（1）

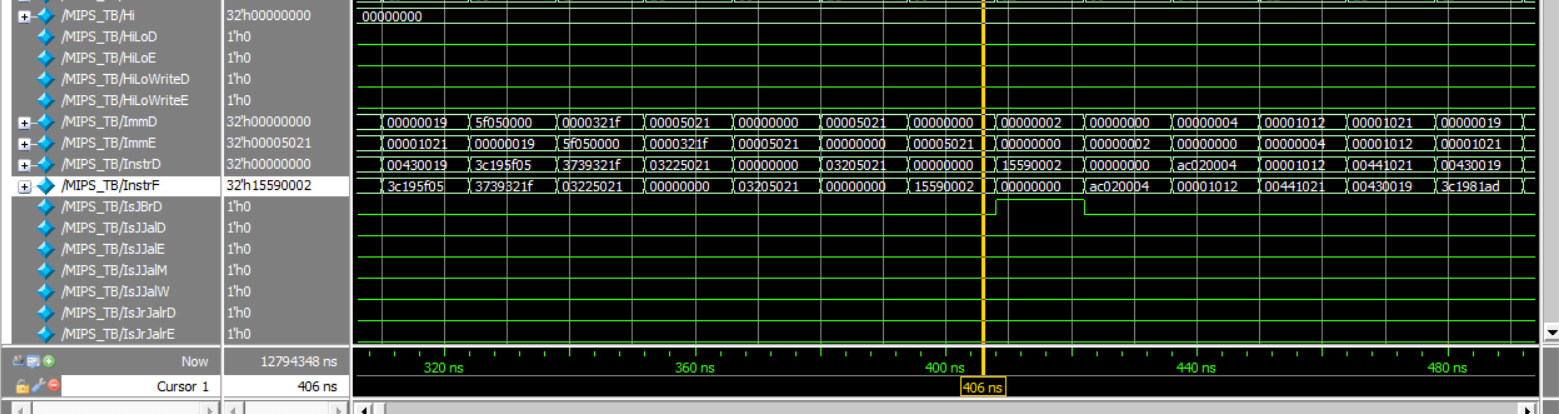


图 1 分支前-1

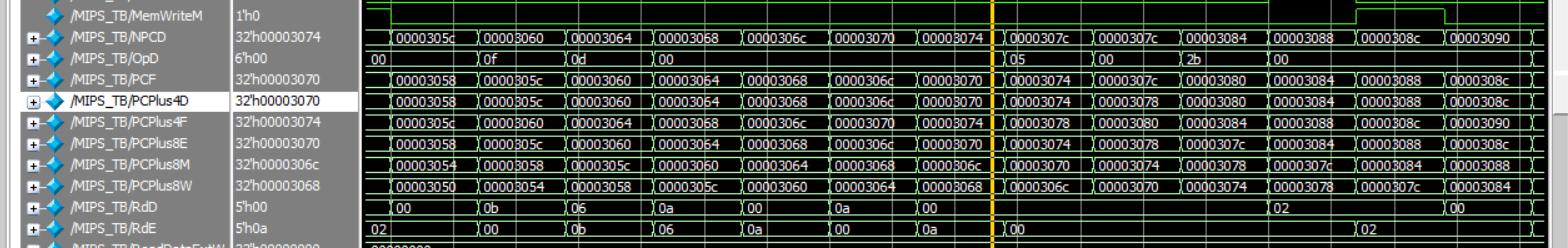


图 2 分支前-2

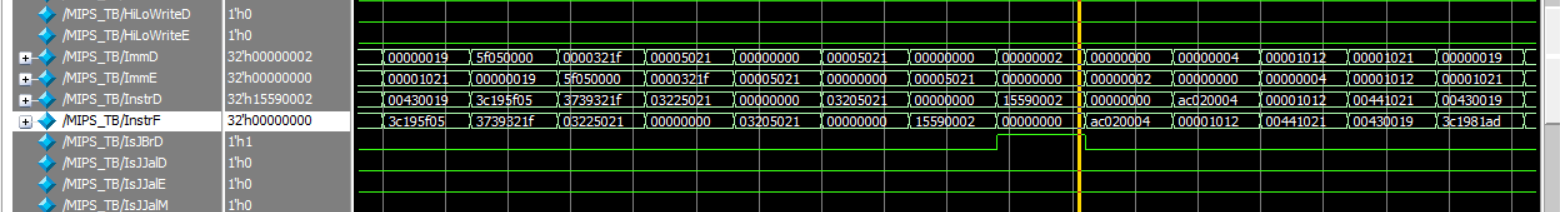


图 3 分支中-1

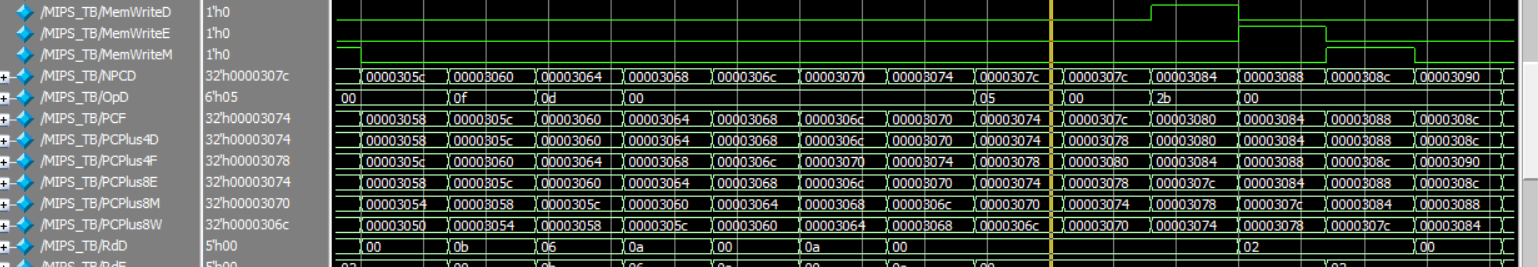


图 4 分支中-2

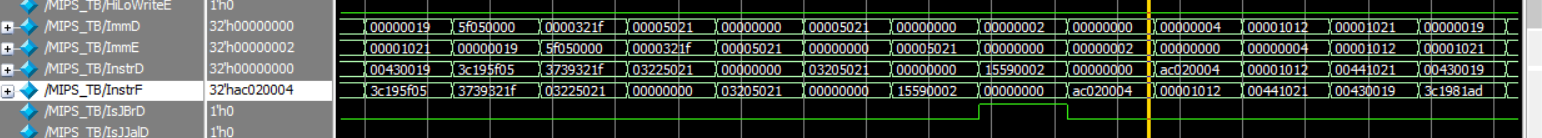


图 5 分支后-1

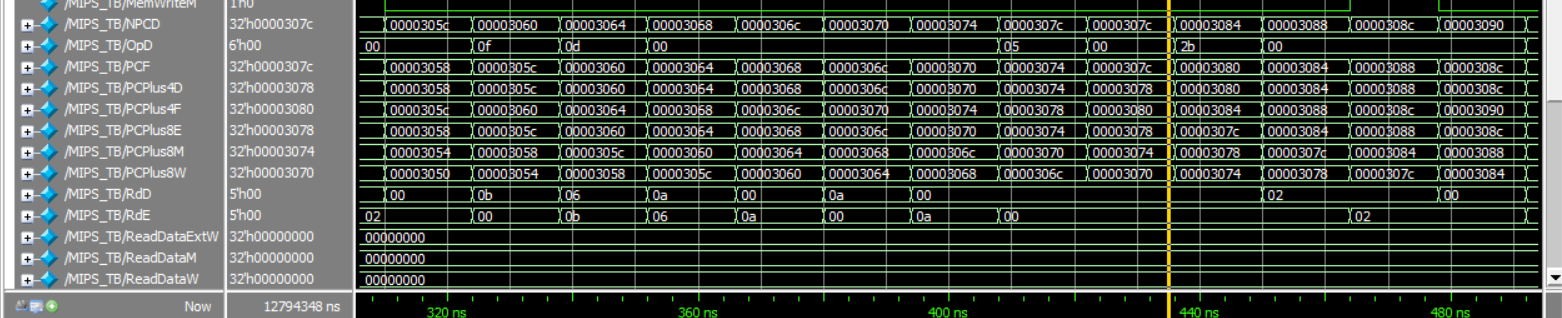


图 6分支后-2

（2）



图 7 分支前-1

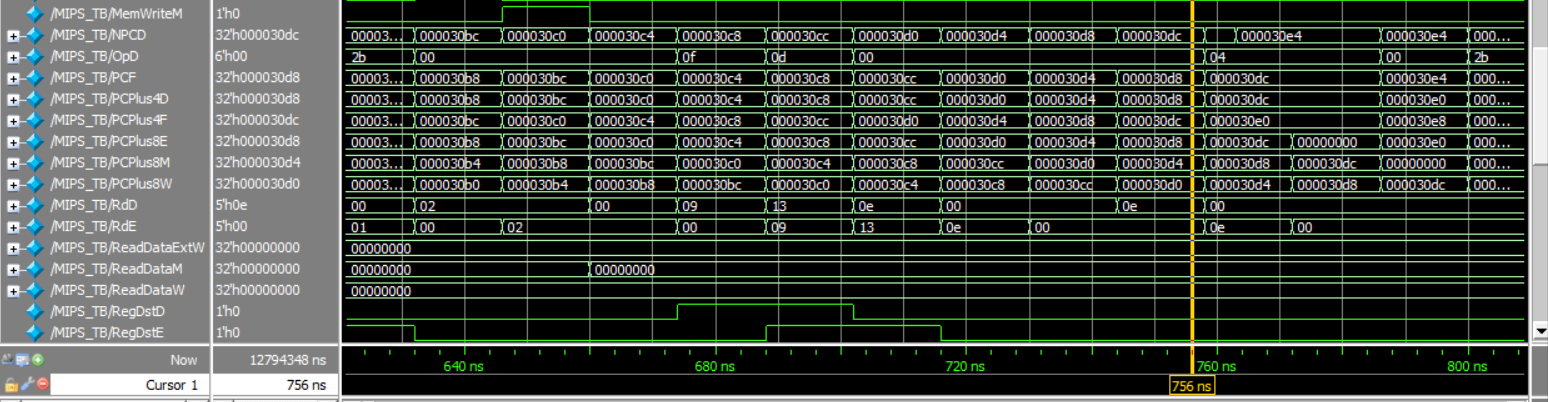


图 8 分支前-2

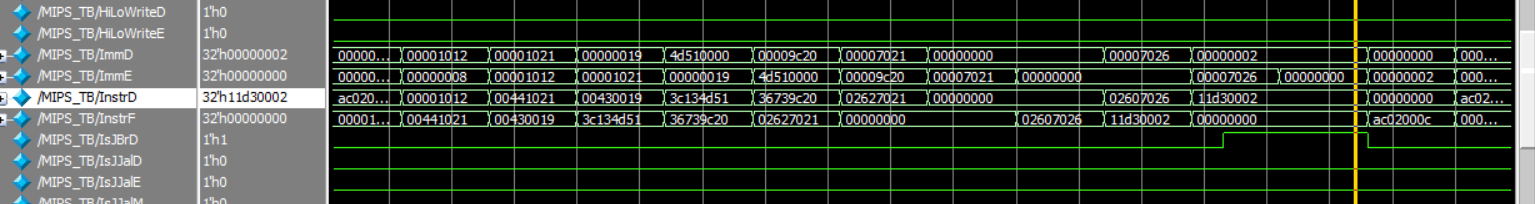


图 9 分支中-1

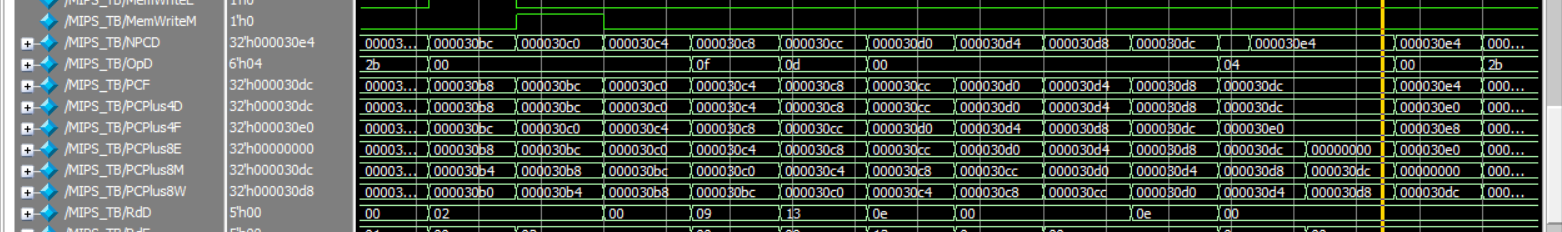


图 10 分支中-2

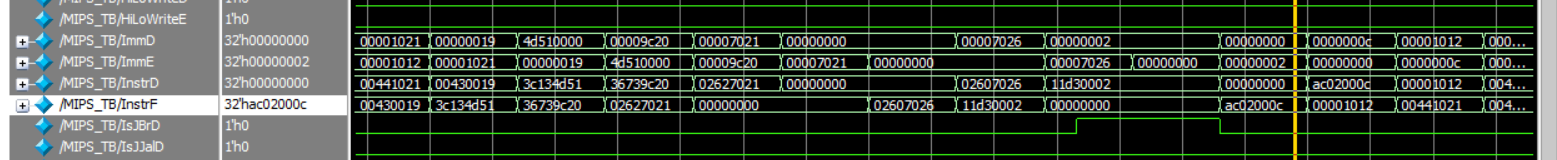


图 11 分支后-1



图 12 分支后-2

在EX阶段判决是否分支，因此会引起流水线暂停，损耗一个周期。

若在ID阶段已经判断是否分支，则不会流水线暂停。

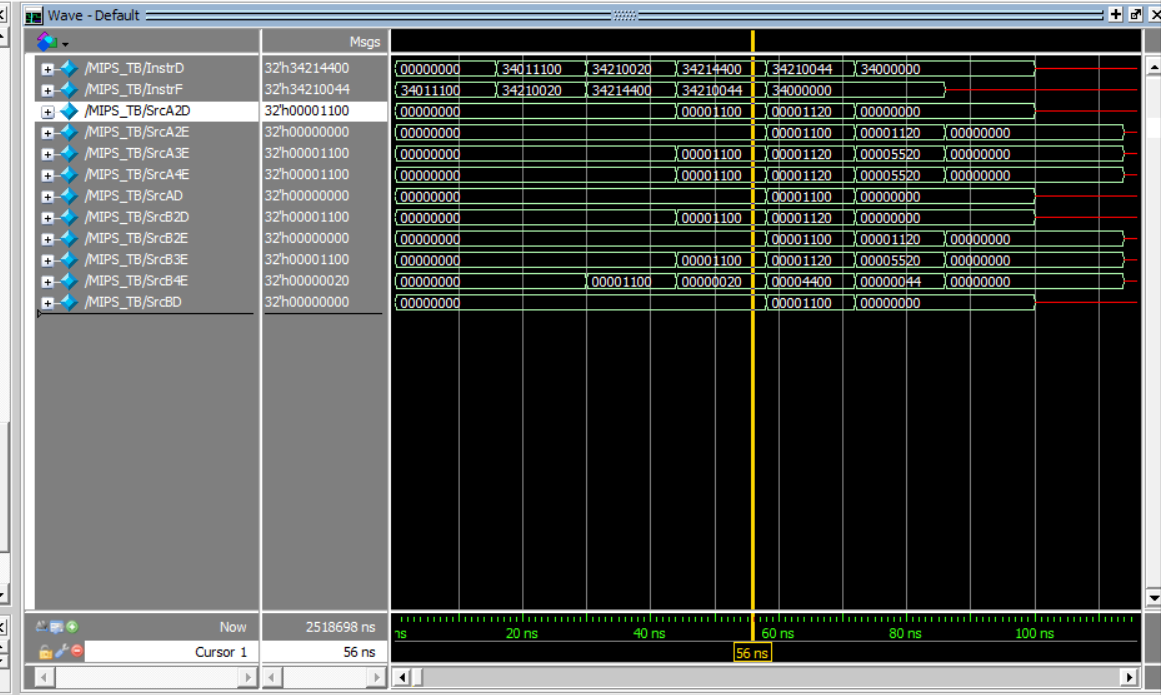


图 13 数据前推

.org 0x0

.global \_start

.set noat

\_start:

ori $1,$0,0x1100 # $1 = $0 | 0x1100 = 0x1100

ori $1,$1,0x0020 # $1 = $1 | 0x0020 = 0x1120

ori $1,$1,0x4400 # $1 = $1 | 0x4400 = 0x5520

ori $1,$1,0x0044 # $1 = $1 | 0x0044 = 0x5564

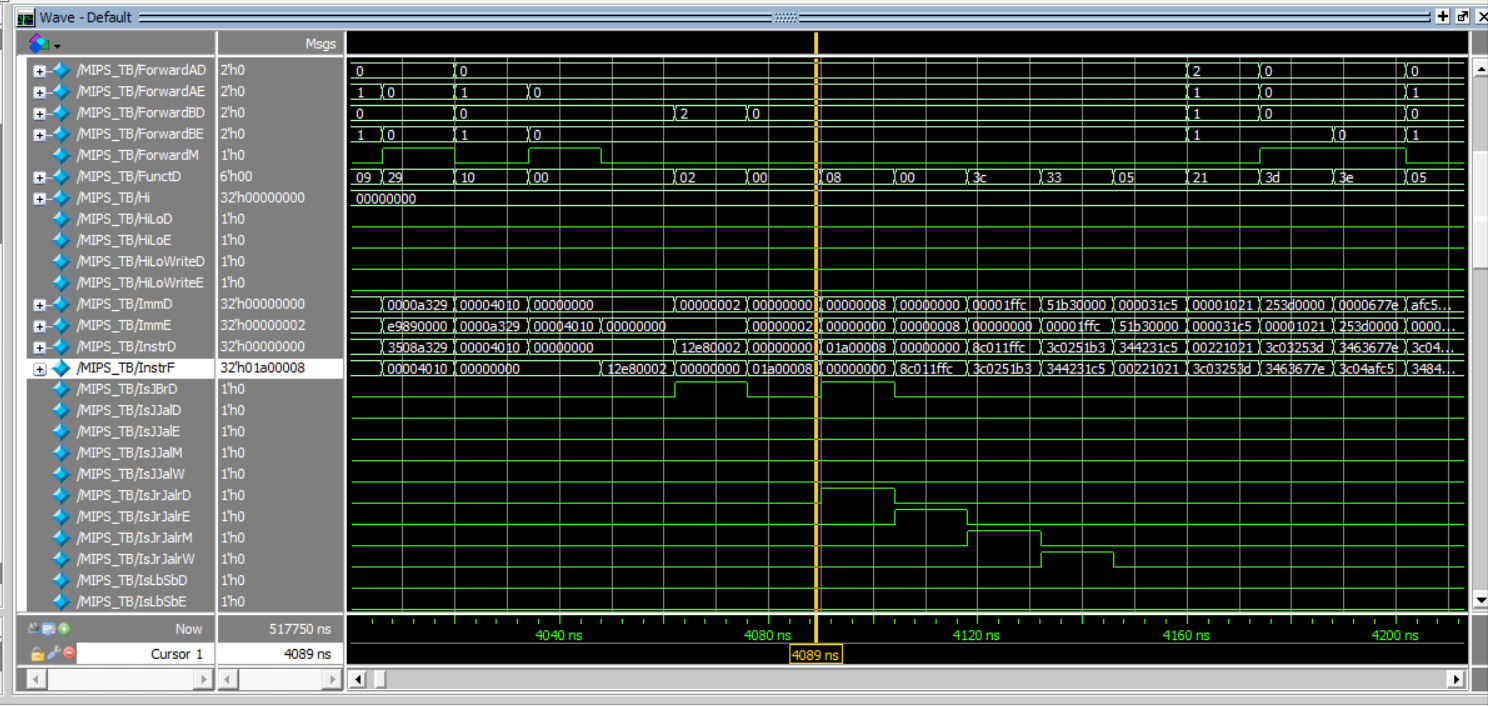


图 14 跳转前-1

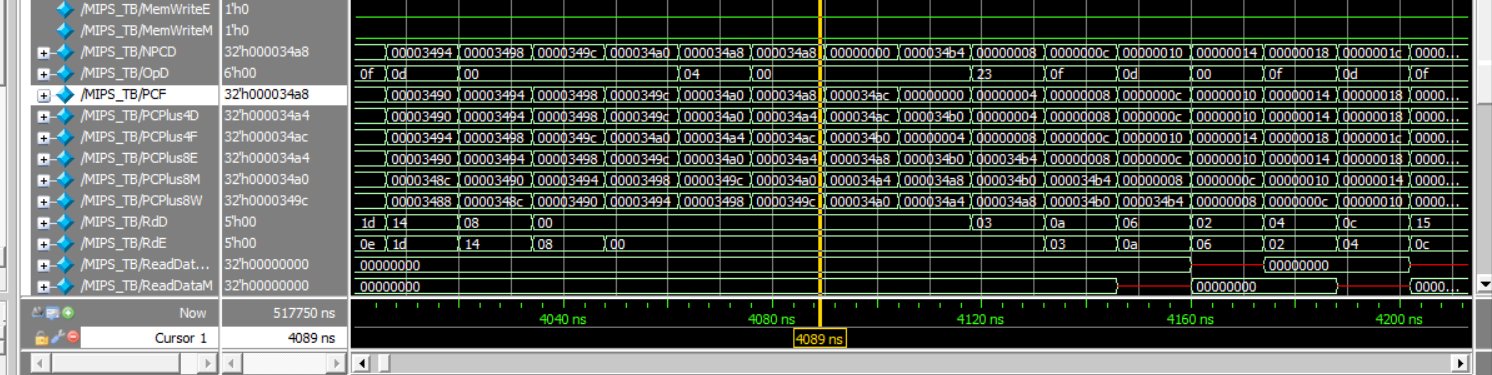


图 15 跳转前-2

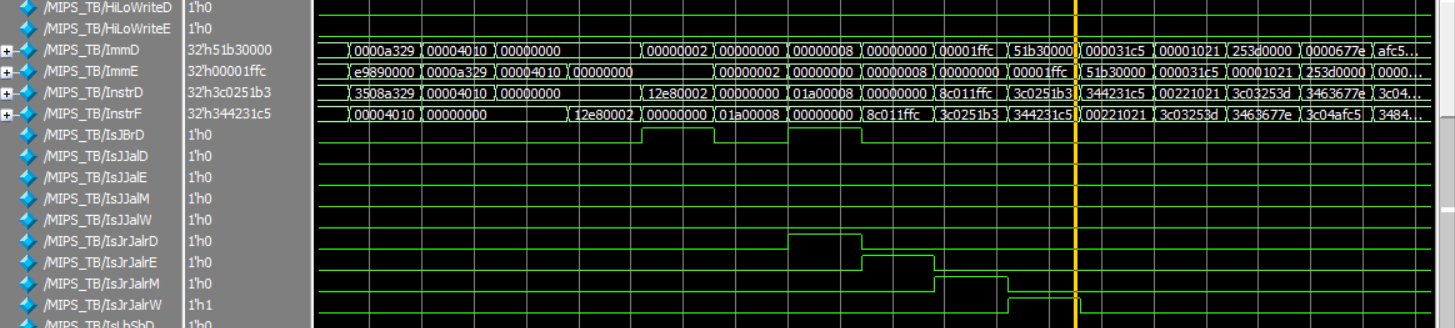


图 16 跳转后-1

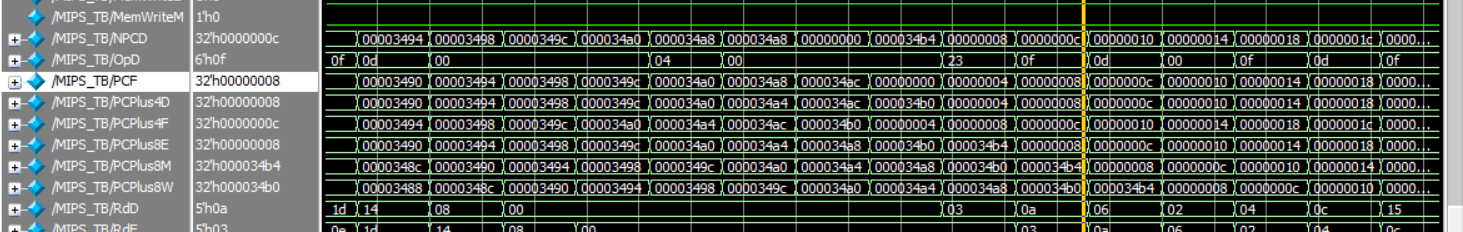
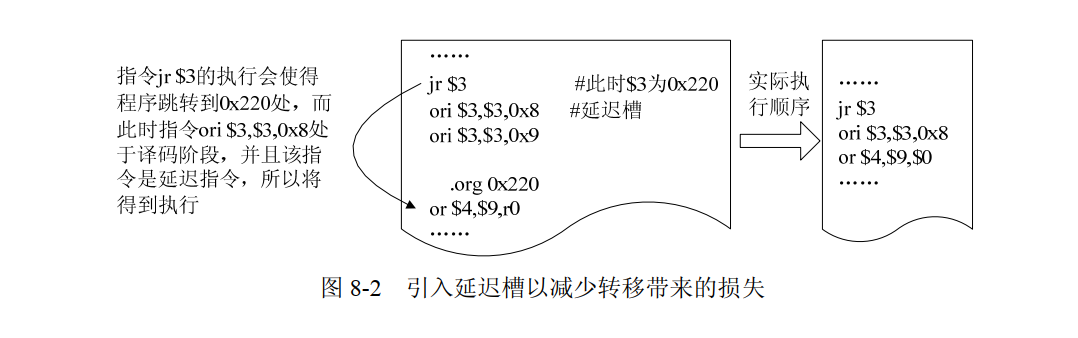


图 17 跳转后-2

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 的值，所以导致后面已经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。

也就是说，在流水线执行阶段进行转移判断，并且转移发生，那么会有 2 条无效指令，导致浪费了两个时钟周期。为了减少损失，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“延迟指令”（也可称之为“延迟槽指令”）。延迟指令总是被执行，与转移发生与否没有关系。



●取指：取出指令存储器中的指令， PC 值递增，准备取下一条指令。

●译码：对指令进行译码，依据译码结果，从 32 个通用寄存器中取出源操作数，有的指令要求两个源操作数都是寄存器的值，比如 or 指令，有的指令要求其中一个源操作数是指令中立即数的扩展，比如 ori 指令，所以这里有两个复用器，用于依据指令要求，确定参与运算的操作数，最终确定的两个操作数会送到执行阶段。

●执行阶段：依据译码阶段送入的源操作数、操作码，进行运算，对于 ori 指令而言，就是进行逻辑“或”运算，运算结果传递到访存阶段。

●访存阶段：对于 ori 指令，在访存阶段没有任何操作，直接将运算结果向下传递到回写阶段。

●回写阶段：将运算结果保存到目的寄存器。

**延迟槽：**

在流水线执行阶段进行转移判断，并且转移发生，那么会有 2 条无效指令，

导致浪费了两个时钟周期。为了减少损失，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“延迟指令”（也可称之为“延迟槽指令”）。延迟指令总是被执行，与转移发生与否没有关系。引入延迟槽后的指令执行顺序如图 8-2 所示。 OpenMIPS 处理器就计划使用延迟槽技术。

但是，即使引入延迟槽，在转移发生时仍然会导致已经进入取指阶段的指令无效，也就是说，仍浪费一个时钟周期，要解决这个问题，可以在译码阶段进行转移判断，这样就可以避免浪费时钟周期。

即使通过数据前推的方法，将访存阶段加载得到的数据前推，也解决不了问题，因为数据加载时， beq 指令已经处于执行阶段了，已经进行了比较判断，这种情况称为 load 相关。

**1．加载指令实现思路**

加载指令在译码阶段进行译码，得到运算类型 alusel\_o、 aluop\_o，以及要写的目的寄存器信息。这些信息传递到执行阶段，然后又传递到访存阶段，访存阶段依据这些信息，设置对数据存储器 RAM 的访问信号。从 RAM 读取回来的数据需要按照加载指令的类型、加载地址进行对齐调整，调整后的结果作为最终要写入目的寄存器的数据。

**2．存储指令实现思路**

存储指令在译码阶段进行译码，得到运算类型 alusel\_o、 aluop\_o，以及要存储的数据。这些信息传递到执行阶段，然后又传递到访存阶段，访存阶段依据这些信息，设置对数据存储器 RAM 的访问信号，将数据写入 RAM。