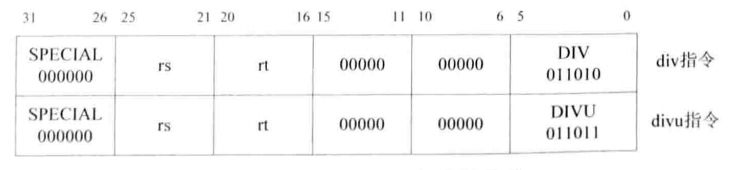
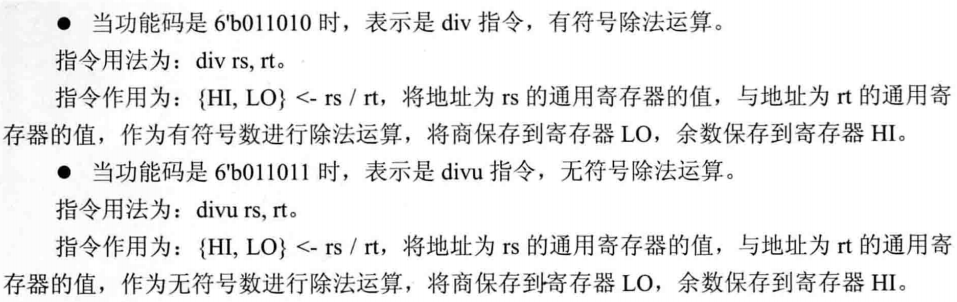
**除法指令的实现**

# 除法指令的说明





# 实现思路

## 2.1 试商法

试商法就是我们小学学习的简单除法，只不过这里全都换成了二进制，反而还简单了。在OpenMIPS中，对于32位除法，至少需要32个时钟周期才能得到结果，具体步骤见下：

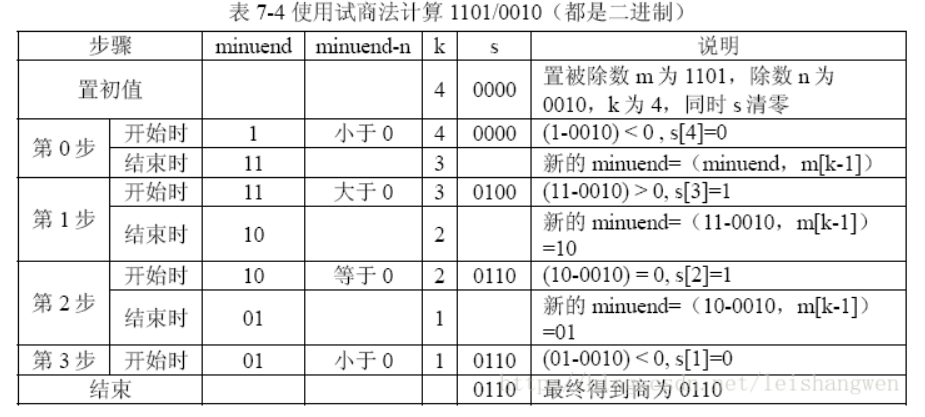
设被除数是m，除数是n，商保存在s中，被除数的位数是k，其计算步骤如下（为了便于说明，在此处将所有数据的最低位称为第1位，而不称为第0位）。

      1、取出被除数的最高位m[k]，使用被除数的最高位减去除数n，如果结果大于等于0，则商的s[k]为1，反之为0；

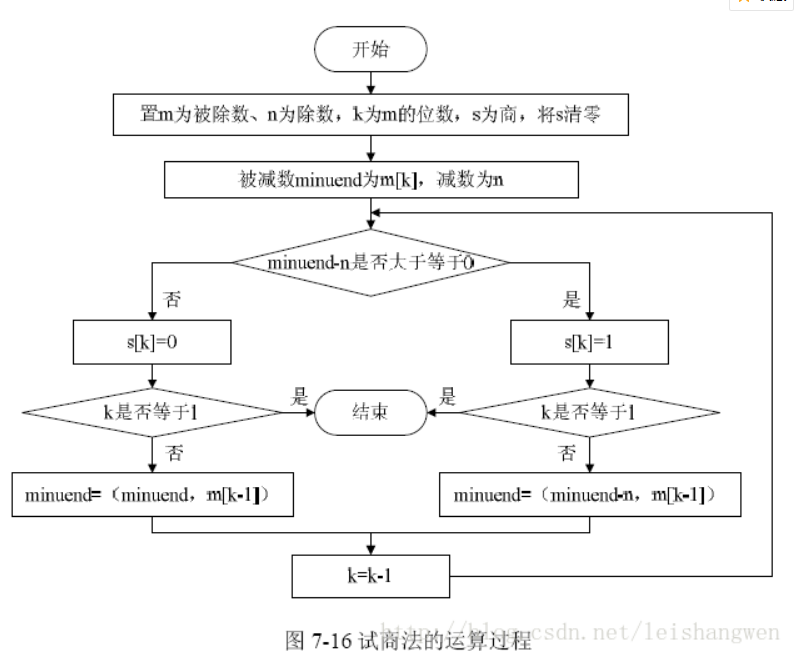
      2、如果上一步得出的结果是0，表示当前的被减数小于除数，则取出被除数剩下的值的最高位m[k-1]，与当前被减数组合做为下一轮的被减数；如果上一步得出的结果是1，表示当前的被减数大于除数，则利用上一步中减法的结果与被除数剩下的值的最高位m[k-1]组合做为下一轮的被减数。然后，设置k等于k-1；

      3、新的被减数减去除数，如果结果大于等于0，则商的s[k]为1，否则s[k]为0，后面的步骤重复2-3，直到k等于1。

举例见表：

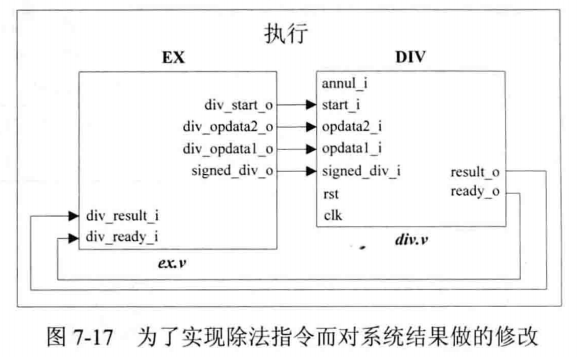


或者我们可以用一个流程图表示：



我们考虑新增加一个DIV模块，用来单独执行除法指令，在此之前要先执行流水线暂停。

## 2.2 系统结构修改



当程序执行到EX模块时，执行流水暂停，将寄存器中的被除数和除数的值传入DIV模块的opdata1\_i和opdata2\_i，用signed\_div\_i判断是否是有符号除法，除法指令通过start\_i进行控制。除法指令执行完后，通过ready\_o接口将状态传递到EX模块，并通过result\_o接口将最终结果传递回EX模块。其中result\_o高32位是余数，低32位是商，这些最终都将传递到HILO寄存器。

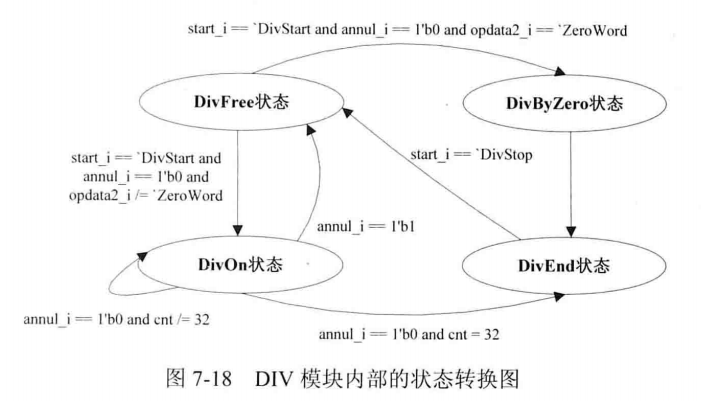
# 具体实现

## 3.1 增加DIV模块

接口描述如下表：



DIV模块的主要部分是一个状态机，共有四个状态，状态转换图如下：



复位的时候，DIV模块处于DivFree状态，当开始信号start\_i为DivStart时，并且输入信号annul\_i为0时，除法操作开始。

当除数为0时，状态机进入DivByZero状态，直接将除法结果的商和余数记为0，然后进入DivEnd状态，将ready\_o设置为DivResultReady，EX模块会设置Div模块的start\_i为DivStop，除法结束。

当除数不为0，进入DivOn状态，使用试商法，经过32个时钟周期得出结果，然后进入DivEnd状态，之后操作同上。

这个状态机有四个状态，而且DIV模块有开始和停止信号start\_i，这也是一个状态。但是这个状态和状态机的状态不一样。

关于这两种状态，后面在仿真时会具体说明。

## 3.2 修改ID模块

ID模块增加了两条除法指令，修改相应的端口，没什么难点。

## 3.3修改EX模块

增加的接口如下表：



该段代码分为三部分理解：

1. 当指令传过来时，并且DIV模块没有进行到第32个周期，也就是div\_ready\_i为DivResultNotReady，那么传递被除数和除数，div\_start\_o信号设置为Divstart，并且设置stall暂停信号，转向DIV模块进行运算。当DIV模块已经进行到第32周期时，即DIV模块中state为DivEnd状态，设置div\_ready\_i为DivResultReady，除法运算停止，撤掉暂停命令stallreq\_for\_div；
2. 暂停指令包括乘累加、乘累减和现在的除法，因此他们之间用逻辑或连接；
3. 最终div\_result\_i就是除法运算的结果，将高32位的余数存入HI寄存器，低32位的商存入LO寄存器。

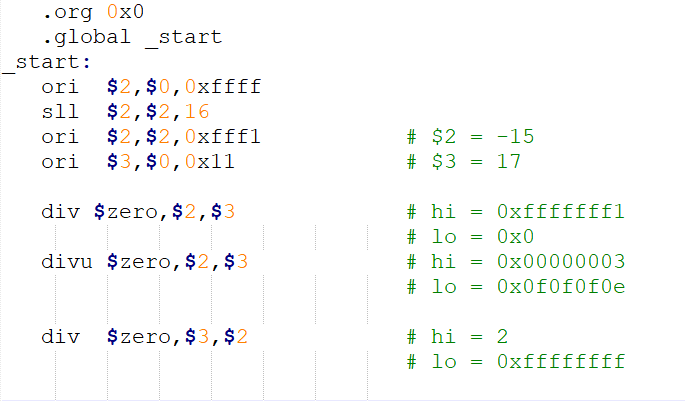
## 3.4 修改OpenMIPS模块

增加接口的例化，注意哪些值是相等的，很简单，细心即可。

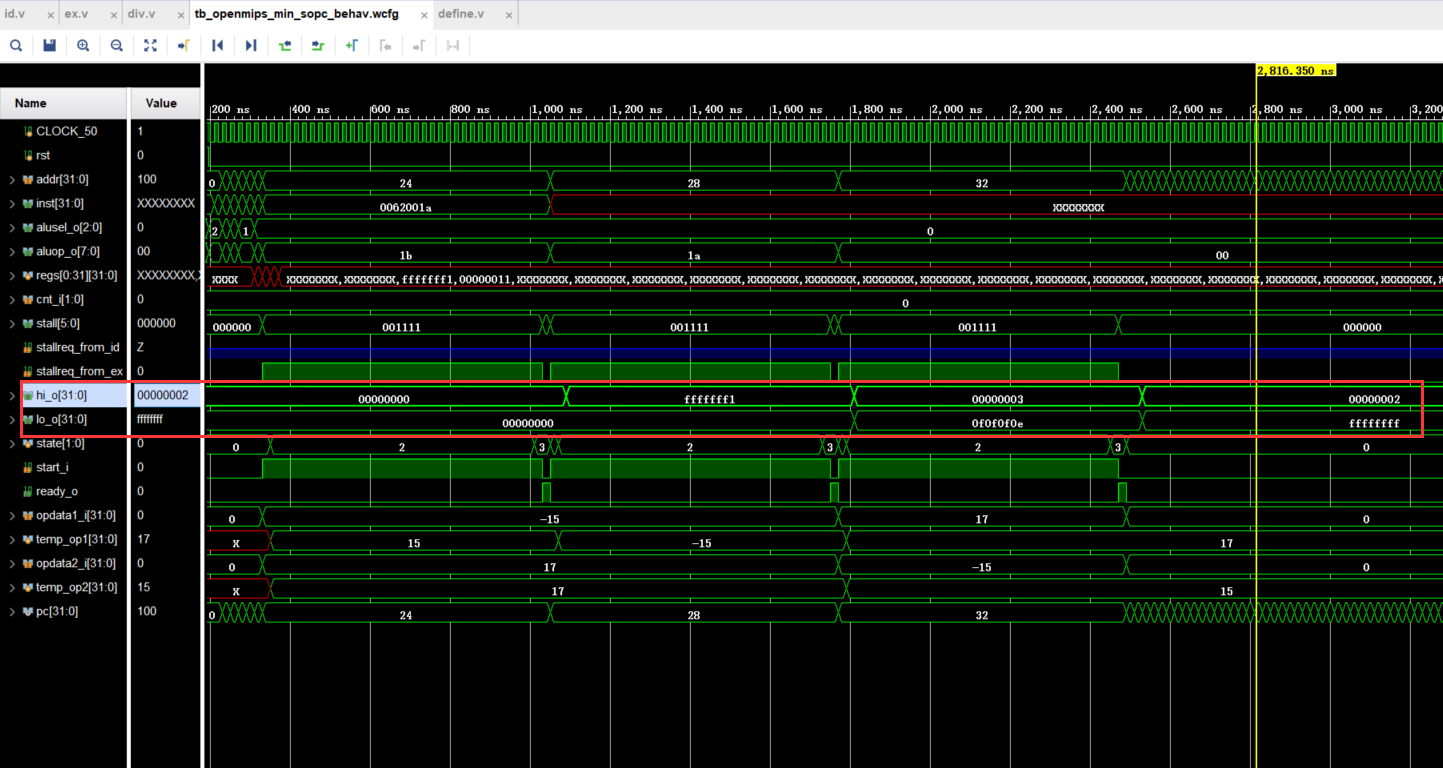
有一点需要注意，DIV模块的输入接口annul\_i目前为常数0，表示不会有取消除法指令的情况发生，但是在后面实现异常处理的时候，会重新确定DIV模块的输入接口annul\_i的值。

# 测试及分析

我使用了如下的汇编测试代码：

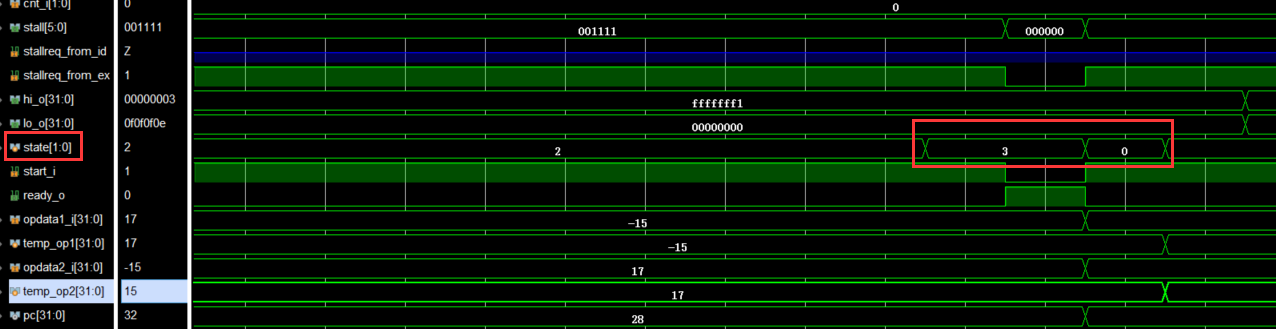


最终得到的仿真结果为：



结果正确。

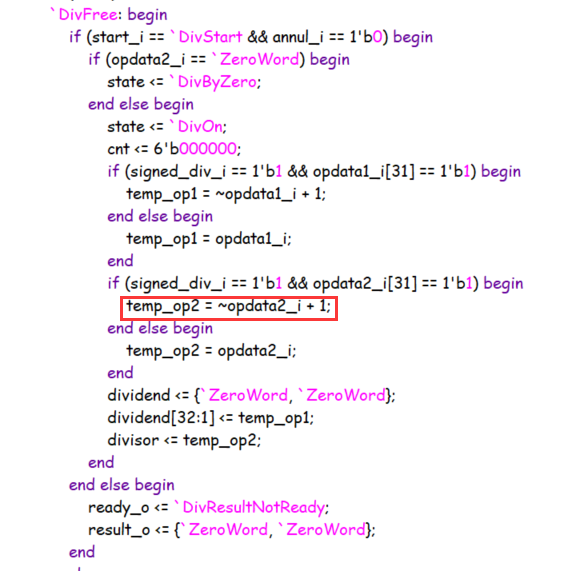
我们看一下state的值：



state为DivEnd状态是两个周期，而后面的DivFree状态只有一个周期，为什么呢？这就是和之前说的状态机的状态和Div模块的状态的区别了。

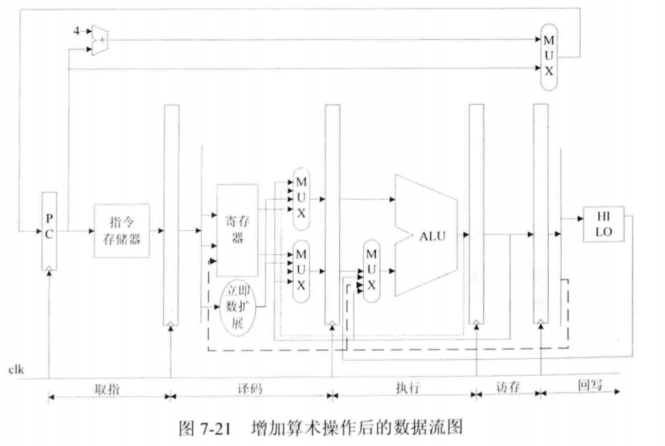
当32个周期执行完后，状态机状态从DivOn转换为DivEnd，但是Div模块还处于DivStart状态，因此还需要一个周期，通过EX模块将Div模块的状态变为DivStop才能彻底终止Div除法模块的运算，因此state会在DivEnd状态停留两个周期。

另外，我在写代码的时候，把红框框里的temo\_op2的值赋值成了opdata1\_i，但是只有最后一个结果的LO寄存器值不正确，找了很长时间才发现问题所在。



实际上就是测试数据比较特殊，正号负号的判断问题，有些数值没有进入我写错的那个判断语句。

在增加完算数操作后的数据流图如下：



# 一些思考

在Div模块中，我们定义了一个65位的dividend和33位的div\_temp，是因为一开始赋值的时候将被除数赋值给了dividend的[32:1]，因此第0位没有用到，所以最终结果也是中间空出了一位（32bit），这一位就是一开始的0bit，因此我有一个想法，若始终定义为64位和32位，而不是65位和33位，是否也可以实现呢？书上这样设计是否只是为了看上去结构更清晰还是有别的考虑？