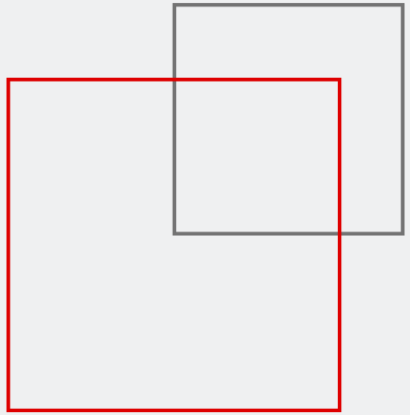
转移预测

动态转移预测

目录



控制冒险（control hazards）

- 控制冒险：当流水线遇到分支（转移）指令和其他改变PC值（不再是PC+4）的指令时，引起流水线的停顿。
- 例如 MIPS：
  - 无条件转移 (j, jal, jr)：跳转到指定位置
  - 条件转移 (beq, bne)
    - 转移失败：PC值加4
    - 转移成功：将PC值改变为转移目标地址

控制冒险（ control hazards ）

- 条件转移（分支转移）成功转移: 导致暂停3个时钟周期。
- 理想CPI = 1，但实际上Branch指令CPI=4;
- 若条件转移指令占比30%，实际CPI（cycle per instr.）= 1 + 30% × 3≈ 2

分支指令	IF	ID	EXE	MEM	WB					
分支后继指令		IF	Stall	Stall	IF	ID	EXE	MEM	WB	
分支后继指令+1						IF	ID	EXE	MEM	WB
分支后继指令+2							IF	ID	EXE	MEM
分支后继指令+3								IF	ID	EXE
分支后继指令+4									IF	ID

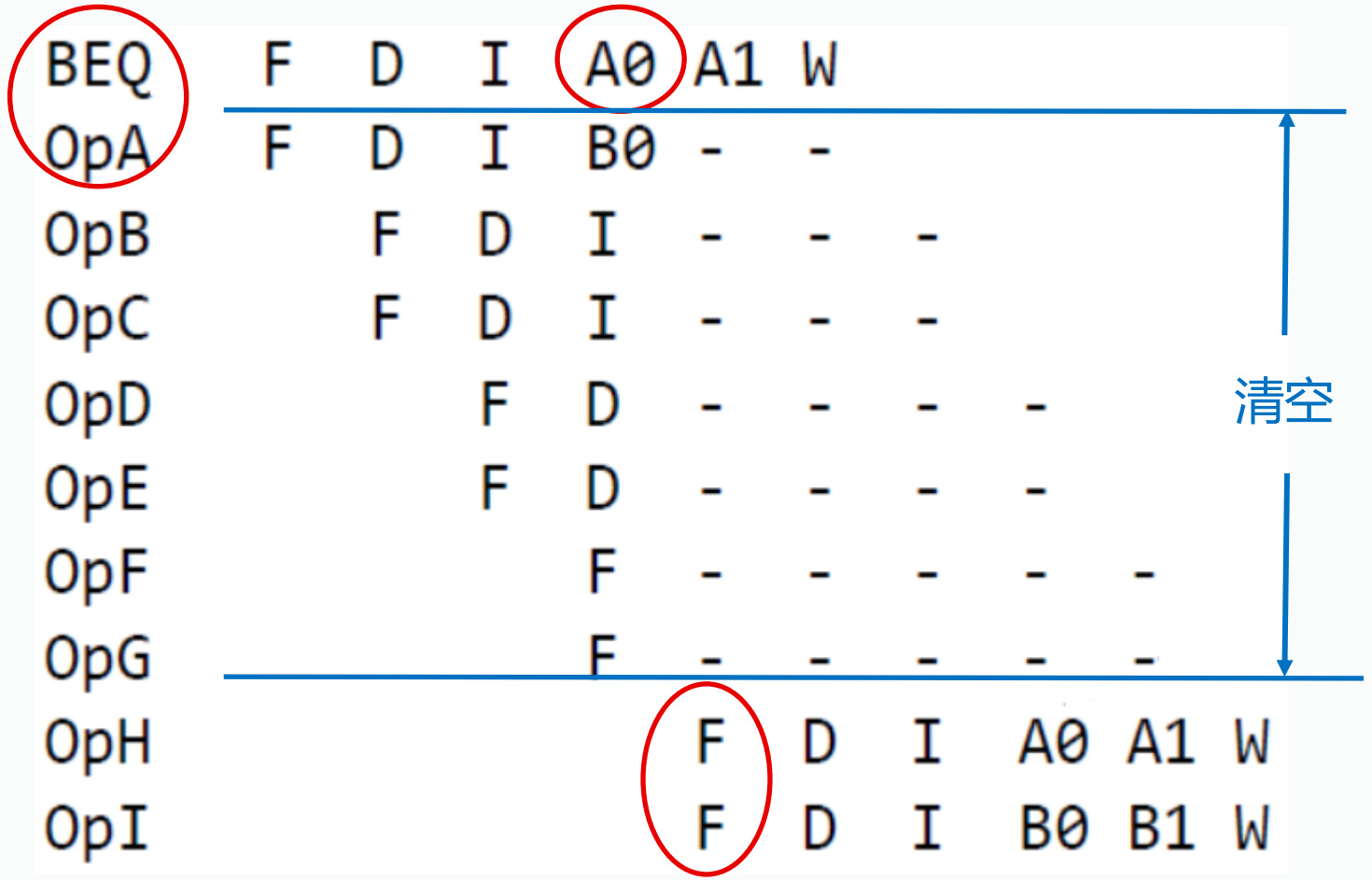
控制冒险（ control hazards ）

- 流水段数越深，控制冒险的开销越大
- 例如 Pentium 3：
  - 转移开销 10周期

1	2	3	4	5	6	7	8	9	10
Fetch	Fetch	Decode	Decode	Decode	Rename	ROBRd	Rdy/Sch	Dispatch	Exec

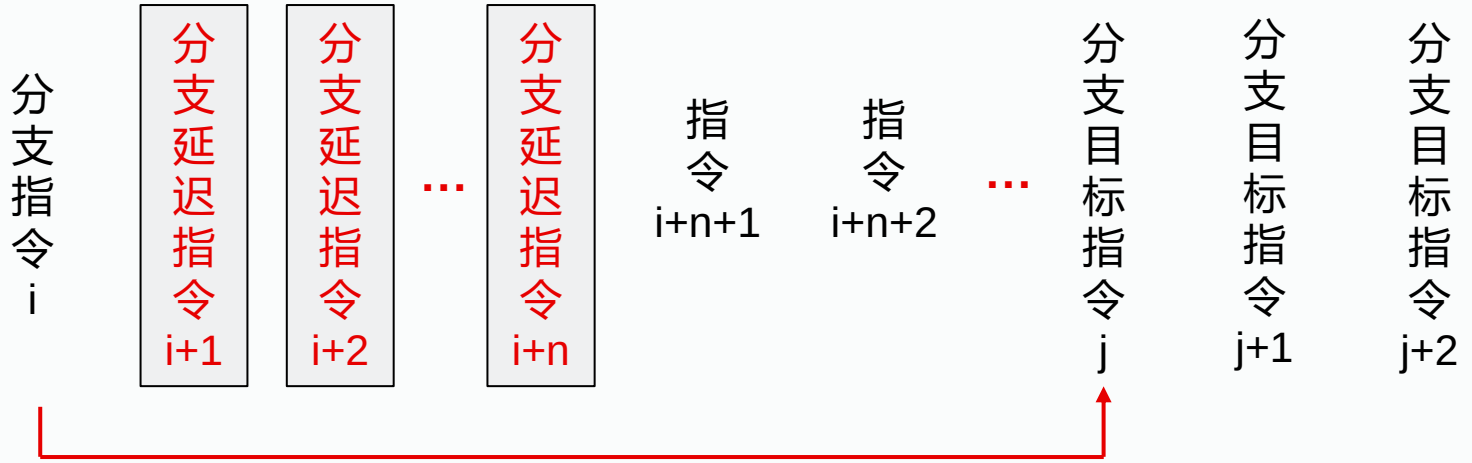
控制冒险（control hazards）

- 多发射
  - 冒险开销成倍增加
  - 转移开销×发射宽度



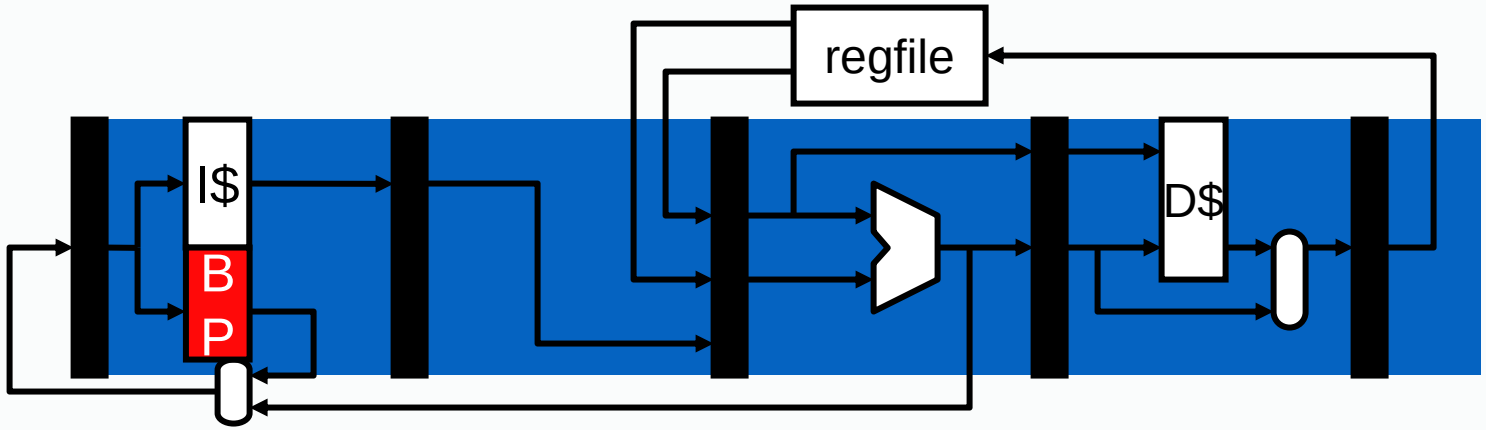
控制冒险（ control hazards ）

- 停顿？
  - 转移开销太大
- 转移延迟槽？
  - 编译器无法找到足够多的肯定会执行的指令（无论转移或是不转移时）移到branch指令后面，以隐藏转移指令的延迟；
- 解决方案：转移预测



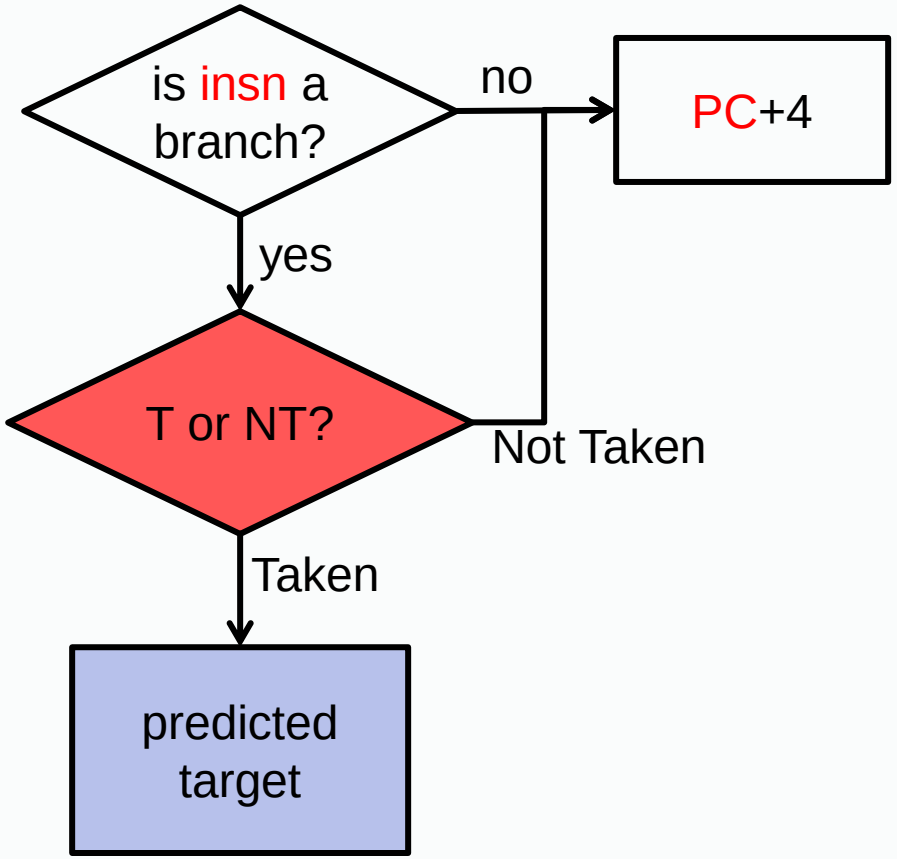
转移预测

- 预测方法：
  - 1. 静态：程序执行前预测
  - 2. 动态：程序执行时预测

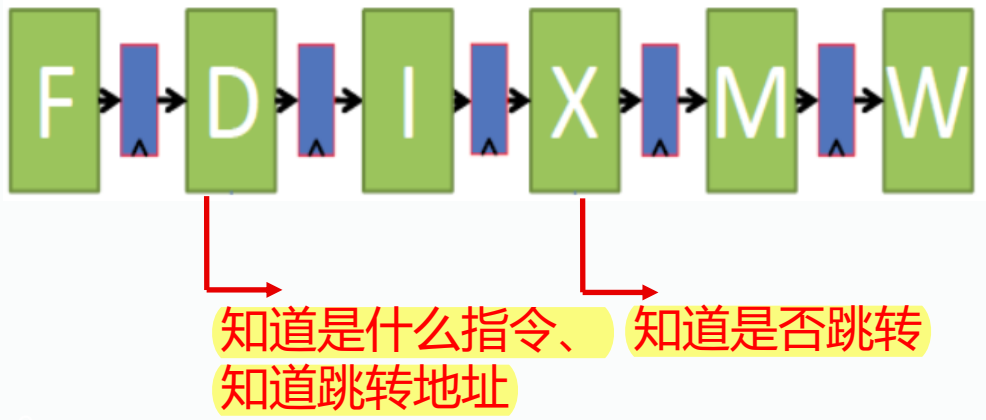


BP: Branch Prediction

动态转移预测

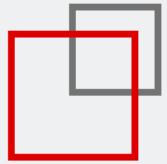


- **Step #1: 这是转移(branch)指令吗?**
- **Step #2: 转/不转?**
  - 转移方向的预测
- **Step #3: 如果转，转往哪里?**
  - 转移地址预测



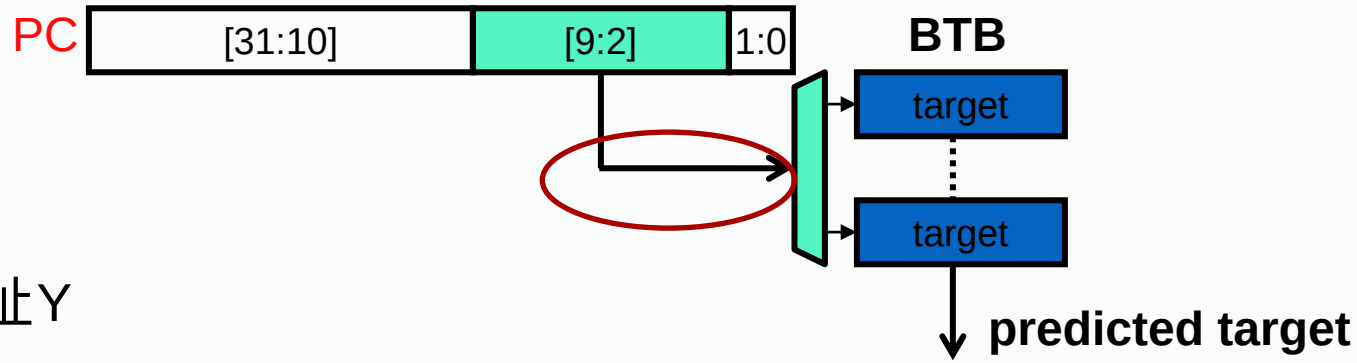
转移地址预测

Branch Target Prediction



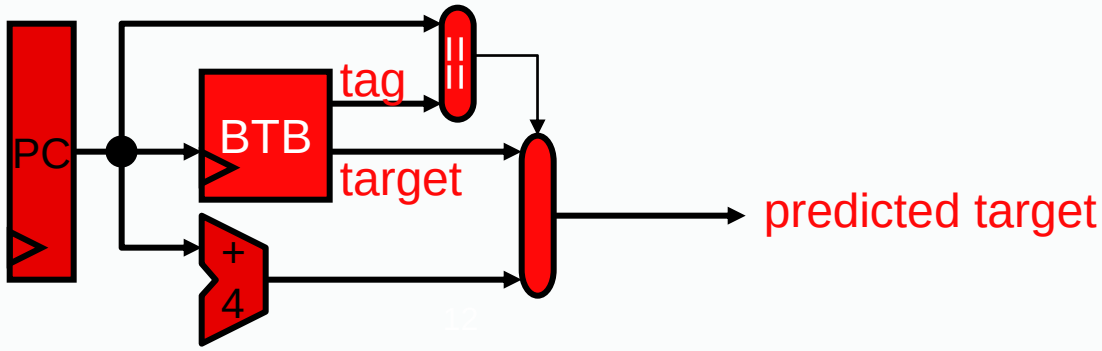
Branch Target Buffer: BTB

- **用历史预测未来**
  - 用硬件实现一张表
    转移历史表
- **Branch target buffer (BTB)**:
  - 上次 branch X 转了、并转往地址Y
  - 如果在X地址取指，接下来在Y地址取指
  - 用PC来检索BTB
  - 同名怎么办?
    - 两个PC最后几位相同
    - 没关系，只是预测



Branch Target Buffer: BTB

- BTB 预测：指令是不是Branch、分支转移目标
- 对每一条转移的Branch指令，更新它在 BTB中的记录：
  - BTB[PC].tag = PC, BTB[PC].target = target of branch
- 指令在取指时，会并行读 BTB
  - 检查tag, 如果等于当前PC，就预测它转移
  - Predicted PC = (BTB[PC].tag == PC) ? BTB[PC].target : PC+4



如何预测非直接转移目标

- BTB对直接转移目标有效
- 如何预测非直接跳转目标?
  - JR： 转移目标在寄存器中 → 每次都不一样
  - 函数调用
    - 动态链接(DLLs): 目标每次不变
    - 动态（虚拟）函数调用: 难以预测，但不常见
  - 函数返回
    - 难以预测，但可以想办法…

返回地址栈 Return Address Stack (RAS)

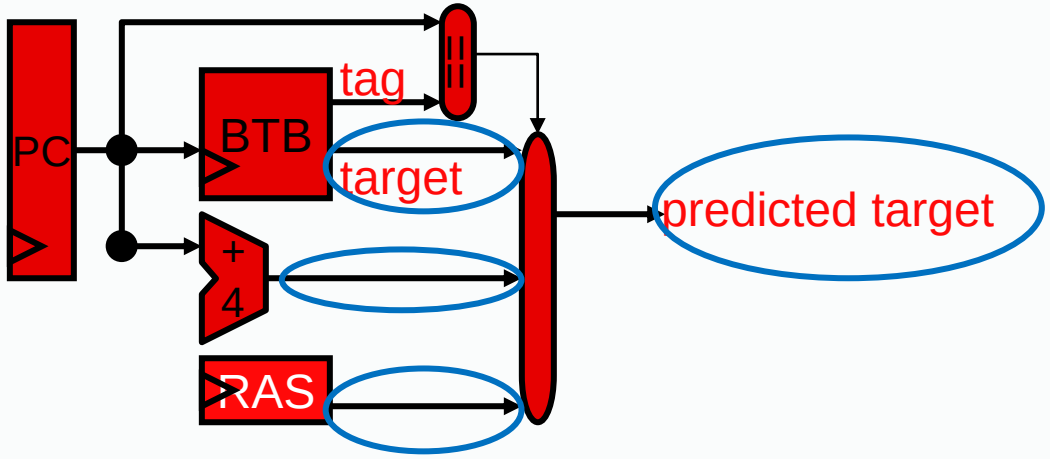
Return address stack (RAS)

- 调用函数时： RAS[TopOfStack++] = PC+4
- 函数返回时： Predicted-target = RAS[--TopOfStack]
- **Q:** 怎么在指令译码前知道一条指令是调用函数指令或函数返回指令？

Answer:

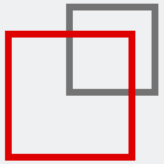
- 使用另外一个预测器
- 或者
  - 在BTB表项中加“return”标记
  - 在指令存储器中，设置预译码位

    （**pre-decode bits**）



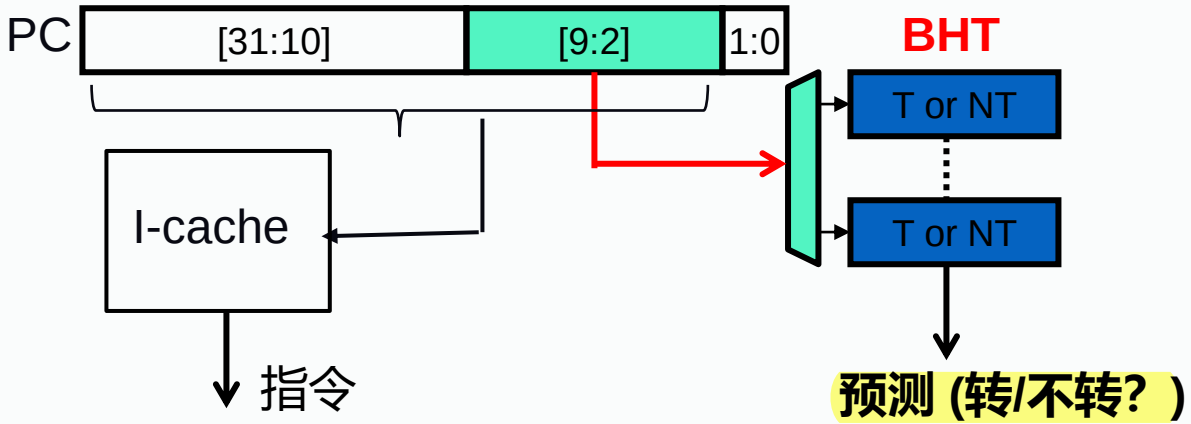
转移方向预测

Branch Outcome Prediction



转移方向的预测：转 / 不转（Branch Outcome）？

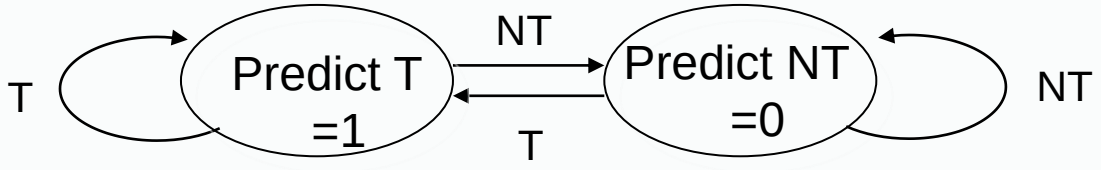
- 学习过去，预测未来
- 90%的branch指令是有倾向性的
  - 例如for、while 语句
- 转移历史表
  - Branch History Table(BHT)
- 一边取指令一边查表预测
- PC索引位重合怎么办?
  - 没关系，这只是一个预测



一位预测器

NT： Not Taken 不转移

T: Taken 转移

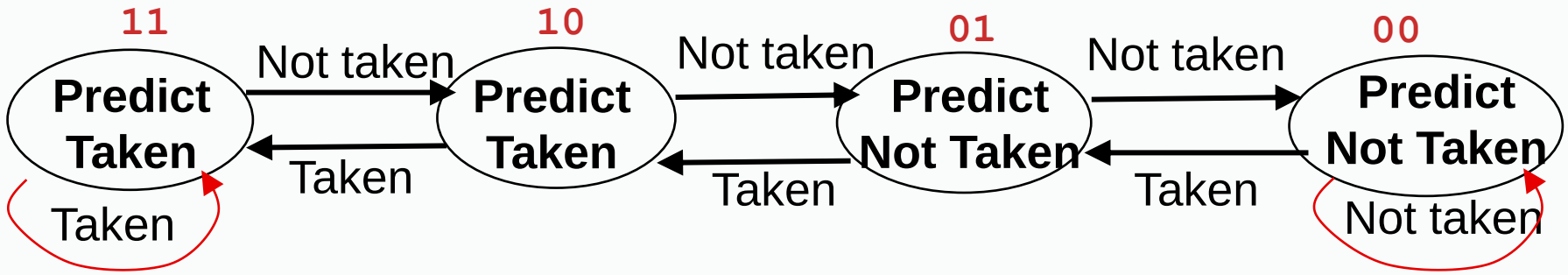



```
for (i=0; i < 100; i++)
    for (j=0; j <4; j++)
        A[i][j] ++;
```

Iteration	Prediction	Actual	Mispredict?
1	NT	T	Y
2	T	T	
3	T	T	
4	T	NT	Y
…			
1	NT	T	Y
2	T	T	
3	T	T	
4	T	NT	Y

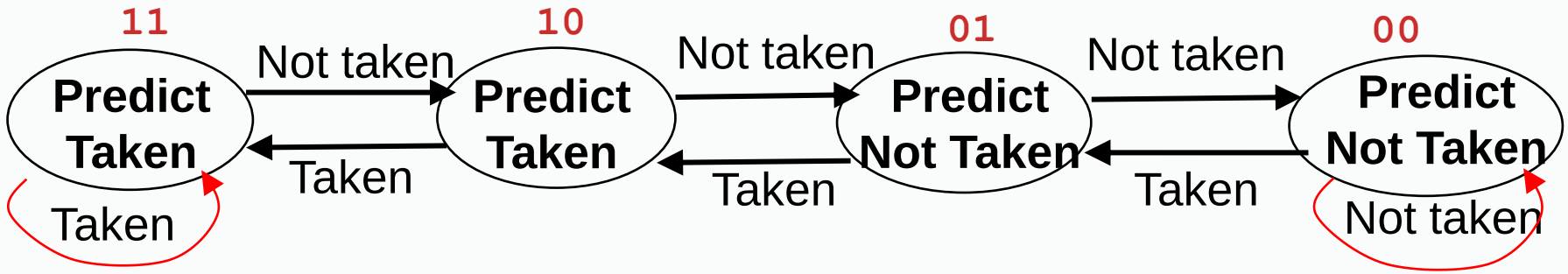
两位预测器

00: strong NT
01: weak NT
10: weak T
11: strong T



两位预测器

00: strong NT
01: weak NT
10: weak T
11: strong T




```
for (i=0; i < 100; i++)
    for (j=0; j <4; j++)
        A[i][j] ++;
```

Iteration	Prediction	Actual	Mispredict?
1	Strong NT	T	Y
2	Weak NT	T	Y
3	Weak T	T	
4	Strong T	NT	Y
…			
1	Weak T	T	
2	Strong T	T	
3	Strong T	T	
4	Strong T	NT	Y

转移指令之间可能有相关性

```
for (i=0; i<1000000; i++) {
  if (random() % 2 == 0) {

}
  if (i % 4 == 0) {
    …
  }

  if (i % 4 == 0) {
      …
  }
}
```

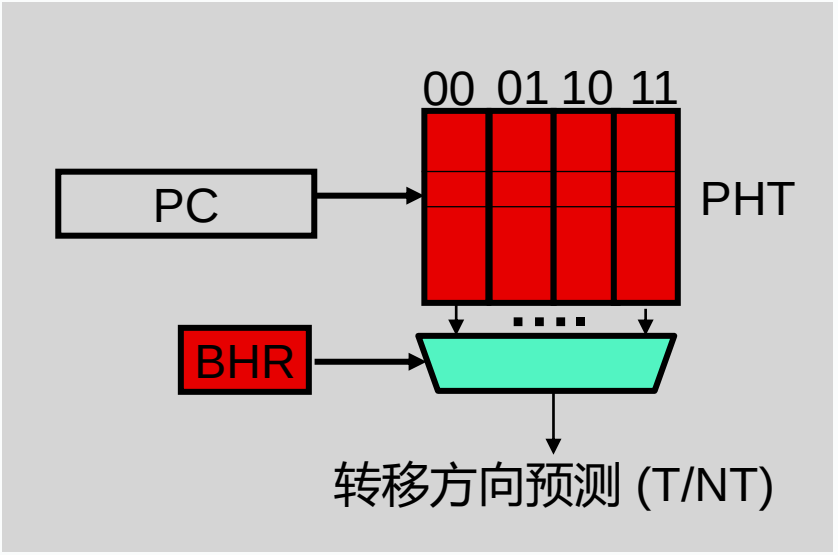
历史模式：111，预测：0

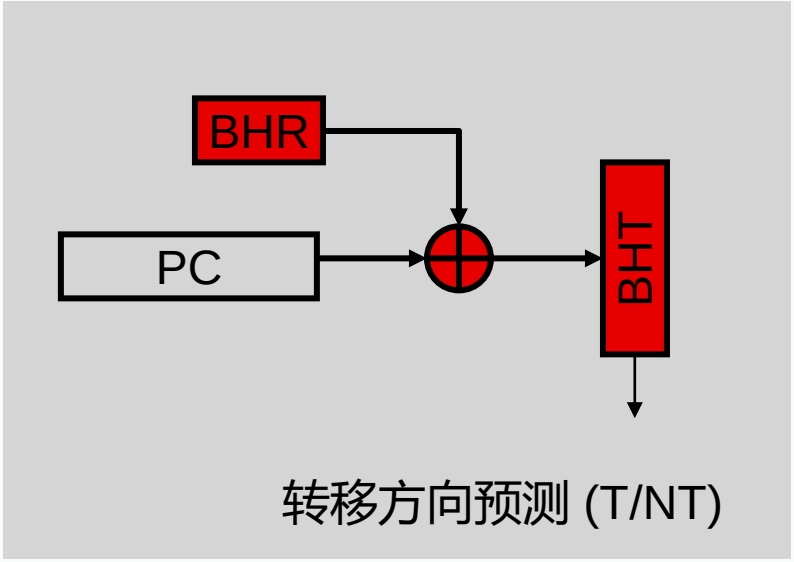
// 局部相关

// 全局相关

与转移历史相关的分支预测器

- 观察转移结果之间的相关性
- **转移历史寄存器 Branch History Register (BHR)**
  - 记录最近几次转移的模式（branch outcomes pattern）
- 转移模式历史表 PHT: Pattern History Table: 分情况的BHT
- Gshare 分支预测：用 PC xor BHR 检索 BHT，低功耗





Gshare 分支预测器：举例

```
for (i=0; i<1000000; i++) {
  if (i % 4 == 0) {
      …
    }   …
}
```

假设：

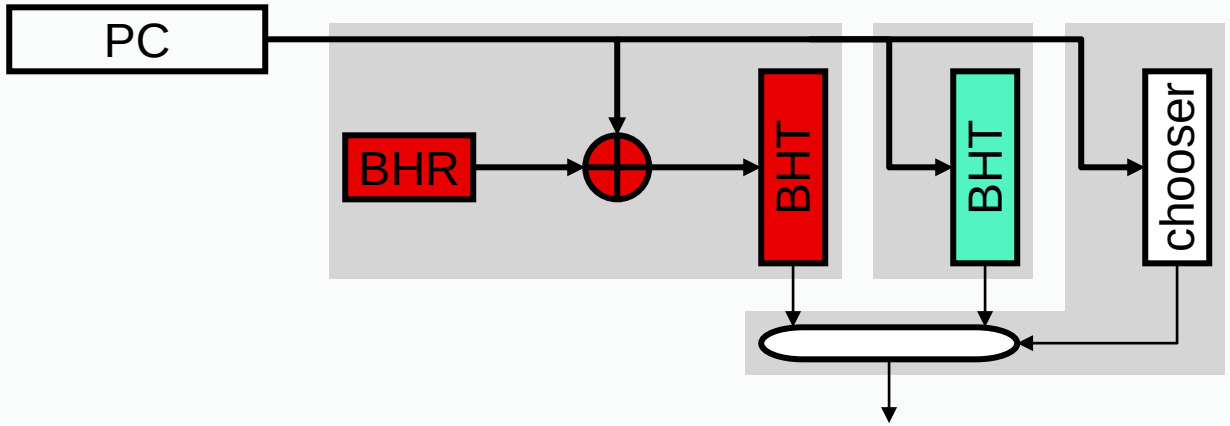
- 指令PC：最后10位都是0
- BHT： 一位预测器, 初始状态为NT

Branch History	Prediction
TTT	N
TTN	T
TNT	T
NTT	T

Time	State	BHR	Prediction	Outcome	Result?
0	N	NNN	N	T	wrong
1	N	NNT	N	T	wrong
2	N	NTT	N	T	wrong
3	N	TTT	N	N	correct
4	N	TTN	N	T	wrong
5	N	TNT	N	T	wrong
6	T	NTT	T	T	correct
7	N	TTT	N	N	correct
8	T	TTN	T	T	correct
9	T	TNT	T	T	correct
10	T	NTT	T	T	correct
11	N	TTT	N	N	correct

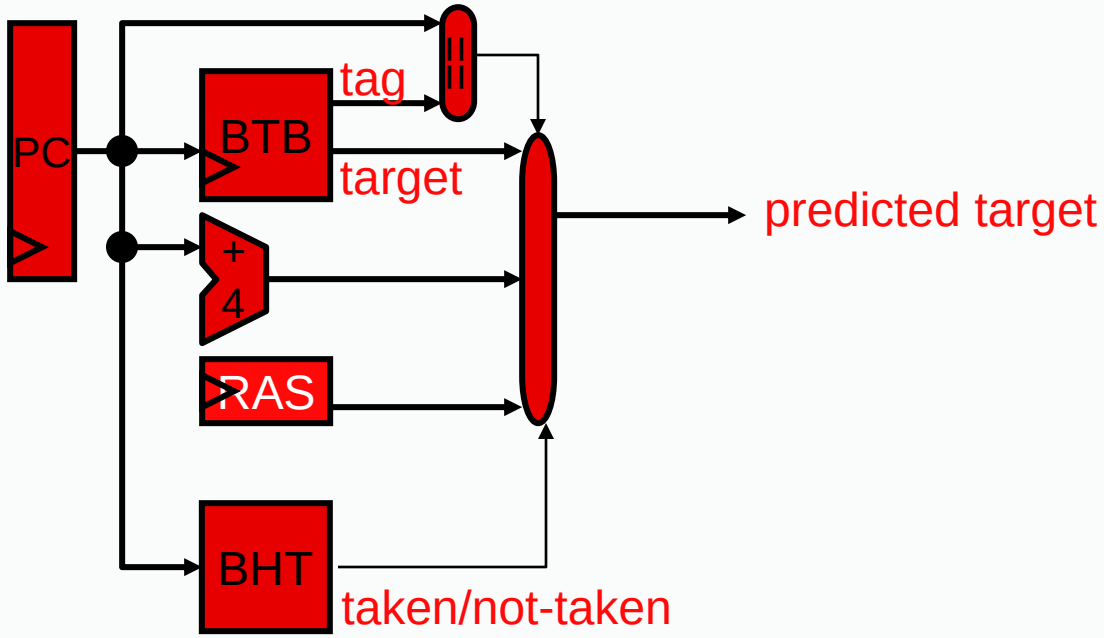
混合预测器

- **锦标赛预测器：**结合两个预测器
  - **简单预测器：1位或两位 （**for history-independent branches)
  - **历史相关预测器**（Correlated predictor ： for branches that need history)
  - **选择器**（Chooser)
    - 一开始选简单BHT，超过某个 threshold选相关预测器
    - 类似于机器学习中的集成学习（ensemble learning ）
  - **90–95**% accuracy



综合

- 将BTB和BHT结合起来



- 如果预测是正确的，转移指令不会增加任何开销

举例：动态转移预测的性能

- Branch转移指令占比 20%
- 预测不正确： Branch转移指令增加2周期的额外开销（Penalty)
  - 简单预测器: 75% 正确率
    - CPI = 1 + (20% * **25%** * 2) = **1.1**
  - 高级预测器: 95%正确率
    - CPI = 1 + (20% * **5%** * 2) = **1.02**

- 转移预测仍是一个重要的难题
  - 流水段数深: 典型的转移开销大于10周期

小结

控制冒险	超标量深度流水处理器中控制冒险引发的开销
转移预测	动态转移预测
转移目标预测	Branch Target Buffer: BTB
转移方向预测	Branch History Table: BHT

谢谢观看

上海交通大学