P5课下

COO

修改了CTRL，一定要记得修改HAZARD单元！！！

否则就不会给转发了，太致命了，这和P4不一样！

BGEZAL

有符号大于等于0的判断

```
RD[31]！=1'b1 //有符号大于等于零，如果$signed不好用的情况下
$signed(RD1)>=0
$signed(RD1)>=$signed(32'b0)
```

需要修改的有CMP，各级流水线寄存器

~~修改CTRL传入Judge信号~~

一定要记得修改HAZRD模块中的内容，和转发相关！！！

~~将D级判断出来的Judge信号流水，bgezal经过的每一级的REGwrite信号都需要由Judge信号辅助判断，判断是否写入~~

```
(bgezal==1'b1&&Judge==1'b1)
```

注意begzal需要用到rs寄存器，在HAZARD模块中要记得添加！！！

BGEZALL

在bgezal的基础上，回传D级的指令到IFU

```
assign Instr = (Instr_D[31:26]==6'b000001&&Judge!=1'b1)?32'b0:ROM[pc[13:2]];
```

blezalc

和bgezal差不多，只需改一下cmp的判断即可

```
$signed(RD1)<=$signed(32'b0)
```

lrm

对MemtoReg进行了修改，最后WB级的11端接GPR[rt]，从M级的WD_DM接过来，需要输入流水线寄存器中

同时对于A3_M添加了一个并列信号RD_DM[4:0],通过信号的选择，最终汇入Write_Addr进行输入到DM_WB寄存器中

其他没有什么增加

有条件的访存

由于A3在D级不能确定，所以对于lrm在E级和M级都进行了阻塞，应对AT法的条件不足

SWC

R型指令，add rd,rs,rt

注意循环右移，循环左移的要求

用for循环的高级做法

C = A;

for(i = 0; i < B[4:0]; i = i + 1) C = {C[0], C[31:1]}; 这是循环右移的例子

addoi

一定要符号判断大于零$signed(temp)>=$signed(32'b0)

或者用最高位来判断

反码是**原码符号位不变，其余位取反**，或者是对于**原码的绝对值按位取反**

补码是反码加一

所以原码是 补码减一取反，**取反符号位不能变**

temp=temp-32'b1;

temp= {temp[31],~temp[30:0]};

bonall

负数反码：符号位不变，其余数按位取反；
负数补码：符号位不变，其余数按位取反后再加1（反码加1）；

一个数无论正负，它的相反数（的补码）获取方式皆是原数（补码形式）！！！含符号位按位取反后再加1！！！得到的，再次强调区别于负数求补码操作。

A和B互为相反数

(A+B)==32'b0 && (!(A==32'h80000000&&B==32'h80000000))

lbget

需要修改CTRL 增加condition模块

利用condition模块，修改Regdst，**但是由于A3_WB是在D级决定的，所以M级需要增加一个并列的信号，用M级的RegDst去辅助控制ADDr（传入DM_WB级寄存器A3端口的数字）**，最后改一个HAZARD单元，注意在每一级都要用一个信号输进去，充当condition的控制条件。

记得有符号比较大于零，其余就是类似lb的操作

阻塞，一直阻塞到该指令的WB的结束

LWSO

条件存储，我的想法是无脑stall，该指令在E、M时都进行阻塞

除此以外我学到是，一定要看好转发的东西，**即要设置好该条指令需要的寄存器编号以及Tuse，为的是别的指令给这个条转发**，虽然可以不给别的转发，但是一定要给自己转发。

除此以外，**注意D级的RegDst，A3_D的判断是有Regwrite辅助判断的，但是因为在D级还没办法得到Regwrite信号，所以这里的A3是失效的，因此，需要在M级判断好条件之后，和A3_M一同输入到一个多路选择器中，作为WB级的A3，通过满足条件选择**！！！！！！

即所以M级需要增加一个并列的信号

WB级的regwrite也使用条件去判断，这里还要开一条新通路，将RT2的值从M级传到WB级，作为最后写入的使用。