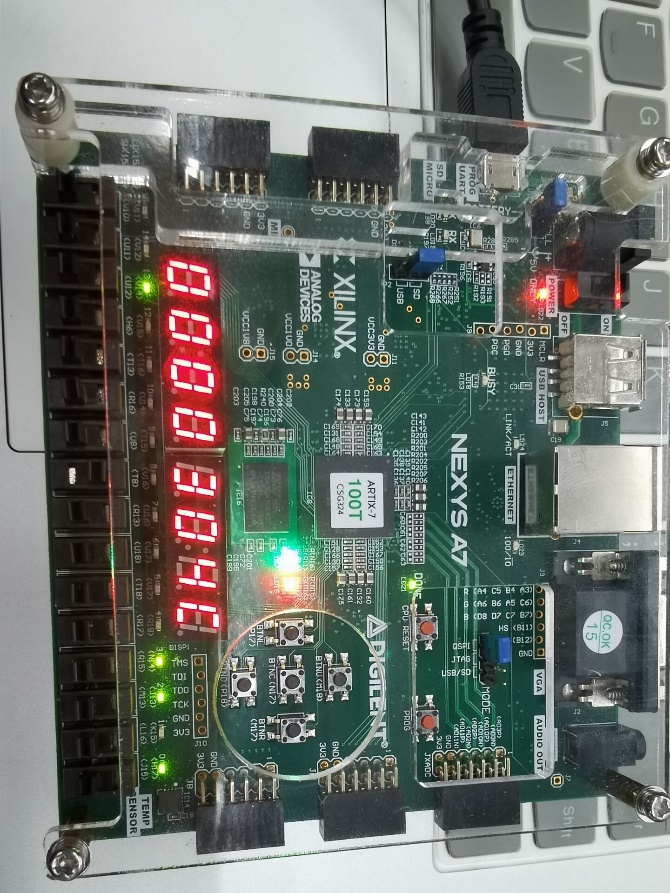


**图13**

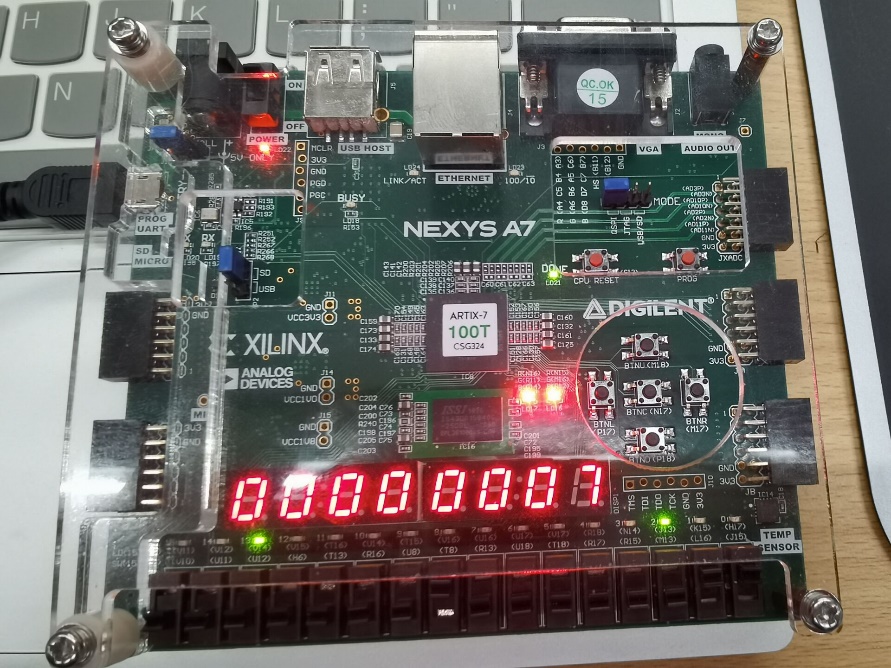


**图14**

至此，基本实现了处理了结构相关、数据相关和控制相关的五级流水线。用Lab3的排序程序进行测试，这个程序是将存储在Memory的数据按照升序进行排序。首先将断点设置在代码的最后一条指令，然后点击连续运行。指令执行情况如下：



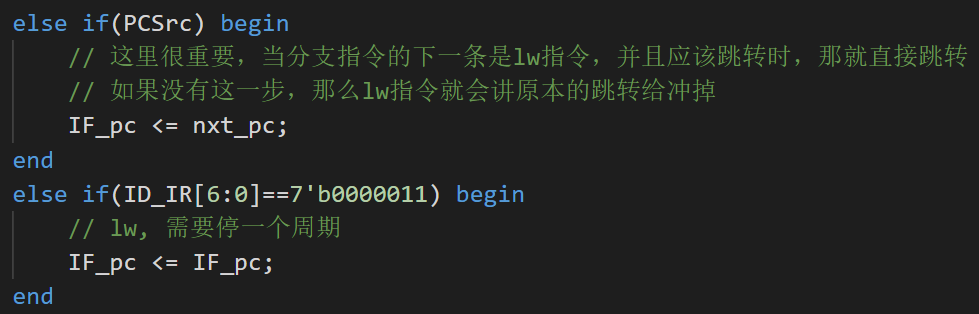
**图15**



**图16**

我完成本次实验所花费的时间，大概可以对半分，一半是写完其他所有内容，另一是寻找一个隐秘的BUG。这个BUG来源于跳转指令+lw指令的情况。根据我的代码逻辑，在lw执行到ID流水段时，会将lw的下一条指令（此时在IF段）重新读一遍，具体实现是令IF\_pc仍为IF\_pc,而不是nxt\_pc，再配合上一些清除段间寄存器的操作，即可实现lw后的指令暂停一周期。而如果是跳转指令+lw指令的组合，lw指令执行到ID段时，跳转指令执行到EX段，如果需要跳转，那么EX段的跳转指令就会修改nxt\_pc，来实现跳转。这时就会出现问题：跳转指令的跳转操作，是通过修改nxt\_pc来实现，而此时处于ID段的lw指令，会阻止IF\_pc被赋值为nxt\_pc，等lw的一个周期的暂停结束后，nxt\_pc又发生了变化，已经不是跳转的目标地址了，于是就出现了问题。

这个BUG比较隐秘，还是多亏了有PDU，可以单步执行，然后查看各个寄存器以及pc、IR的值。这个BUG的修改很简单，如果要发生跳转，那么对pc赋值首先是nxt\_pc，然后才是lw的暂停一周期操作。核心代码如下：



**图17**

这样一个先后的顺序，解决了上述BUG

【总结与思考】

1. 本次实验难度适中，流水线CPU可以直接基于已经实现的单周期CPU去做。
2. 本次实验收获较多，最重要的是可以深入到代码实现的具体细节上，体会流水线cpu是如何运作的。而且verilog是硬件编程语言，与实际的电路比较接近，所以其实是从电路的角度体会了流水线cpu的执行过程
3. 本次实验也帮助我更深入理解了各种相关，尤其是数据相关，也从代码层面体会到了数据定向是怎样一个操作。
4. 最后，本次实验给出的数据通路和控制器图并不全，缺少jal、jalr、auipc的数据通路和控制信号，建议之后的实现文档将这一部分进行补充。

【附录】

Lab5\_q1.srcs完整流水线的代码文件，包括设计文件和引脚约束文件