

5. 考虑一个深度流水线处理器，无分支指令时其基本 CPI 为 1。对于分支指令采用两种方案，方案 A 使用一个分支目标缓存（BTB），缓存缺失代价为额外 3 个周期，缓存命中但预测错误的代价为额外 4 个周期，缓存命中且预测正确则无分支代价。假设这个 BTB 的命中率为 90%，预测正确率为 90%。方案 B 不使用分支预测，分支代价固定为额外 2 个周期。假设分支频率为所有指令的 15%，则处理器采用方案 A 比采用方案 B 快多少？

$$S_{AB} = \frac{T_B}{T_A} = \frac{CPI_B \times T_{clkB} \times N_{instrB}}{CPI_A \times T_{clkA} \times N_{instrA}} = \frac{CPI_B}{CPI_A}$$

$$CPI_A = 1 + 15\% \times [(1-90\%) \times 3 + 90\% \times (1-90\%) \times 4]$$

$$= 1.099$$

$$CPI_B = 1 + 15\% \times 2 = 1.3$$

$$\Rightarrow S_{AB} = 1.18$$

12. 考虑如下的代码片段：

```

li      a0,0
li      a4,10000
addi    a1,a0,0
Loop:   addi    a3,a0,2
        rem     a2,a1,a3
0xe44:  bne     a2,a0,Rem2      //B1
        #...CodeA
Rem2:   addi    a3,a0,5
        rem     a2,a1,a3
0xe84:  bne     a2,a0,End       //B2
        #...CodeB
End:    addi    a1,a1,1
0xec0:  bne     a1,a4,Loop      //B3

```

- 1) 写出与该汇编代码功能一致的 C 语言代码。
- 2) 无分支预测时，上述代码中的三条 bne 指令发生跳转的比例分别是多少？
- 3) 引入一个静态分支预测器，该预测器对向前跳转总是给出“跳转”预测，对向后跳转总是给出“不跳转”预测，则上述代码中的三条 bne 指令的预测准确率分别是多少？

```

(1) for (i=0; i<10000; i++) {
    if (i%2==0) {#CodeA}
    else if (i%5==0) {#CodeB}
}

```

$$(2) B_1: \frac{1}{2} = 50\%$$

$$B_2: \frac{4}{5} = 80\%$$

$$B_3: \frac{9999}{10000} = 99.99\%$$

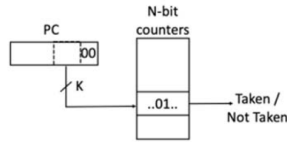
(2) 共进行 10000 次循环。

跳转次数: $B_1: 5000$

$B_2: 2000 \Rightarrow$ 比例为 $5000:2000:9999$

$B_3: 9999$

13. 仍考虑题 12 中的代码片段，现引入局部预测器，如下图所示。该预测器使用 PC 的第 $[(K+2):3]$ 共 K 位索引一张预测器表，该表的每个表项是一个 N-bit 的计数器，计数器的最高位用于预测是否跳转（1 为跳转，0 为不跳转），并根据实际跳转结果更新计数器的值（跳转自增 1，增至 2^N-1 后不再变化；不跳转自减 1，减至 0 后不再变化）。假设所有计数器的初始值均为 0。



- 1) 要保证上述代码片段被映射到不同的局部预测器，K 的最小值是多少？
- 2) 要使该预测器对三条 bne 指令的预测准确率均不低于题 12 中的静态预测器，N 的最小值是多少？
- 3) 对上述给出的最小 N，在程序稳态时，三条 bne 指令的预测准确率分别是多少？

(1) ∵ 有 3 条分支跳转指令

∴ K 的最小值为 2

(2) $N=1$ 时，对 B_1 的预测准确率为 0，故不可取。

$N=2$ 时，预测准确率： $B_1: 50\%$

$B_2: 80\%$

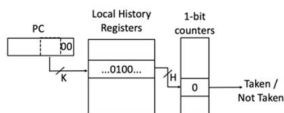
$B_3: 99.99\%$

∴ N 的最小值为 2

(3) 稳态： $B_1: 50\%$ $B_2: 80\%$ $B_3: 100\%$

14. 仍考虑题 12 中的代码片段，现引入局部分支历史，如下图所示。该预测器使用 PC 的第 $[(K+2):3]$ 共 K 位索引一个局部分支历史表，其每个表项是一个 H 位的局部分支历史，

该 H 位的历史被进一步用于索引一张单比特计数器构成的预测表（1 为跳转，0 为不跳转）。计数器会根据实际跳转结果进行更新。

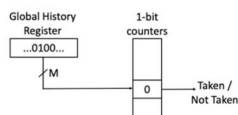


假设 K 的值足够大，使得上述代码片段中的不同分支会被映射到局部分支历史表中的不同位置。则为了使得三条 bne 指令都能在程序稳态时被完全准确地预测，H 的最小值是多少？

考虑跳转情况变化，由于 B_1 周期为 2， B_2 周期为 5

⇒ H 的最小值为 5

15. 仍考虑题 12 中的代码片段，现引入全局分支历史，如下图所示。该预测器拥有一个 M 位的 GHR，记录了程序中任意分支的跳转历史。当一个新分支被执行时，跳转分支使得 GHR 左移 1 位并在末位写入 1，未跳转分支则使得 GHR 左移 1 位并在末位写入 0。GHR 被用于索引一张单比特计数器构成的预测表（1 为跳转，0 为不跳转）。计数器会根据实际跳转结果进行更新。



为了使得三条 bne 指令都能在程序稳态时被完全准确地预测，M 的最小值是多少？

经历 10 次循环可以得到完整的跳转周期，故 M 最小值为 $3 \times 10 = 30$

16. 在实际应用中常常能遇到如下的代码场景：

```
for (int i=0; i<P; i++){           //outer loop
    for (int j=0; j<Q; j++){       //inner loop
        //SomeCode
    }
}
```

$Q \geq 2$.

视进入循环体执行为“跳转”，不进入为“不跳转”。现有两种分支预测器方案：方案 A 使用题 13 中的预测器结构 ($N=1$)，方案 B 使用题 14 中的预测器结构 ($H=Q$)。假设 $Q \geq 2$ 且内循环体 (inner loop) 内部没有分支指令、外循环体 (outer loop) 针对变量 i 的分支总是被预测器忽略、预测器的 K 值足够大、所有计数器的初始值均为 0。试分析当 P 和 Q 满足什么数值关系时，方案 A 的预测准确率优于方案 B？

循环次数为 PQ .

A: 进出循环各错一次. 错误总数: $2P+2$

B: 稳态时对于内层循环预测正确率为 100%

错误数: $\left. \begin{array}{l} \text{内层: 只有第一次错} \Rightarrow Q \\ \text{外层: 只有一个循环} \Rightarrow P \end{array} \right\} \text{错误总数} = P+Q$.

A 优于 B $\Leftrightarrow 2P+2 < P+Q$

即 $P < Q-2$

17. 考虑如下的指令序列：

```
Loop:  lw    a4,0(a3)
      addi   a3,a3,4
      addi   a1,a1,-1
B1:    beqz  a4,B2
      addi   a2,a2,1
B2:    bnez  a1,Loop
```

假设 a1 初值为 n ($n > 0$)，a2 初值为 0，a3 初值为 p (p 为一个指向 32 位整型数组首地址的指针)。

- 假设处理器使用 2 位局部预测器，分支 B1 和 B2 映射到不同的预测器表项。若 $n=8$ 且数组的数据模式为 $p[] = \{1,0,1,0,1,\dots\}$ 。则上述代码执行过程中一共会发生多少次错误预测？
- 现引入 1 位的全局分支历史，1) 中的其他假设不变，则上述代码执行过程中一共会发生多少次错误预测？
- 若改为 2 位的全局分支历史表，1) 中的其他假设不变，则上述代码执行过程中一共会发生多少次错误预测？
- 比较上述结果，分析在该情境中全局分支历史表的位数对预测准确率有怎样的影响？当 n 非常大时，上述哪种预测器表现最好？
- 当数组 $p[]$ 的数据模式变为在 0 和 1 之间以均等概率随机取值时，4 中的结论有什么变化？

```
for (i=n; j=0; k=0; i>0; i--) {
    if (a[j]==0) k++;
    j++;
}
```

① 进循环: 2 次 出循环: 1 次

循环中 B1 分支预测正确率为 50%。 \Rightarrow 错 $n \times 50\% = 4$ 次。

\therefore 总共错 7 次。

② 由于只有 1 位，故只能记录前 1 条指令的跳转情况。第一次循环后 B1 前一条分支均跳转与 1 无差别，错 4 次，而由于 B2 要维护 GHR=0 或 1 的两个 LHT，故进循环时会错 4 次，出循环会错 1 次 \Rightarrow 共错 9 次。

③ 若 0 表示不跳，1 表示跳。

B1: GHR 为 00 时，下一次 B1 会跳转，会在开始错 2 次

GHR 为 10 时，下一次 B1 不会跳转，故不会错

B₂: 稳态内跳转. 要维护 GHR 为 10 和 11 的两个 LHT
故进循环会错 4 次. 出循环错 1 次.

⇒ 错误总数为 2+5=7 次.

(4) 位数增加时更易找到跳转的周期性规律, 在稳态时可显著降低错误率.
但初始维护的代价也会提高.

当 n 比较大时, (2) 中预测器表现更好.

(5) 随机取值时, (2) 与 (3) 中预测器对于 B₁ 的跳准预测准确度都是 50%.

但由于 (2) 中 B₂ 维护的 LHT 个数少, 达到稳态代价更低.

∴ (2) 比 (3) 中预测器表现更好

18. 解释为什么即使在顺序的 5 级 RISC 流水线中, 指令引发的异常也可能会乱序产生? 为了支持精确的异常处理, 流水线是如何做到对乱序产生的异常进行 (按程序顺序的) 顺序处理的?

异常的发生不一定与流水线中的当前处于执行状态的指令有直接关系, 可能是由前面或后面的指令引起的.
如取指错误或访存错误.

使用 ROB 以支持精确的异常处理, 将指令完成执行的过程和指令提交过程分开, 记录指令的顺序.

如果指令发生异常, 在指令提交阶段, 会触发异常处理机制. 完成异常处理以后, 处理器会从发生异常的指令位置重新执行. 由于 ROB 是顺序提交指令的, 可以保证实现精确异常处理.

20. 考虑一个拥有浮点单元的单发射乱序处理器, 该处理器包含以下假设:
- 处理器的浮点单元包含一个 2 运算周期的加法器, 一个 10 运算周期的乘法器, 和一个单执行周期的浮点加载/存储单元, 加法和乘法器均是完全流水化的。
 - 当发生写回冲突时, 更早的指令会获得优先写回。
 - 浮点指令的结果只能在写回阶段完成后被其他指令使用, 整型指令的结果则可以提前。
 - 处理器使用寄存器重命名, 从 T0、T1、T2 起有不受限制的重命名寄存器可用。
 - 译码级每周可以至多 1 条重命名后的指令添加到 ROB 中, 指令通过 ROB 顺序提交且每周至多提交 1 条指令。指令能够被提交的最早时间是完成写回后的下一个周期。
 - 忽略前端取指, 指令经过译码、发射、执行和写回后即可完成执行并提交。

现考虑如下的指令序列:

```

I1:    fld      f1, 5(a0)
I2:    fmul.d   f2, f1, f0
I3:    fadd.d   f3, f2, f0
I4:    addi     a0, a0, 8
I5:    fld      f1, 5(a0)
I6:    fmul.d   f2, f1, f1
I7:    fadd.d   f2, f2, f3
    
```

- 1) 如果 ROB 的深度是无限的, 将下表补充完全。(部分结果已给出)

	周期				操作码	目标	源 1	源 2
	Decode (ROB enqueue)	Issue	WB	Committed				
I1	0	1	2	3	fld	T0	a0	—

I2	1	3	13	14	fmul.d	T1	T0	f0
I3	2	14	16	17	fadd.d	T2	T1	f0
I4	3	16	18	19	addi	a0	a0	—
I5	4	17	19	20	fld	T0	a0	—
I6	5	20	30	31	fmul.d	T1	T0	—
I7	6	31	33	34	fadd.d	T1	T1	T2

2) 如果 ROB 仅容纳 2 条指令，当一条指令提交后的下一周期该条目可以被新指令占据。重新将下表补充完全。（部分结果已给出）

	周期				操作码	目标	源 1	源 2
	Decode (ROB enqueue)	Issue	WB	Committed				
I1	0	1	2	3	fld	T0	a0	—
I2	1	3	13	14	fmul.d	T1	T0	f0
I3	4	14	16	17	fadd.d	T2	T1	f0
I4	15	17	19	20	addi	a0	a0	—
I5	18	19	20	21	fld	T0	a0	—
I6	21	22	32	33	fmul.d	T1	T0	—
I7	22	33	35	36	fadd.d	T1	T1	T2