实验 2 报告

学号 2016K8009907007
姓名 黄熠华

一、实验任务（10%）

设计五级流水线 CPU，每条指令经过取址、译码、执行、访存、写回五个阶段，添加乘除法器以及相应指令，考虑数据相关，最终仿真验证功能正确、上板通过并进行性能测试。

二、实验设计（30%）

五级流水线是以 CPU 内部硬件分工来划分五级的，因此需要将 CPU 划分为五个模块。方便起见我在 my_cpu.v 文件里用注释的星号行划分出五个模块。五个模块的硬件分别以 PC, 译码器, ALU, 访存接口, RegFile 为主体，各自进行相关阶段的操作。

注意到流水线的五个阶段并不是相互独立的，而存在着许多数据依赖，因此有必要将一些数据在五级中某些级之间进行流动。译码阶段得到的控制信号，执行阶段得到的计算结果，访存阶段得到的数据等，都需要流到后面的阶段并加以使用。所以流水线 CPU 的“流水”体现在指令以及一些控制信号的流动。

代码中定义了五条 done 信号，用以表明当前阶段是否结束。不完全参考老师上课所讲授的，我给每个状态设定了 valid, ready_go 和 allow_in 信号，并以将它们链式关系连接。一个阶段的 valid 信号拉高表明其包含了有效内容，ready_go 信号拉高表明该阶段的指令在下一时钟上升沿会流动到下一个阶段，allow_in 信号的拉高意味着这个阶段在下一时钟上升沿可以接收来自上一级的指令。其中 ready_go 和 allow_in 的逻辑赋值为 ready_go_i = done_i & valid_i & allow_in_i+1，allow_in_i = ~valid_i | ready_go_i。

在考虑了访存、乘法器未出结果给译码阶段需要读取的寄存器等阻塞条件后，将相关逻辑赋值到译码阶段的 ID_done 上，使得译码阶段可以在合适时刻阻塞。执行阶段若是除法，则阻塞等到除法器发出 cmoplete 信号再拉高 EX_done。这样阻塞情况就都能够实现了。

有一处值得注意的是，如果 ID 阶段发生阻塞，在下一个上升沿 FI 阶段取到的新指令会不可避免的出现在 inst_sram_data 端口，造成了在 ID 流通之前，不能根据该输入端口进行译码。因此，需要在 ID_done 拉低时将 inst_sram_data 保存到特殊寄存器 IR_ID 中，并添加判断 ID 阶段是否进入阻塞状态的寄存器，在 ID 阶段阻塞时，选择 IR_ID 而非 inst_sram_data 进行译码。

数据前递技术有助于避免阻塞或减少处理器的阻塞时间，提高处理器工作效率。代码中利用了寄存器将 EX 以及之后各个阶段的即将写入寄存器堆的数据保存起来，在判断需要前递的条件下，将相应的数据直接提供给译码阶段即可。需要注意的是，寄存器堆之外的 HI 和 LO 两个特殊寄存器同样需要考虑数据相关。另外在数据前递过程中，需要考虑前递的优先级，越晚产生的数据优先级越高。

三、实验过程（60%）

（一）实验流水账

10.15 : 完成实验二第三阶段

（二）错误记录

1、错误 1

（1）错误现象

写寄存器错误。

（2）分析定位过程

分析汇编代码后发现是测试 BGEZAL 的代码段,便检查了相关代码。

（3）错误原因

看错指令要求，误认为在跳转条件不成立时不能将 PC+4 写入 ra 寄存器，而要求却是无论如何都将其写入。

（4）修正效果

功能仿真通过。

（5）归纳总结（可选）

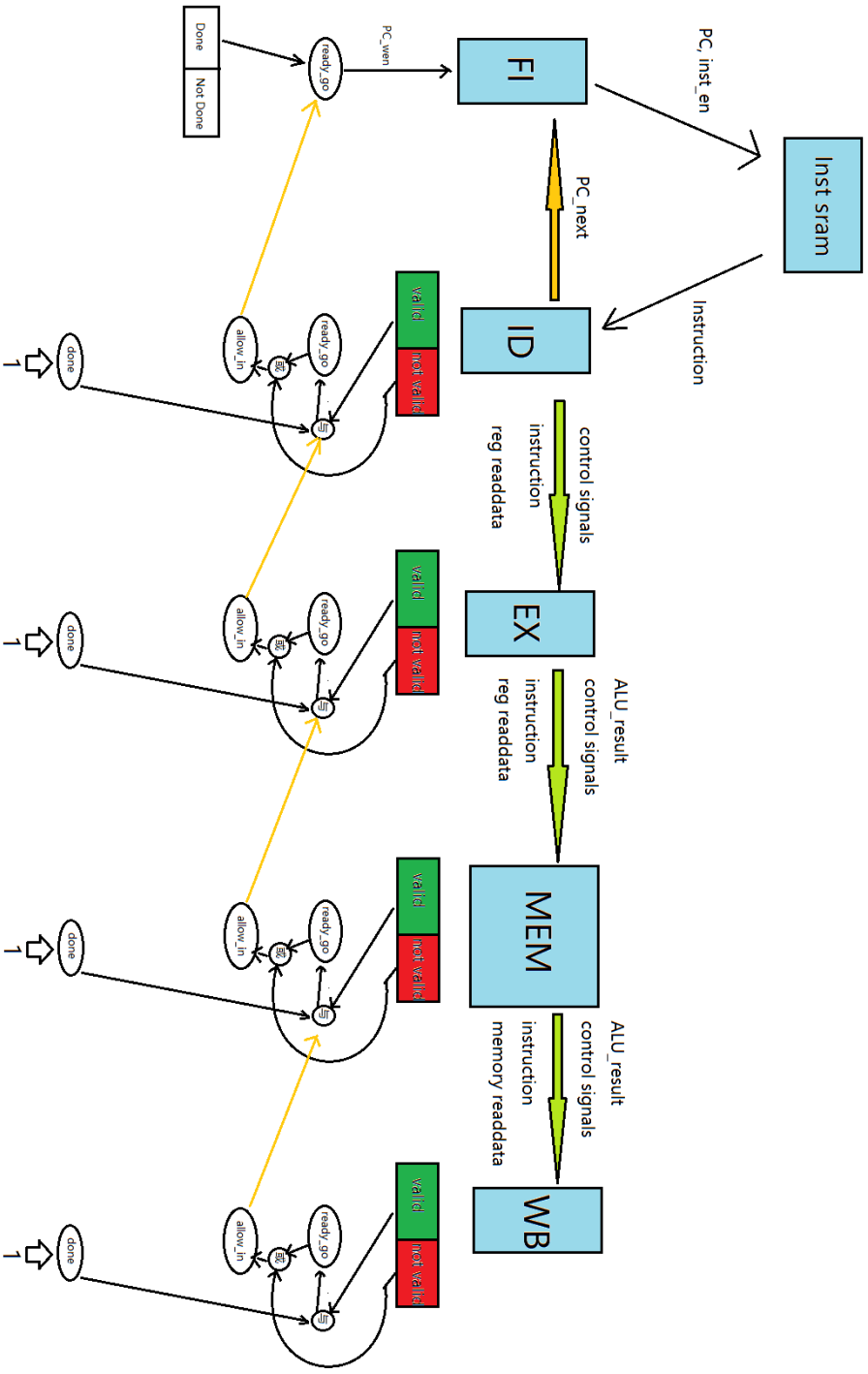
认真阅读指令内容是写好处理器的基本要求。

四、实验总结（可选）

本次实验任务量最少，基本实现了较为完备的 CPU，但是其性能却没达到预期要求，还需对其进行线路优化，提升主频。

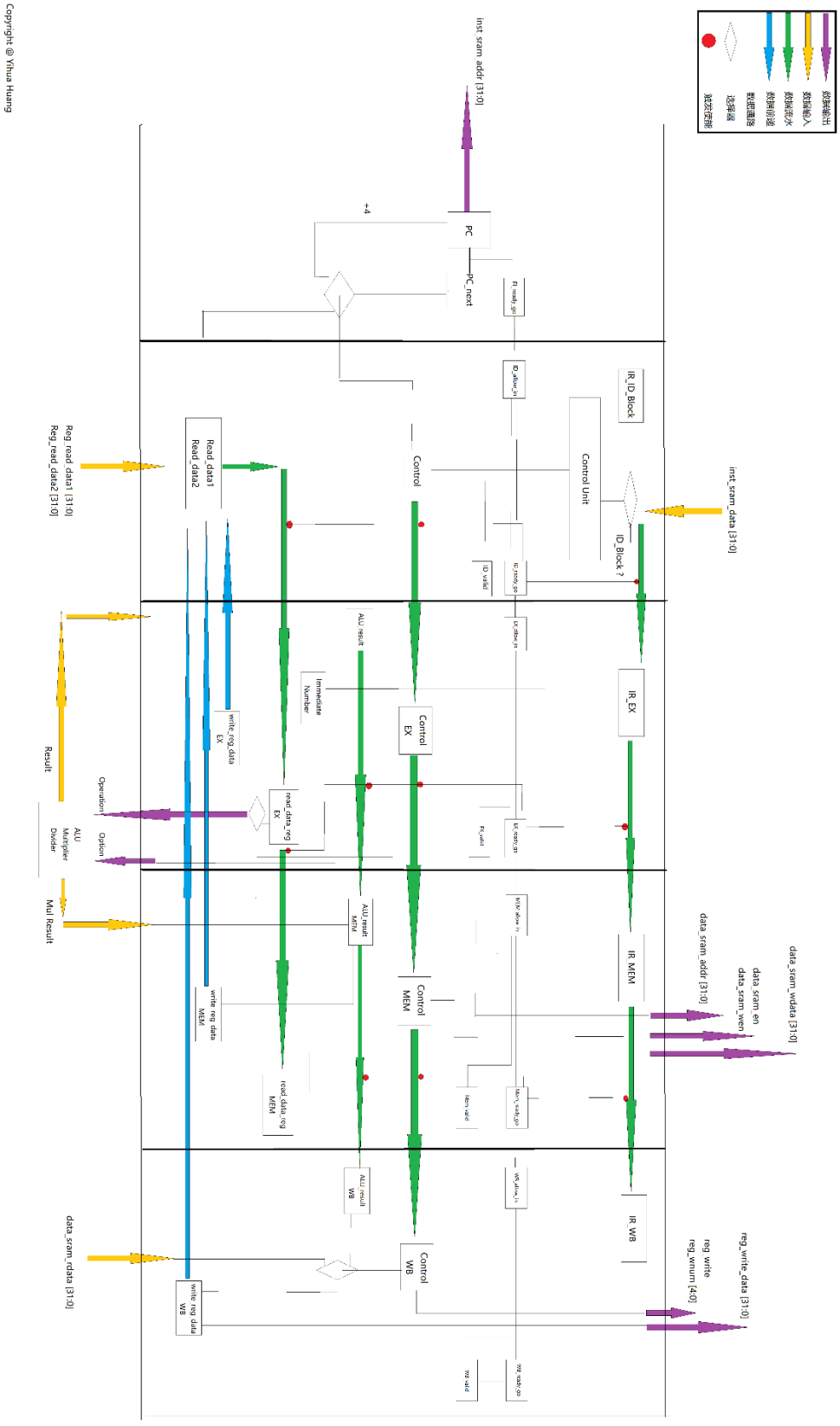
五、逻辑结构图

流水信号设计图如图一所示



图一

CPU 设计如下图


五级流水CPU结构简图
Copyright @ Yihua Huang
inst_sram_addr [31:0]
PC
PC_next
+4
IR_ID Block
Control Unit
ID_Block ?
inst_sram_data [31:0]
Control
Reg_read_data1 [31:0]
Reg_read_data2 [31:0]
Read_data1
Read_data2
IR_EX
Control
EX
Immediate Number
ALU_result
write_reg_data
EX
read_data_reg
EX
Result
Operation
Option
ALU
Multiplier
Divider
Mul Result
IR_MEM
Control
MEM
ALU_result
MEM
write_reg_data
MEM
read_data_reg
MEM
data_sram_addr [31:0]
data_sram_en
data_sram_wen
data_sram_wdata [31:0]
IR_WB
Control
WB
ALU_result
WB
write_reg_data
WB
data_sram_rdata [31:0]
reg_write
reg_wnum [4:0]
reg_write_data [31:0]

图二