三、目标地址预测

1、直接跳转类型的分支预测

跳转地址是固定的，只用考虑如何存储的问题

BTB(BranchTargetBuffer) 跳转地址Cache

- 组相连：选择合适的组数
- 直接映射：只取tag的一部分作为tag

当BTB缺失时，也有两种考虑：一种是停止取指直到目标跳转地址被计算出来，这可能导致产生较多的气泡；另一种是顺序取指，如果预测发生了跳转，则需要清洗流水线，功耗大。

2.间接跳转类型

①Call/Return类型：Call指令地址的目标永远是固定的，因此可以用BTB来处理；Return指令的目标地址一定是Call指令的下一条，因此可以用一个栈来专门存放返回地址→RAS(ReturnAddressStack)

希望能在分支预测阶段就知道指令是否为Call/Return类型，因此可以在BTB中增加一项，用来标记分支指令的类型。

RAS大小有限，若栈已满，一般会覆盖掉最旧的一项。还可以考虑对栈中每一项增加一个计数器，如果连续Call跳转到同一个地址，则并不压栈而是栈顶项计数器+1。这样可以很好地处理递归调用。

②其他预测方法：对于一些情况，跳转地址是有限的，因此有可能用一个Cache来覆盖跳转地址。由此可以完全依照方向预测的方法，只把PHT换成 Target Cache

四、分支预测失败时的恢复

当执行阶段发现预测失败时，可以记录在ROB中，停止取指，当这条分支指令是最旧的指令时，就可以清空流水线。

也可以选择Checkpoint的方式：遇到分支指令时把处理器的状态保存下来，具体包括重命名映射表和PC等。这种方式需要更多硬件资源，但是高效。

在清空流水线时，需要判断哪些指令位于分支预测错误的路径上。对此可以设计一个编号列表，为每一条分支指令分配一个编号，编号会一直伴随着每一条指令。

发现预测错误的分支指令时，在发射阶段之前的所有指令都要被抹掉，而之后的指令要根据编号进行判断。具体来说，ROB中记录了每条指令对应的编号，发生预测失败后将每条指令的编号与之比较，在ROB中置为无效。发射队列中进行类似处理。如果需要抹掉的指令较多，可以分多个周期进行，只要在新指令到达发射阶段之前完成即可。

分配编号值应当在解码阶段完成。为了简化编号队列，可以规定一个周期内只允许一条分支指令进入解码阶段。

另一个问题：判断预测结果时如何得知预测的跳转地址？对此可以使用PTAB来保存预测目标地址、NPC。

五、超标量处理器的分支预测

☆多端口问题 ☆并行问题 ☆取指判断问题