信号汇总：

信号	R-format	lw/lb/lbu/lh/lhu	sw/sb/sh	beq/bne/BLEZ / BGTZ / BGEZ / B	lui	jal	jr	ADDI/ADDIU/ANDI/ORI/XORI/SLT
aluop1	1	0	0	0	X	X	X	1
aluop0	0	0	0	1	X	X	X	1
regdst	1	0	X	X	0	0	X	0
alusrc	0	1	1	0	X	X	0	1
memtoreg	0	1	X	X	0	X	X	0
regwrite	1	1	0	0	1	1	0	1
memread	0	1	0	0	0	0	0	0
memwrite	0	0	1	0	0	0	0	0
branch	0	0	0	1	0	0	0	0
lui	0	0	0	0	1	0	0	0
jal	0	0	0	0	0	1	0	0
jr	0	0	0	0	0	0	1	0
jump	0	0	0	0	0	0	0	0
alusrcA(aluA端的选择)	0	0	0	0	0	0	0	0
mutstart(乘法器开启)	0	0	0	0	0	0	0	0
mutop(乘法器操作)[2:0]	X	X	X	X	X	X	X	X
dmop(数据存储器非对齐访问操作)[2:0]	X	根据指令情况确定	根据情况而定	X	X	X	X	X

syscall的实现

实验要求的syscall只需要v0和a0两个寄存器的值，单独设置一个syscall信号,在读寄存器堆的rs和rt段设置一个二路选择器，如果为syscall指令则将rs和rt设置为v0和a0.
syscall和跳转一样，在译码阶段实现，即读出寄存器值后立即处理。不同的是syscall不能利用旁路，否则会在v0写入前finish造成正确性检查出错，因此阻塞逻辑需要一直阻塞知道ex和mem都没有写回v0/a0的操作

运算器信号

alu,MultiplicationDivisionUnit,处理branch指令的运算器，数据存储器都需要运算信号
aluop:与alu实验中相同(lw/sw类型的指令将aluop设置为100000，因为操作数本身为补码，利用无符号加法即可)
MultiplicationDivisionUnit_op:与实验指导中对部件的解释一致。
DataMemory_op：
011:LB
100:LBU
001:LH
010:LHU
000:LW
101:SB
110:SH
111:SW

在dm中会先取出lw的结果并计算所有I指令的输出再根据op确定最终输出
branchop:由于branch指令较少，直接在branch判断条件的部件中解析,具体键TopLevel中的Equel_unit模块

旁路

需要对条件跳转以及ex阶段的alu运算设计旁路（mem阶段的B在ex阶段顺便修正），syscall不用旁路(可以忽略这块代码)
旁路的逻辑与课件相同，唯一不同的是由于实际汇编中会对零号寄存器写入非零值，因此需要将对零号寄存器的写考虑在外

阻塞

由于乘除法的存在，需要将阻塞分为id阶段阻塞和ex阶段阻塞。
ex阶段阻塞：发生在乘法器的busy信号为1且开启乘法器时。清空ex/mem,冻结id/ex,if/id,pc
id阶段阻塞：分为两种情况：第一种是id阶段为跳转指令，只需判断ex阶段是否为lw类型执行就可以；第二种情况是id阶段为跳转指令，需要判断mem阶段是否为lw指令，ex阶段是否为lw或者r指令。清空id/ex,冻结if/id,pc。
发现两种情况对于id/ex的操作有冲突，在发生冲突时优先处理ex阶段