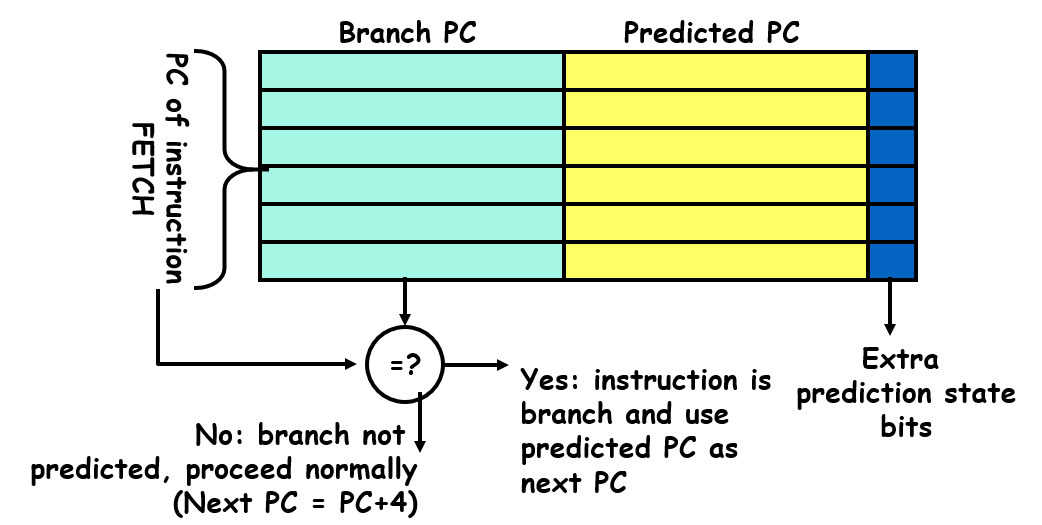
# 分支预测设计指导

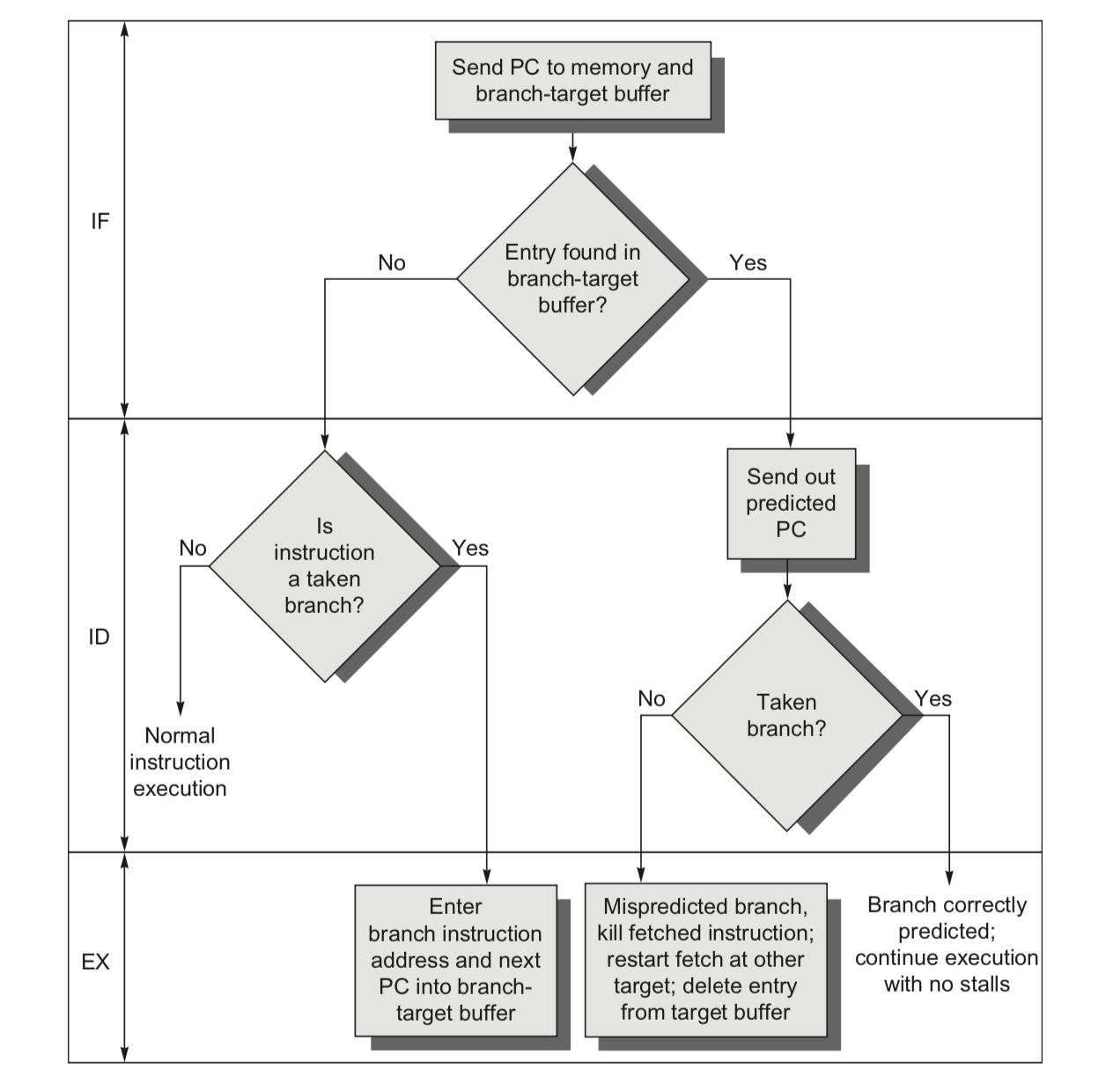
#### BTB实现

BTB（Branch Target Buffer）是动态分支预测的一种基本方法。它使用一个Buffer，里面记录了历史指令跳转信息。对于每一条跳转的Branch指令，它都将其写入buffer，记录其跳转的地址，并有一个标志位标记最近一次执行是否跳转。这样如果有一条在Buffer里的跳转指令将执行时，可以根据buffer记录的历史跳转信息，预测下一条要执行的指令地址，预测正确的话可以减小分支开销。BTB只使用了1-bit的历史信息，也可以视作1-bit BHT。



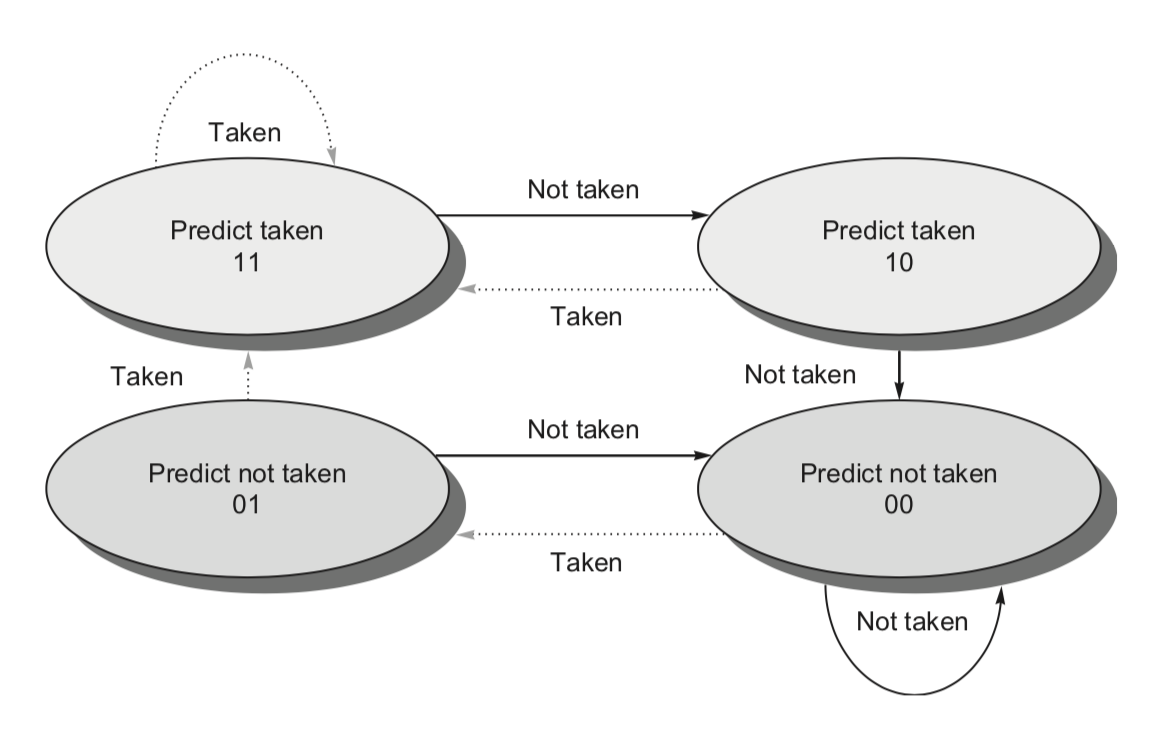
在我们之前实现的lab2 RV32I Core中，下一条PC地址是PC + 4。在添加了BTB之后，对于IF阶段产生的PC，在BTB Buffer里检查是否有对应项，如果有的话，根据其历史跳转记录，确定是否选择predicted PC作为下一PC。如果当前PC不在BTB表里，但在EX段发现是一条需要跳转的Branch指令，则在EX阶段更新BTB表。另外，如果PC在BTB表中，在EX阶段发现预测的跳转失败，也需要更新BTB表，并flush错误装载的指令。

状态机如下图。



#### BHT实现

BHT（Branch History Table）是动态分支预测的另一种基本策略。类似BTB，它也维护了一个N \* 2的cache作为buffer。其中，N是BHT表的项数（**一般取4096项**），根据PC的低位查找BHT表，每个项都维护了一个独立的2-bit状态机。如下图：

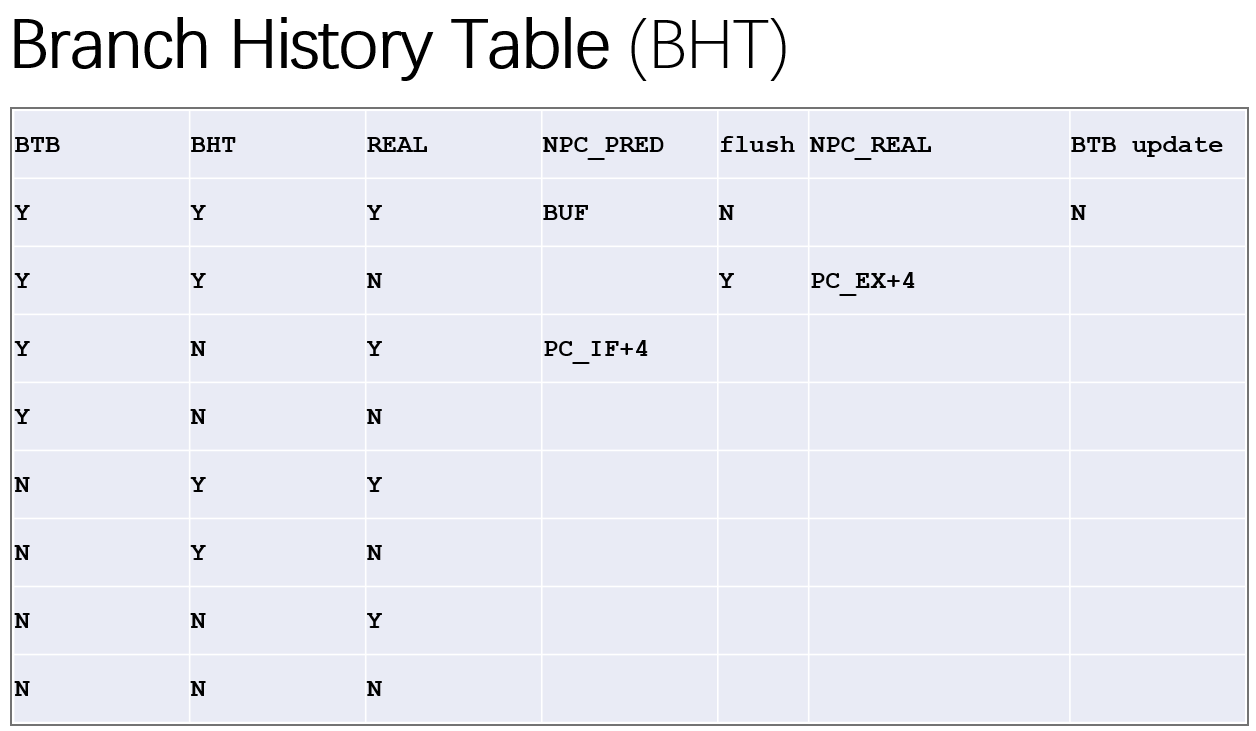


BHT和BTB一样，在IF阶段对当前PC预测其是否跳转。相较于BTB，BHT的预测准确度更高，在IF阶段，首先判断当前PC在BTB表中是否跳转，如果跳转，再到BHT表中寻找其是否跳转。只有两者都预测跳转时，才预测当前指令跳转，并将BTB表中的预测跳转地址作为下一条指令的PC地址。特别的，如果BHT表预测跳转，BTB表预测不跳转，或者BHT表预测不跳转，BTB表预测跳转，都不预测当前指令跳转。

在EX阶段，BHT表根据实际的跳转结果，更新2-bit的状态机，BTB表则在冲突时更新。

#### 实验报告

1. **在实验报告中补全下表：**



1. 上表前三列是输入，其余是输出。
2. BTB表示BTB的buffer是否命中；BHT表示当前指令地址对应BHT中的状态是否是predict taken状态；REAL表示当前分支指令是否真正跳转。其中BTB和BHT是否命中信号在IF阶段产生，随流水线段寄存器传递到EX阶段；REAL信号在EX阶段产生。
3. NPC\_PRED表示预测下一条指令地址，BUF表示从BTB中取出的地址；flush表示刷新流水线；NPC\_REAL表示EX阶段正确判断出的NPC。
4. BHT根据状态机更新；BTB在cache冲突时更新。
5. 动态分支预测根据BHT的是否命中来确定（因为更精确）；但是如果BTB没命中，BHT命中，那么NPC\_PRED选择PC\_IF+4。
6. **我们提供了btb.s、bht.s、QuickSort.s、**MatMul.s**四个测试样例，分别执行这四个测试样例，并在实验报告中分析以下内容：**

\* 分析分支收益和分支代价

\* 统计未使用分支预测和使用分支预测的总周期数及差值

\* 统计分支指令数目、动态分支预测正确次数和错误次数

\* 对比不同策略并分析以上几点的关系

\* 计算整体CPI和加速比

**注意**：由于QuickSort.s和MatMul.s执行完成后会陷入死循环，测试终止时间记为第一次执行到死循环的jal指令为止。