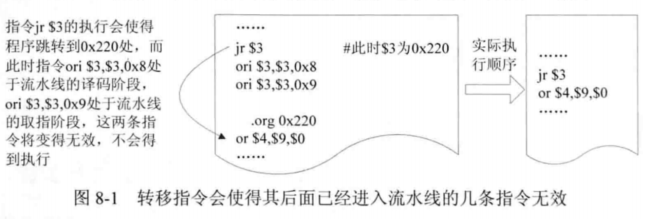
**转移指令的实现**

# 延迟槽的定义及说明

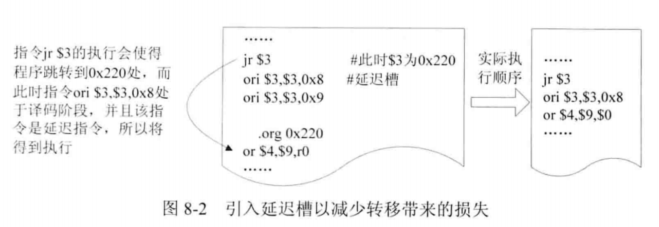
在之前设计流水线的时候，存在三种相关，分别为数据相关、结构相关、控制相关。其中控制相关是指流水线中的转移指令或者其他需要改写PC的指令造成的相关。这些指令改写了PC值，所以导致后面进入流水线的几条指令无效。比如：如果转移指令在流水线的执行阶段进行条件转移条件判断，在发生转移时，会导致当前处于取指、译码阶段的指令无效，需要重新取指。

如图：



尽管转移指令后面的指令确实不需要执行，但是这样却白白浪费了两个时钟周期，当转移指令很多的时候，会大大影响流水线性能。

因此我们引入延迟槽概念，规定转移指令后面的指令位置为“延迟槽”，延迟槽中的指令被称为“延迟槽指令”。该指令总是被执行，与是否被转移没有关系。引入延迟槽后的指令执行顺序如下图：



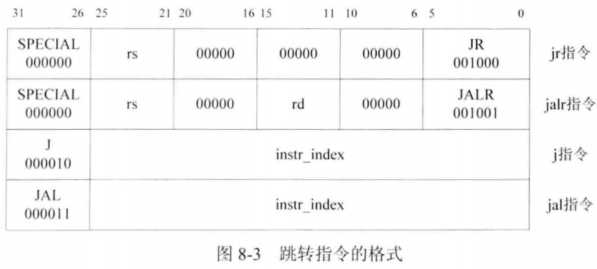
但是我们又发现一个问题，因为是在执行阶段开始判断转移，因此前面有两个阶段的指令已经被读取，使用延迟槽仍然会使得已经进入取指阶段的指令无效，也就是仍然浪费一个时钟周期。因此我们计划在译码阶段进行转移判断。

**那为什么不能用两个延迟槽呢？还有延迟槽的作用到底是什么呢？（我的疑问）**

# 指令说明

## 2.1 跳转指令

跳转指令格式如下：

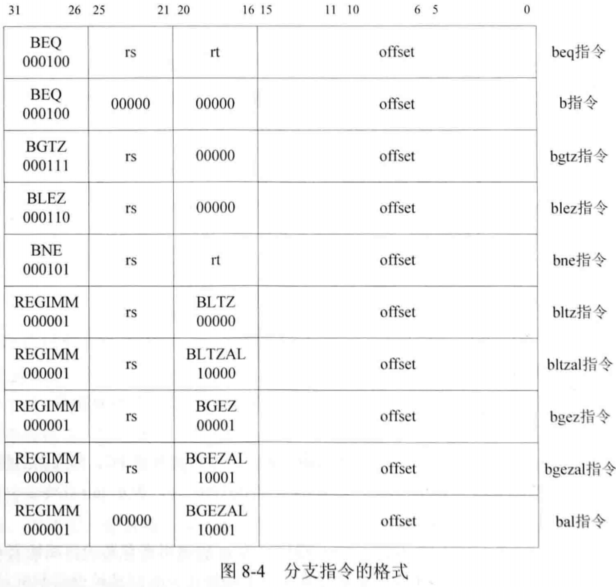


其中jalr指令表示格式：jalr rs或者jalr rd,rs 表示将地址为rs的通用寄存器的值赋给寄存器PC，作为新的指令地址，同时跳转到指令后面第二条指令的地址作为返回地址保存到地址为rd的通用寄存器，如果没有在指令中指明rd，那么默认将返回地址保存到寄存器$31.

J指令表示格式：j target，作用为：pc->（pc+4）[31, 28] || target || ‘00’，新指令地址的低28位是instr\_index左移两位的值，新指令地址的高4位是跳转指令后面延迟槽指令的地址高4位。

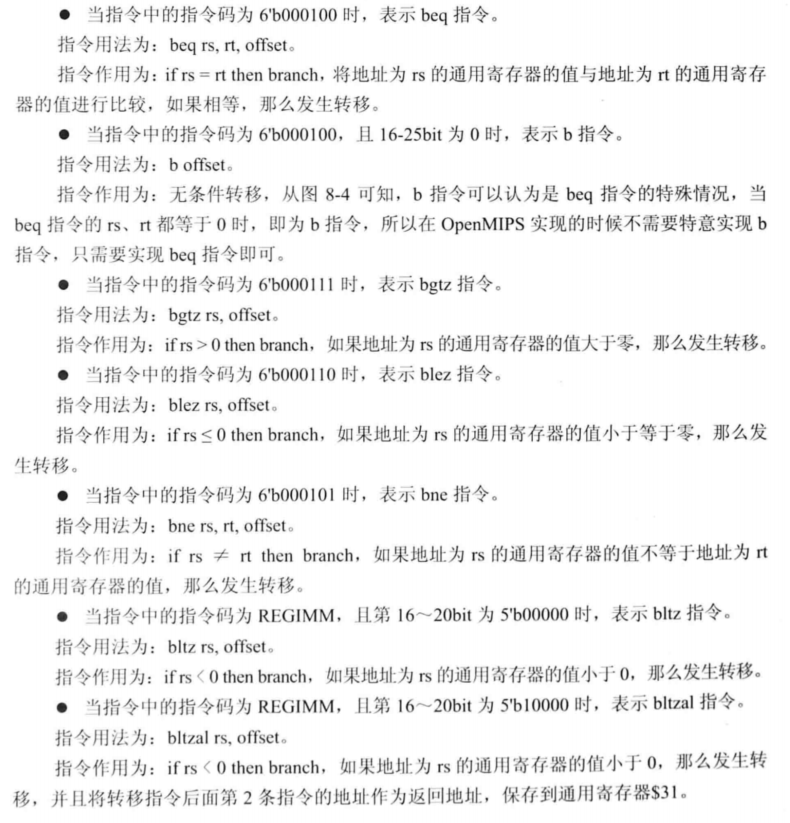
## 2.2 分支指令

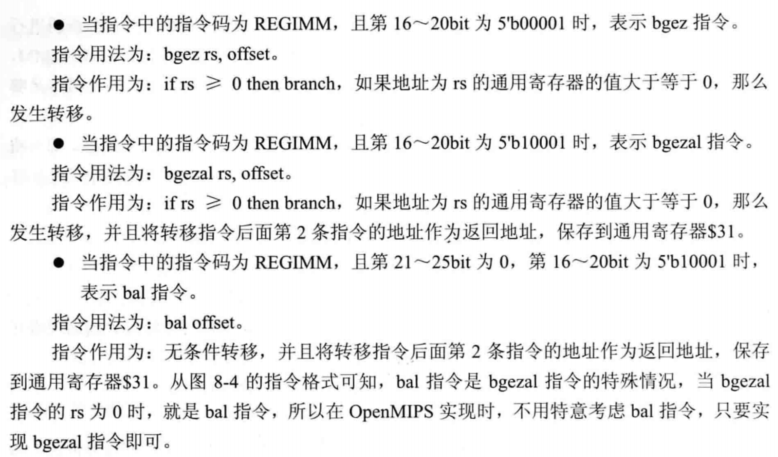
分支指令格式如下图：



从图中知道，前五条指令可以直接根据指令码进行判断，后五条指令指令码都是REGIMM，需要根据16-20bit进一步判断。并且所有分支指令的第0-15bit都是offset，如果发生转移将其左移两位再扩展成32位，然后与延迟槽指令的地址相加，得到的结果就是转移目的地址。







跳转指令和分支指令有点复杂，这也是汇编中子程序设计中比较复杂和难以理解的地方，因此我把书中的内容全部截图过来了。

值得注意的是，所有的分支指令在转移到目标地址前都要先执行延迟槽中的指令。

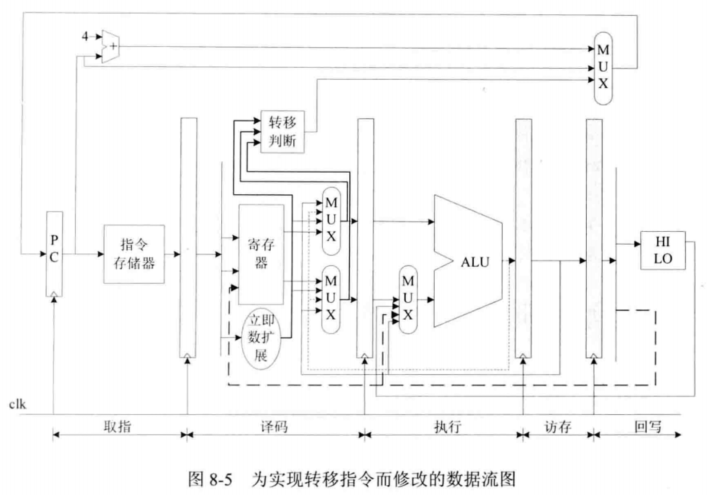
**现在我有个疑问，分支指令和转移指令有什么区别？例如b指令和j指令有什么区别？**

**还有为什么要保存转移指令后面第二条指令的地址？**

# 三、转移指令的实现

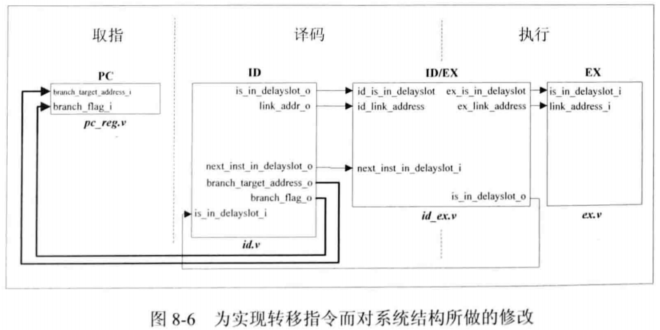
## 3.1 修改数据流图

我们将在译码阶段进行转移条件的判断，数据流图修改如下：



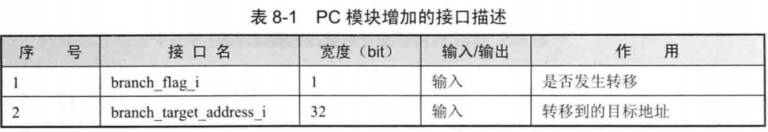
此时PC的取值有三种情况：PC+4，保持不变或者变成转移判断的结果。

## 3.2 系统结构的修改

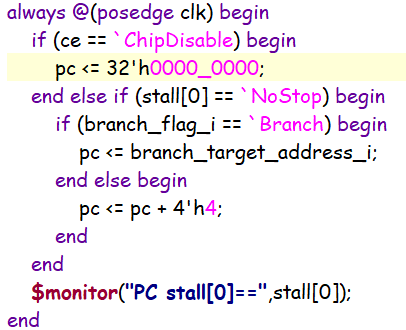


## 3.3 修改PC模块

增加的接口如下表：



代码如下：

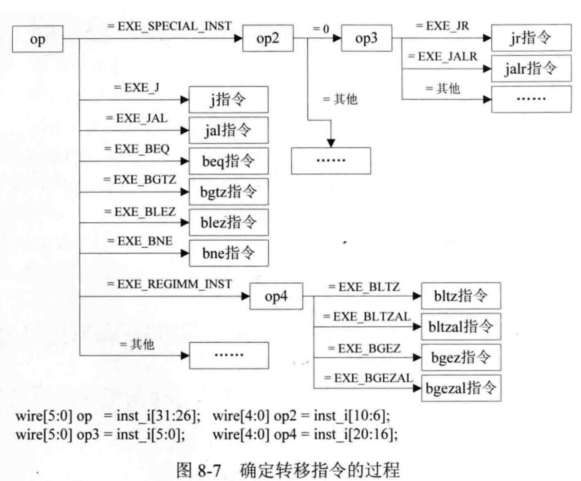


## 3.4 修改ID模块

增加的接口如下表所示：



确定指令转移的过程：

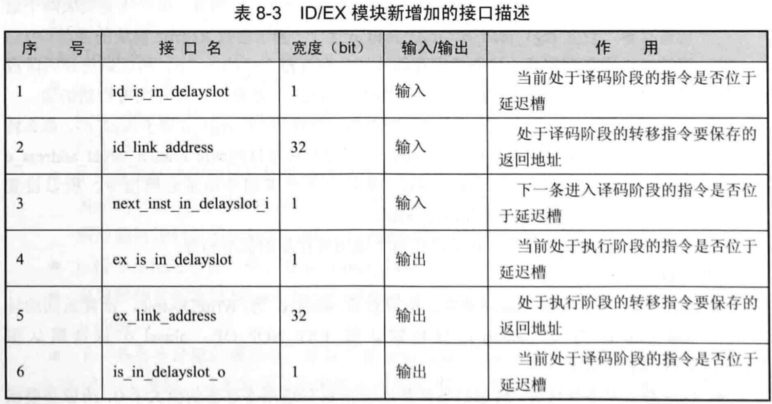


有几个问题：

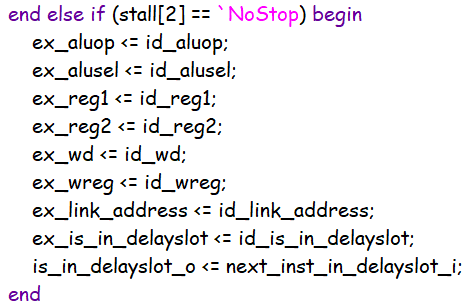
1. 为什么将BNE的类型赋值为`EXE\_BLEZ\_OP，类似的有BLTZ赋值为`EXE\_BGEZAL，BLTZAL赋值为`EXE\_BGEZAL等。实际上aluop这个运算子类型的值是为了在EX执行阶段方便进行运算而定义的，但是本章的跳转和分支无需运算，故这里的所有aluop都可以定义为`EXE\_NOP\_OP.
2. 转移目标地址怎么计算，为什么那样计算？
3. 保存的地址为什么是延迟槽之后第二条指令？这样有什么作用和意义？

## 3.5 修改ID/EX模块

增加的接口如下表所示：

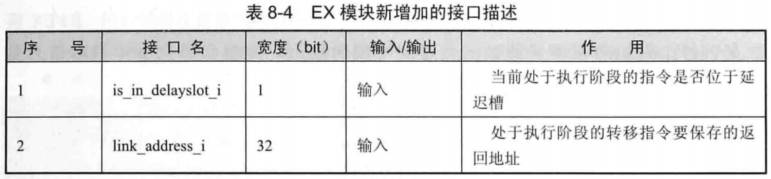


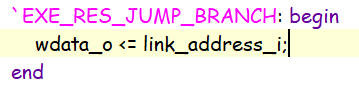
该部分代码很简单，当流水线译码阶段没有暂停时，ID/EX模块在时钟上升沿将新增加的输入传递到对应的输出。



## 3.6 修改EX模块

增加的接口如下表所示：





注意一点，此处没有用到输入的信号is\_in\_delayslot\_i，该信号表示当前处于执行阶段的指令是否为延迟槽指令，这个信号在异常处理中会使用，这里暂时不需要。

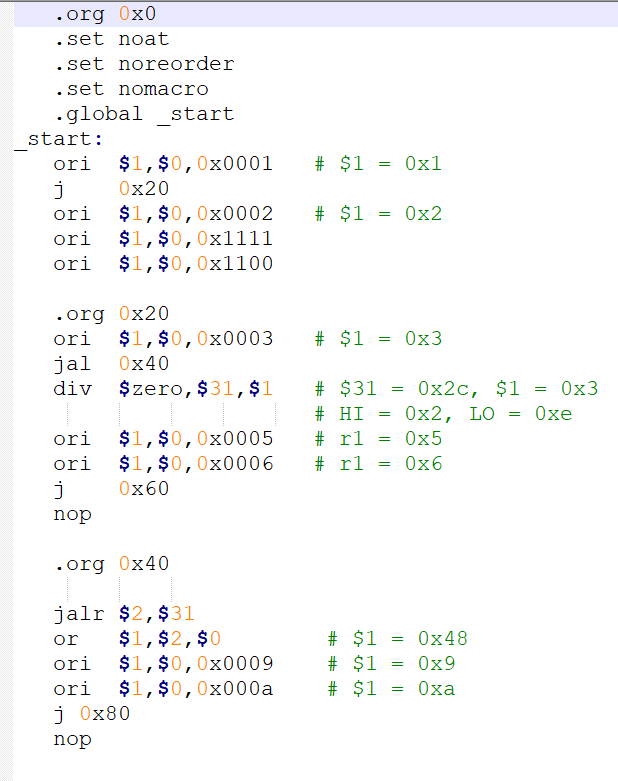
## 3.7 修改OpenMIPS模块

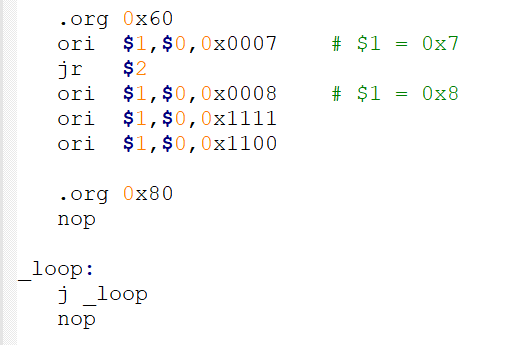
细心一点添加接口即可。

# 四、测试结果

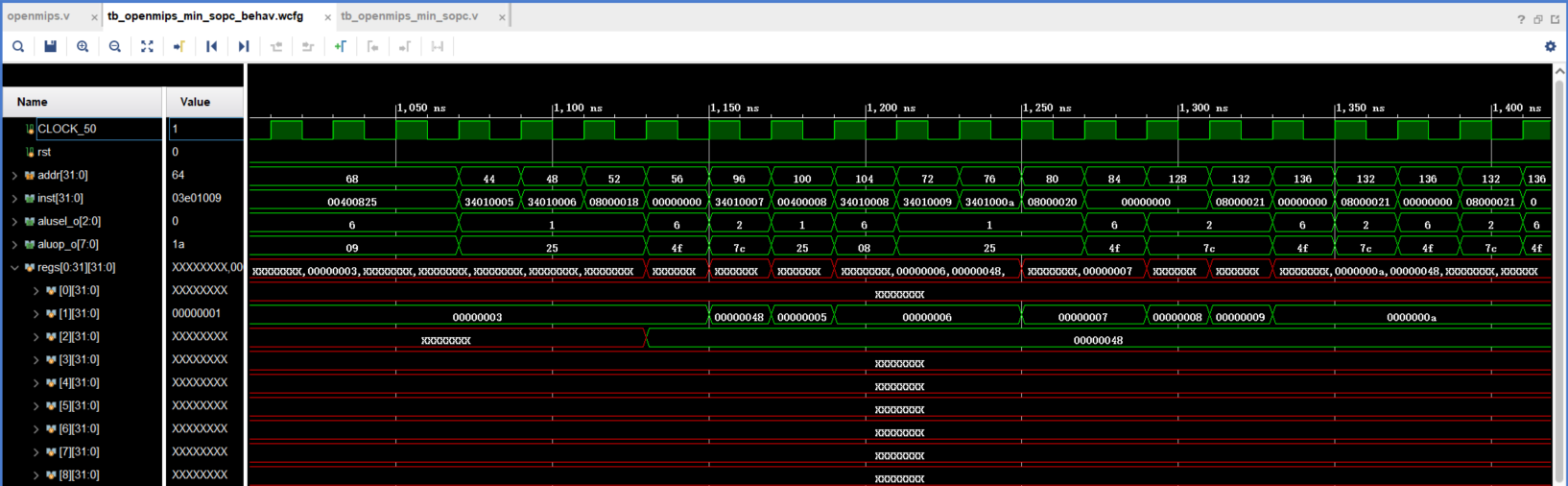
## 4.1 测试跳转指令

我们使用测试文件：



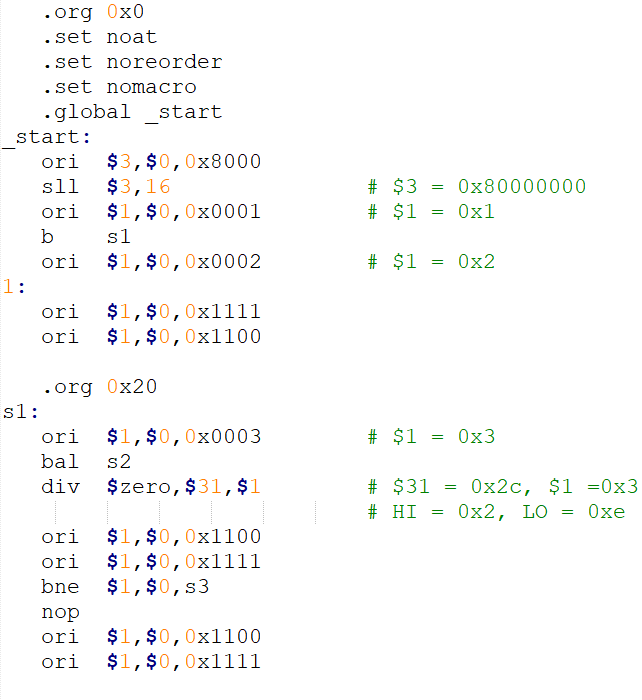
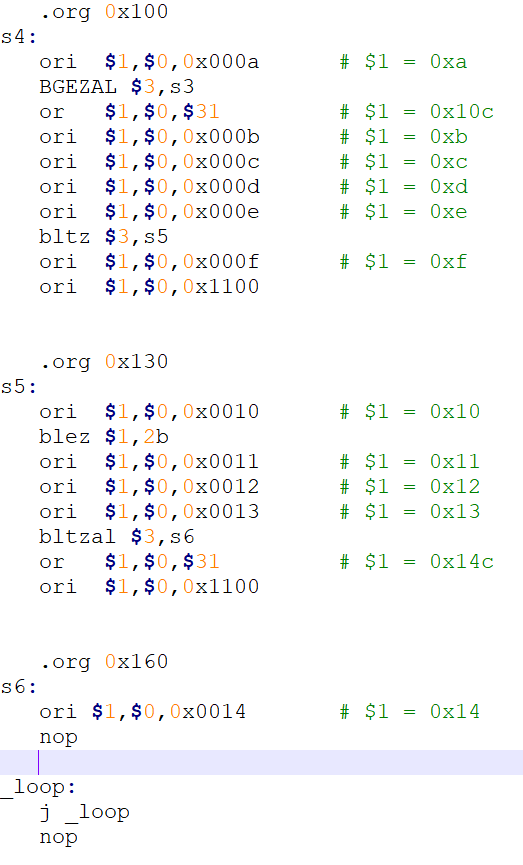


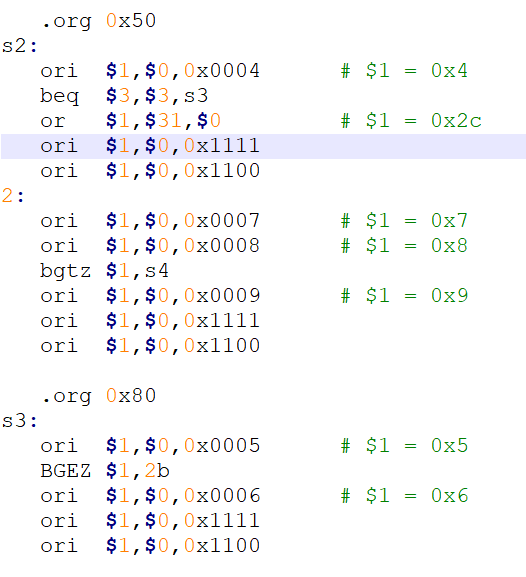
测试结果为，结果正确。



## 4.2 测试分支指令

使用测试文件：





测试结果如下，结果正确。

