**流水线实验报告**

组员：陈星霖、温冬、梁鹏

**一、简介**

在本次的实验中，我们采用xilinx的vivado2018.1作为我们的IDE，以进行代码的编写以及仿真测试。考虑到多人开发时存在的版本问题，我们使用了github作为我们的版本管理工具。

**二、GitHub版本管理**

在本实验中，我们采用GitHub进行代码版本的管理，在GitHub Desktop中，可以很好地看到组员对项目的贡献。根据分工，小组成员在规定的时间内很好的完成了自己的任务。

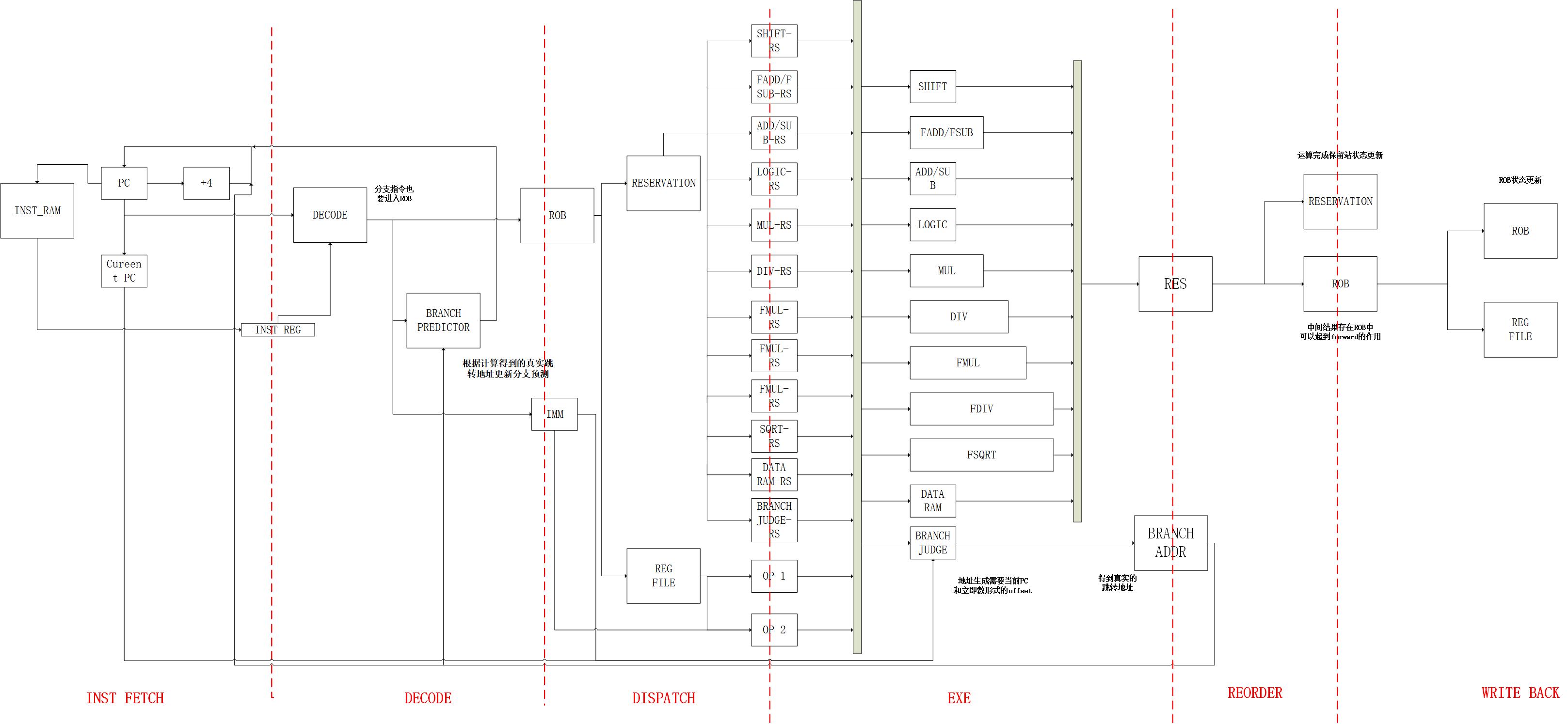


**三、实验步骤**

（一）流水线的设计

考虑到后面需要实现Tomasulo算法，我们小组将原来的五级流水线中的ID段的译码和取数分开，以方便我们分配保留站。且考虑到程序执行可能乱序提交的问题，还需加入一个Reorder段来在写回之前处理好排序问题。又考虑到在RISC-V指令集中，只有LS指令会进行访存操作，于是我们将EX段和MEM段合并，将LS指令的存取操作放到了指令的EXE阶段完成。于是我们设计了一个六级流水线，每一段分别处理取指、译码、取数及分配、运算执行、重排和写回。其中取数及分配（dispatch）段主要完成保留站的分配任务和操作数的取出任务。

以下是我们的六段流水线示意图

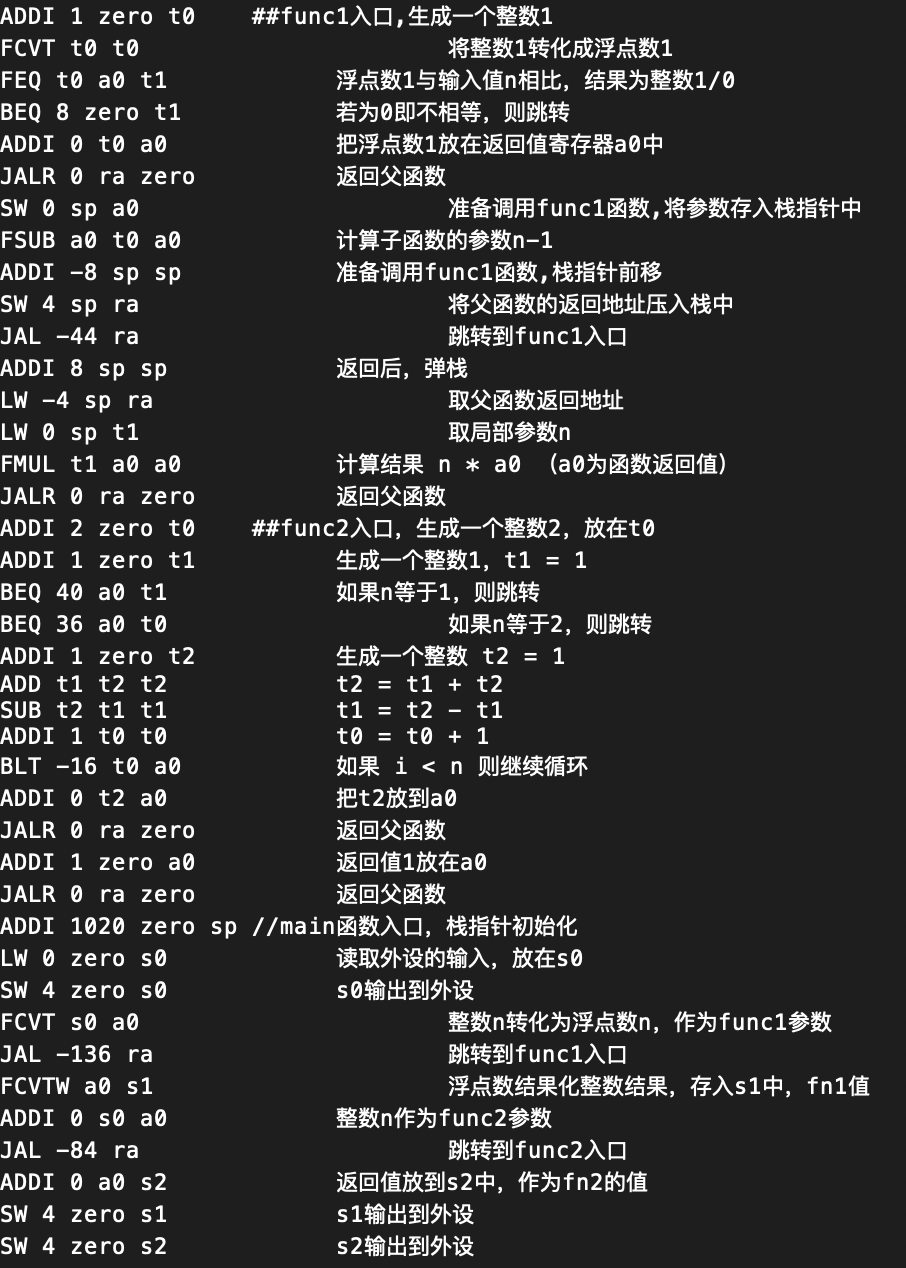


在本周提交的实验中，由于还没有实现Tomasulo算法，我们暂时没有实现Reorder段，因此测试的时候是一个去掉了Reorder段的五级流水线。

（二）测试程序的翻译

先理清代码的逻辑，然后用gcc进行编译，得到对应的符合ABI规范的其他指令集的汇编语言代码。再对相应的指令进行替换，得到一个RISCV版本的汇编程序。然后根据该汇编程序和我们的编码方案，得到对应的机器代码。

**注：在本实验中，我们的汇编格式为op rs1 rs2/imm rd**



（三）Verilog代码的编写

按照事先画好的结构图，进行流水线verilog代码的编写。在流水线的实现中，主要有两个问题需要解决：数据的旁路问题以及分支时流水线的冲刷问题。

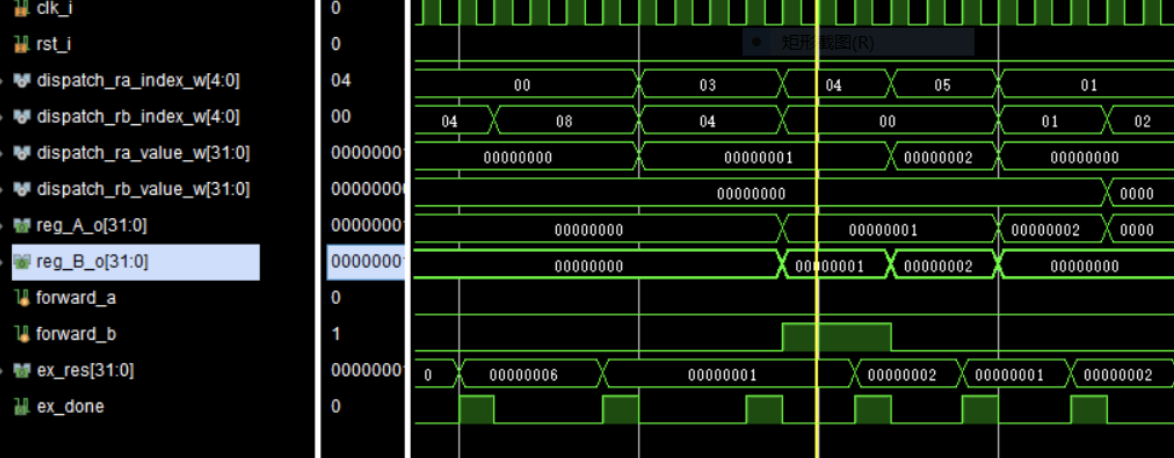
（1）数据的旁路

设计了数据相关的两条指令，

LW 8 x0 x4 # x4 = ram[8]，值为1

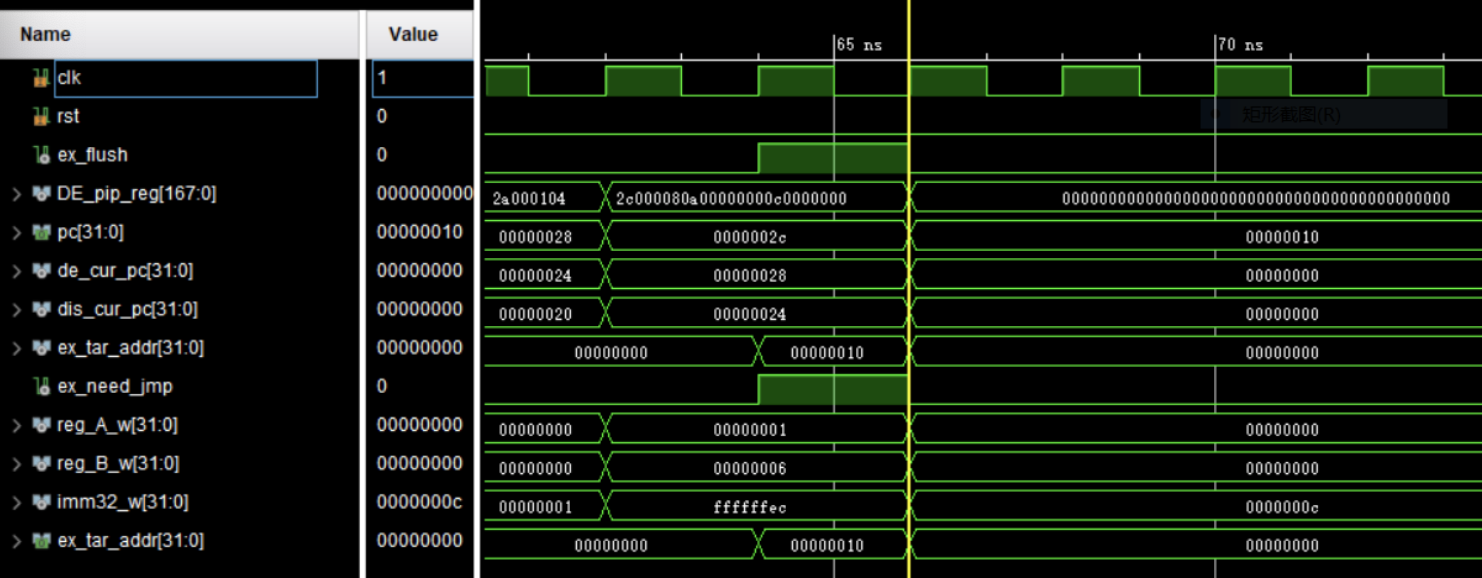
ADDI x3 x4 x5 #x5=x3+x4，其中x3的值为1

从图中可以看出，由于x4数据相关，我们置forward\_b信号为1，同时用组合逻辑将reg\_B\_o的值从旁路ex\_result结果中取，而非dispatch\_rb\_value\_w中的0。



（2）流水线的冲刷

在EX段确认分支指令需要进行分支之后，需要将EX段之前的流水线中的寄存器进行冲刷，以下为结果的仿真示意图，可见当冲刷信号ex\_flush来了以后，在时钟上边缘来了之后对应的流水寄存器会被置零，且同时IF模块继续取跳转pc对应的指令，分支开销为2个周期。



（四）测试

测试函数1 （阶乘）

在阶乘的测试中，为了扩大数据的表示范围，我们拓展了浮点操作指令集，使得我们可以计算更大的结果。

测试函数2（斐波拉契数列）

斐波拉契数列的实现比较简单，注意好逻辑即可。

整体测试：我们将汇编代码对应的机器代码放到了指令存储器中进行模拟测试。并且用load和store指令模拟了外设的访问操作。最后我们编写的cpu可以支持最高输入值为34的阶乘结果。如下图所示，r9值为7f5e1bc6，为阶乘的结果；r10值为5702887，为第34个斐波拉契数。

