# LAB2 Report

钟作奇 2000012741

# 1 设计思路

本次lab主要关注IF与ID两个过程。大致上，我仍然按照助教在tutorial中给出的框架搭建结构。

## IF Module

取指令模块由PC模块`IF_PC`，ADD模块`IF_ADD`， Instruction Memory模块`IF_Instr_Mem`。PC模块是上升沿触发的D寄存器（一开始没太理清楚意思，后面一边做一边figure out），ADD模块是一个可变长度的加法器，在本设计中采用`WIDTH=32`。 Instruction Memory使用上个lab中写的RAM来进行创建；吸收上次讲课助教给出代码的优点，将一些特殊情况（case中的default等）进行了完善。

## ID Module

Decode模块相对于IF模块的设计更难一些。挺有意思的是之前在另外一门课上写了一个编译器（用scala语言），对于这次的lab有一种熟悉的感觉（虽然还是写了巨久）。模块由立即数产生模块`ID_Imm_Gen`,两个控制信号产生模块`ID_Control_Unit`、`ID_ALU_Control`,寄存器堆`ID_RegFile`。立即数产生器的设计参考了上次助教给出代码的写法（用`signed`转换后进行算数右移`>>>`）。

两个控制模块分工合作，ALU_Control主要用来产生ALU的控制信号，其中`alu_src1/2`用0来表示以RegFile里读出的数作为alu_op，1则表示用imm来作为alu_op，这次用alu_op测试所以两个alu_src就暂时没起作用；`reg_write_enable`这个信号在tutorial和code里都没太说清楚什么意思，后面和同学讨论后发现这个信号大概可以认为是指令里有`rd`则为真，与写寄存器的名字配合得也比较好。

对于regfile，这次没有涉及到写，只涉及到读，output.ref文件中的那个29很奇怪。后面发现是当指令中没有rs2时，也将instr[24:20]作为rs2，将其作为地址读regfile中的数所得。这样的写法虽然作为tb有点难理解，但能避免reg2出现坏地址，善。

# 2 Debug经验

这次虽然不是第一次写多文件verilog项目了，但因为代码量的增加，出现了一些其他问题：

- 一开始应该去弄清楚各种命令是什么意思（实际上那个英文的tutorial的结构我实在没弄清楚，后面看的《手把手教你设计 CPU——RISC-V 处理器篇》的附录A感觉舒服多了，虽然具体的信息还是需要从手册上查）；
- debug的过程中，发现在top模块在写其他代码的过程中没有及时更新端口，有的端口没有接上，导致输出很有问题；
- case中拿字符串作索引的话，前面需要加上`这个符号，否则会报错；
- wire应该翻译、理解成“结点”；reg应该翻译成变量，给变量赋值后变量值保持不变；assign是连续赋值，右边信号只要一改变就会重新计算左边信号，适合描述连接关系（并行语句）；
- 由于这次要产生alu_op且我是在alu_control模块中考虑的，所以R_Data1/2是有三端的，做出了一些奇怪的写法……

# 3 提交结果

以下是这次lab的提交结果

答案正确

作者: zzq0219

**Performance**

**score: 100**

## Compile Log

```
WARNING: ./src/ram.v:20: $readmemh(tb/ram_data.hex): Not enough words in the file for the re
quested range [0:65535].
LXT2 info: dumpfile wave.vcd opened for output.
```