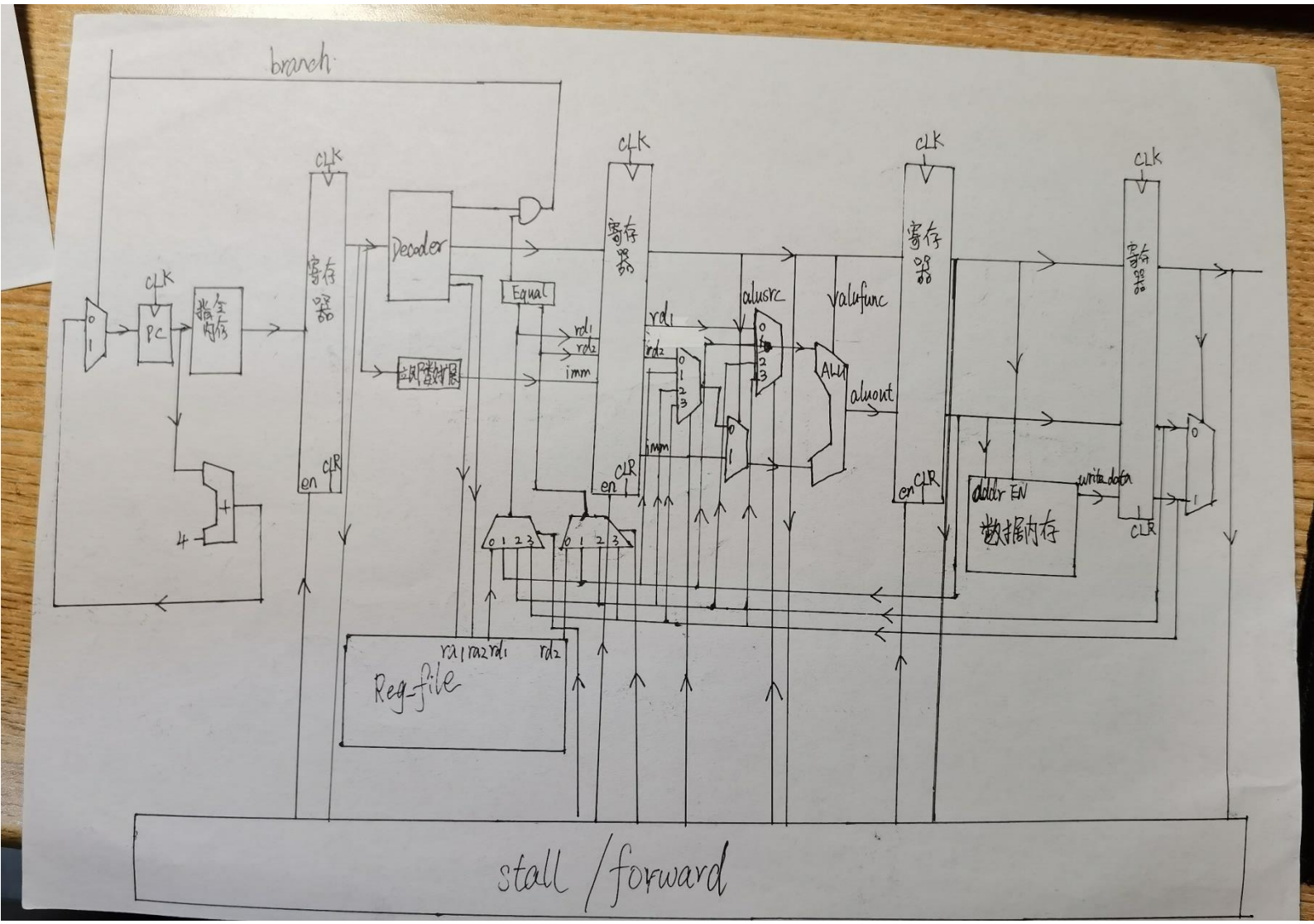
一、电路图



二、流水线冲突的情况及解决方案

1.控制冒险

如果当前指令为跳转指令，在 decode 阶段检出，而此时 fetch 阶段已经取出了原来的 pc+4 为错误的指令地址

解决方案：在 decode 阶段发现这是一个跳转指令则将一个选择信号与目标的指令地址传回 fetch 阶段，并将 fetch 与 decode 之间的流水线寄存器清零（插入气泡）——因为此时里面存的是从错误的地址取出的指令。

电路实现：在 pc 前插入一个二路选择器，控制信号为来自 decode 阶段的跳转指令，选择下一个 pc 是 pc+4 还是需要跳转的指令地址。

2.数据冒险

decode 阶段从寄存器中取数时，前几条指令可能存在往相同的寄存器中写入的情况，此时前三条指令尚未 writeback，寄存器中数据尚未更新，则 decode 阶段取出来的数据是错误的。

冲突情况	具体指令	解决方案	如何阻塞	电路实现
正处于 execute 阶段的指令的前两条指令中存在写入寄存器的情况，且不是从内存中取出的值	addi、xori、ori、andi、lui 、jal 、add 、sub 、and、or 、xor、auipc	使用转发，将前两条指令即将写入寄存器的值直接替换 execute 阶段的操作数	无需阻塞	①将前两条指令要写入寄存器的地址与 execute 阶段两个操作数来源的寄存器地址比较， 若一致，②则进行转发，直接替换 execute 的操作数。（注意写入$0 寄存器的值不做转发）
正处于 execute 阶段的指令的前一条指令中存在从内存写入寄存器的情况。由于转发的数据源只能来自流水线寄存器的输出，需要等前一条指令到 writeback 阶段才能转发	ld	此时需要等一个时钟周期才能够获得转发回来的数据， 让 execute 阶段及之后的指令在原地停留一个时钟周期， 并使 memory 阶段执行的下一条指令为空指令	decode/execute 和 fetch/decode 寄存器阻塞 execute/memory 寄存器中插入气泡	①判断 memory 阶段的操作是否为 ld 操作， 若是②则将 memory 阶段写入寄存器的地址与 execute 阶段操作数来源的寄存器的地址进行比较， 若一致，③则插入气泡和阻塞
正处于 decode 阶段的指令需要从寄存器中取出值直接进行比较立即数加载等操作，此时其前一条指令的结果尚未从 alu 中算出	beq、jalr	此时前一条指令将要写入寄存器的值尚未从 alu 中运算得出，因此让 decode 阶段及其之后的指令在原地等待一个周期	decode/execute 寄存器插入气泡， fetch/decode 寄存器阻塞	①判断 decode 阶段指令是否为 beq 或 jalr ②判断 decode 取出的寄存器地址是否与 execute 即将写入寄存器的地址一致 若一致③则进行插入气泡和阻塞