跳转指令与分支指令。《自己动手学 CPU》第八章内容，延迟槽的概念、分支指令、跳转指令的解释，具体实现思路。

8.1 延迟槽

在实现转移指令之前，先介绍一下延迟槽的概念。在第 5 章已经介绍了流水线中存在的三种相关：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 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。如图 8-1 所示。

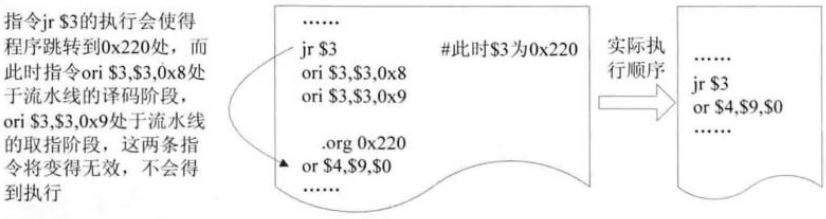

图 8-1 转移指令会使得其后面已经进入流水线的几条指令无效

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。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。OpenMIPS 处理器就设计为在译码阶段进行转移判断。

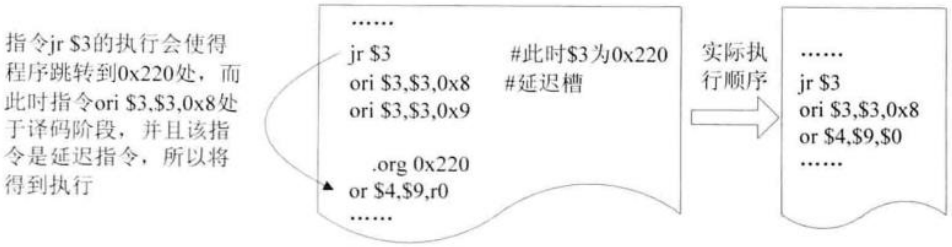

图 8-2 引入延迟槽以减少转移带来的损失

8.2 转移指令说明

MIPS32 指令集架构中定义的转移指令共有 14 条，可分为如下两类。

- 跳转指令： jr 、jalr 、j 、jal。
- 分支指令：b、bal、beq、bgez、bgezal、bgtz、blez、bltz、bltzal、bne。

其中，跳转指令是绝对转移，分支指令是相对转移。本节分别介绍这两类指令。

1. 跳转指令

跳转指令的格式如图 8-3 所示。

31 26	25 21	20 16	15 11	10 6	5 0	
SPECIAL 000000	rs	00000	00000	00000	JR 001000	jr指令
SPECIAL 000000	rs	00000	rd	00000	JALR 001001	jalr指令
J 000010	instr_index					j指令
JAL 000011	instr_index					jal指令

图 8-3 跳转指令的格式

从图 8-3 可知，j、jal 指令可以通过指令码进行判断，jr、jalr 指令的指令码为 SPECIAL，还需要依据功能码进一步判断。

- 当指令中的指令码为 SPECIAL，功能码为 6'b001000 时，表示 jr 指令。

指令用法为：jr rs。

指令作用为：pc <- rs，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。

- 当指令中的指令码为 SPECIAL，功能码为 6'b001001 时，表示 jalr 指令。

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。

- 当指令中的指令码为 6'b000010 时，表示 j 指令。

指令用法为：j target。

指令作用为：pc <- (pc+4)[31,28] || target || '00'，转移到新的指令地址，其中新指令地址的低 28 位是指令中的 target（也就是图 8-3 中的 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。

- 当指令中的指令码为 6'b000011 时，表示 jal 指令。

指令用法为：jal target。

指令作用为：pc <- (pc+4)[31,28] || target || '00'，转移到新的指令地址，新指令地址与指令 j 相同，不再解释。但是，指令 jal 还要将跳转指令后面第 2 条指令的地址作为返回地址保存到寄存器$31。

j、jal、jr、jalr 指令在转移之前都要先执行延迟槽指令。

2. 分支指令

分支指令的格式如图 8-4 所示。

31 26	25 21	20 16	15 11 10 6	5 0	
BEQ 000100	rs	rt	offset		beq指令
BEQ 000100	00000	00000	offset		b指令
BGTZ 000111	rs	00000	offset		bgtz指令
BLEZ 000110	rs	00000	offset		blez指令
BNE 000101	rs	rt	offset		bne指令
REGIMM 000001	rs	BLTZ 00000	offset		bltz指令
REGIMM 000001	rs	BLTZAL 10000	offset		bltzal指令
REGIMM 000001	rs	BGEZ 00001	offset		bgez指令
REGIMM 000001	rs	BGEZAL 10001	offset		bgezal指令
REGIMM 000001	00000	BGEZAL 10001	offset		bal指令

图 8-4　分支指令的格式

8.3 转移指令实现思路

8.3.1 实现思路

根据 8.1 节的论述，为了尽量减少转移指令带来的损失，OpenMIPS 在译码阶段进行转移条件的判断，如果满足转移条件，那么修改 PC 为转移目标地址。

8.3.2 数据流图的修改

为了实现转移指令，修改数据流图如图 8-5 所示。

从图 8-5 中可知，在译码阶段多了转移判断的步骤，此外，PC 的取值变为三种情况。

情况一：PC 等于 PC+4。这属于一般情况，每个时钟周期 PC 加 4，指向下一条指令。

情况二：PC 保持不变。当流水线暂停的时候，就会发生这种情况，参考第 7 章中流水线暂停的实现。

情况三：PC 等于转移判断的结果。如果是转移指令，且满足转移条件，那么会将转移目标地址赋给 PC。

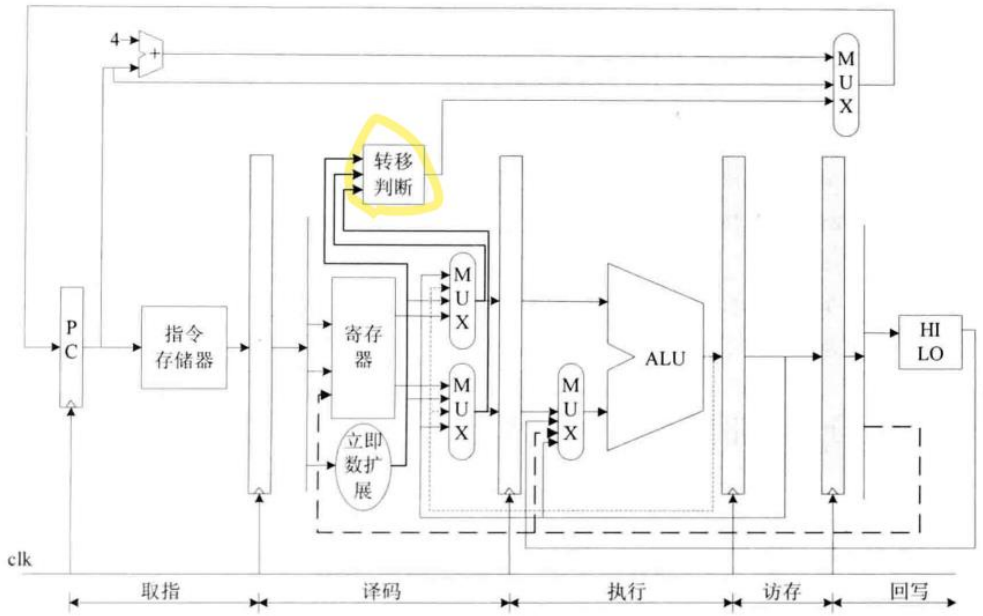

图 8-5 为实现转移指令而修改的数据流图

8.3.3 系统结构的修改

为了实现转移指令，需要对系统结构进行修改，增加部分模块的接口，主要修改如图 8-6 所示。

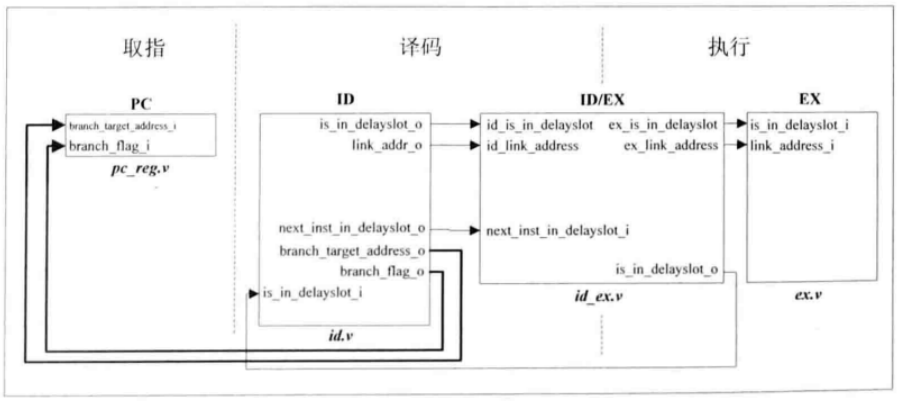

图 8-6 为实现转移指令而对系统结构所做的修改

有以下几点说明。

（1）如果处于译码阶段的指令是转移指令，并且满足转移条件，那么 ID 模块设置转移发

生标志 branch_flag_o 为 Branch，同时通过 branch_target_address_o 接口给出转移目的地址，送到 PC 模块，后者据此修改取指地址。

（2）如果处于译码阶段的指令是转移指令，并且满足转移条件，那么 ID 模块还会设置 next_inst_in_delayslot_o 为 InDelaySlot，表示下一条指令是延迟槽指令，其中 InDelaySlot 是一个宏定义。next_inst_in_delayslot_o 信号会送入 ID/EX 模块，并在下一个时钟周期通过 ID/EX 模块的 is_in_delayslot_o 接口送回到 ID 模块，ID 模块可以据此判断当前处于译码阶段的指令是否是延迟槽指令。

（3）如果转移指令需要保存返回地址，那么 ID 模块还要计算返回地址，并通过 link_addr_o 接口输出，该值最终会传递到 EX 模块，作为要写入目的寄存器的值。