指令的地址。对于 call 和 jmp(无条件转移)来说，下一条指令的地址是指令中的常数字 valC，而对于其他指令来说就是 valP。因此，通过预测 PC 的下一个值，在大多数情况下，我们能达到每个时钟周期发射一条新指令的目的。对大多数指令类型来说，我们的预测是完全可靠的。对条件转移来说，我们既可以预测选择了分支，那么新 PC 值应为 valC，也可以预测没有选择分支，那么新 PC 值应为 valP。无论哪种情况，我们都必须以某种方式来处理预测错误的情况，因为此时已经取出并部分执行了错误的指令。我们会在 4.5.8 节中再讨论这个问题。

猜测分支方向并根据猜测开始取指的技术称为分支预测。实际上所有的处理器都采用了某种形式的此类技术。对于预测是否选择分支的有效策略已经进行了广泛的研究[46，2.3 节]。有的系统花费了大量硬件来解决这个任务。我们的设计只使用了简单的策略，即总是预测选择了条件分支，因而预测 PC 的新值为 valC。

旁注　其他的分支预测策略

我们的设计使用总是选择(always taken)分支的预测策略。研究表明这个策略的成功率大约为 60%[44，122]。相反，从不选择(never taken，NT)策略的成功率大约为 40%。稍微复杂一点的是反向选择、正向不选择(backward taken，forward not-taken，BTFNT)的策略，当分支地址比下一条地址低时就预测选择分支，而分支地址比较高时，就预测不选择分支。这种策略的成功率大约为 65%。这种改进源自一个事实，即循环是由后向分支结束的，而循环通常会执行多次。前向分支用于条件操作，而这种选择的可能性较小。在家庭作业 4.55 和 4.56 中，你可以修改 Y86-64 流水线处理器来实现 NT 和 BTFNT 分支预测策略。

正如我们在 3.6.6 节中看到的，分支预测错误会极大地降低程序的性能，因此这就促使我们在可能的时候，要使用条件数据传送而不是条件控制转移。

我们还没有讨论预测 ret 指令的新 PC 值。同条件转移不同，此时可能的返回值几乎是无限的，因为返回地址是位于栈顶的字，其内容可以是任意的。在设计中，我们不会试图对返回地址做任何预测。只是简单地暂停处理新指令，直到 ret 指令通过写回阶段。在 4.5.8 节中，我们将回过来讨论这部分的实现。

旁注　使用栈的返回地址预测

对大多数程序来说，预测返回值很容易，因为过程调用和返回是成对出现的。大多数函数调用，会返回到调用后的那条指令。高性能处理器中运用了这个属性，在取指单元中放入一个硬件栈，保存过程调用指令产生的返回地址。每次执行过程调用指令时，都将其返回地址压入栈中。当取出一个返回指令时，就从这个栈中弹出顶部的值，作为预测的返回值。同分支预测一样，在预测错误时必须提供一个恢复机制，因为还是有调用和返回不匹配的时候。通常，这种预测很可靠。这个硬件栈对程序员来说是不可见的。

PIPE－的取指阶段，如图 4-41 底部所示，负责预测 PC 的下一个值，以及为取指选择实际的 PC。我们可以看到，标号为“Predict PC”的块会从 PC 增加器计算出的 valP 和取出的指令中得到的 valC 中进行选择。这个值存放在流水线寄存器 F 中，作为程序计数器的预测值。标号为“Select PC”的块类似于 SEQ＋的 PC 选择阶段中标号为“PC”的块(图 4-40)。它从三个值中选择一个作为指令内存的地址：预测的 PC，对于到达流水线