计算机系统结构
第五章　指令级并行

吴非，曹强

华中科技大学信息存储系统教育部重点实验室
武汉光电国家实验室存储与显示功能实验室

目　录

- 第一章 计算机体系结构的基本概念
- 第三章 流水线技术
- 第五章 指令级并行
- 第七章 存储层次
- 第八章 输入输出系统
- 第九章 互联网络
- 第十章 多处理机

5.1 指令级并行概念

- 当指令之间不存在相关时，它们在流水线中是可以重叠起来并行执行的。这种指令序列中存在的潜在并行性称为指令级并行。

  （Instruction-Level Parallelism，简记为ILP）

- 本章研究：如何通过各种可能的技术，获得更多的指令级并行性。

  （硬件技术）

- 必须要硬件技术和软件技术互相配合，才能够最大限度地挖掘出程序中存在的指令级并行。

1.流水线处理器的实际CPI

- $CPI_{流水线} = CPI_{理想}$ + 各类停顿周期数的总和

  流水线的理想CPI是流水线的最大流量。

  各类停顿包括：

  - 结构相关停顿：是由于两条指令使用同一个功能部件而导致的停顿。
  - 控制相关停顿：是由于指令流的改变（如分支指令）而导致的停顿。
  - RAW、WAR和WAW停顿：由数据相关造成的。

- 减少其中的任何一种停顿，都可以有效地减少CPI，从而提高流水线的性能。

2. 本章要研究的技术以及它们所克服的停顿

技术	主要克服的停顿	章节
基本流水线调度	数据先写后读相关停顿	5.3
寄存器换名	数据写后写相关和先读后写相关停顿	5.3
动态分支预测	控制相关停顿	5.4
多指令流出 （超标量和超长指令字）	提高理想CPI	5.5

5.4 动态分支预测

所开发的ILP越多，控制相关的制约就越大，分支预测要有更高的准确度。

- ✓ 在n-流出的处理机中，遇到分支指令的可能性增加了n倍
- ✓ 要给处理器连续提供指令，就需要准确地预测分支

5.4 动态分支预测

动态分支预测：在程序运行时，根据分支指令过去的表现来预测其将来的行为。

- 如果分支行为发生了变化，预测结果也跟着改变。

分支预测的有效性取决于：

- 预测的准确性
- 预测正确和不正确两种情况下的分支开销

决定分支开销的因素：

- ✓ 流水线的结构
- ✓ 预测的方法
- ✓ 预测错误时的恢复策略

5.4 动态分支预测

采用动态分支预测技术的目的

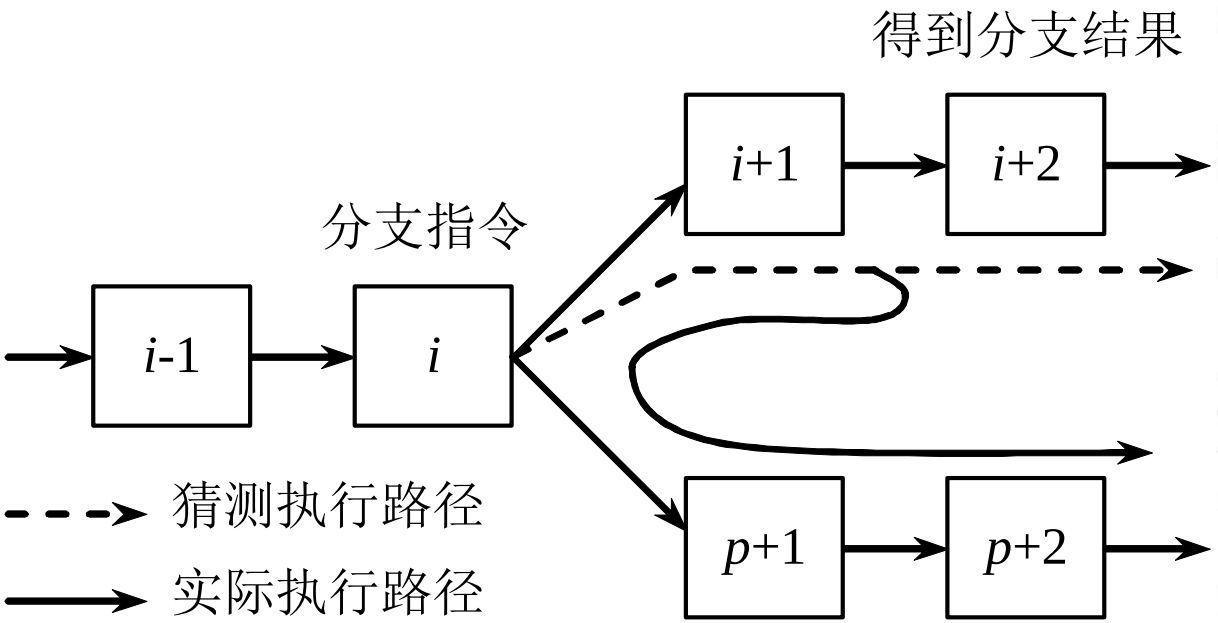
- 预测分支是否成功
- 尽快找到分支目标地址（或指令）

需要解决的关键问题

- 如何记录分支的历史信息，要记录哪些信息？
  - 仅记录最近一次或最近几次的分支历史；
  - 记录分支成功的目标地址；
  - 记录分支历史和分支目标地址，相当于前面两种方式的结合；
  - 记录分支目标地址的一条或若干条指令
- 如何根据这些信息来预测分支的去向，甚至提前取出分支目标处的指令？

5.4 动态分支预测

在预测错误时，要作废已经预取和分析的指令，恢复现场，并从另一条分支路径重新取指令。



5.4.1 采用分支历史表 BHT

1. 分支历史表BHT（Branch History Table）

- 最简单的动态分支预测方法。
- 用BHT来记录分支指令最近一次或几次的执行情况（成功还是失败），并据此进行预测。

2. 只有1个预测位的分支预测

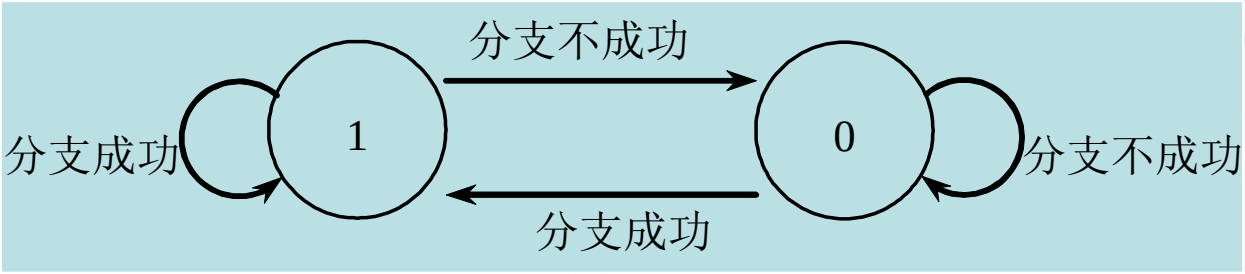
记录分支指令最近一次的历史，BHT中只需要1位二进制位。

5.4 动态分支预测

分支历史表采用1位来记录历史信息的方法

1. 只有1个预测位的分支预测缓冲 索引：分支指令地址的低位。

- 存储区：1位的分支历史记录位，又称为预测位，
  记录该指令最近一次分支是否成功。
  - “0”记录分支不成功
  - “1”记录分支成功
- 状态转换图



只有1个预测位的分支预测缓冲状态转换图

5.4 动态分支预测

2. 分支预测缓冲技术包括两个步骤

- 分支预测

  如果当前缓冲记录的预测位为“1”，则预测分支为成功；

  如果预测位为“0”，则预测分支为不成功。

- 预测位修改

  如果当前分支成功，则预测位置为“1”；如果当前分支不成功，预测位置为“0”。

5.4 动态分支预测

例 一个循环共循环10次，它将分支成功9次，1次不成功。假设此分支的预测位始终在缓冲区中。那么分支预测的准确性是多少？

解：这种固定的预测将会在第一次和最后一次循环中出现预测错误。

第一次预测错误是源于上次程序的执行，因为上一次程序最后一次分支是不成功的。

最后一次预测错误是不可避免的，因为前面的分支总是成功，共九次。

因此，尽管分支成功的比例率是90%，但分支预测的准确性为80%（2次不正确，8次正确）。

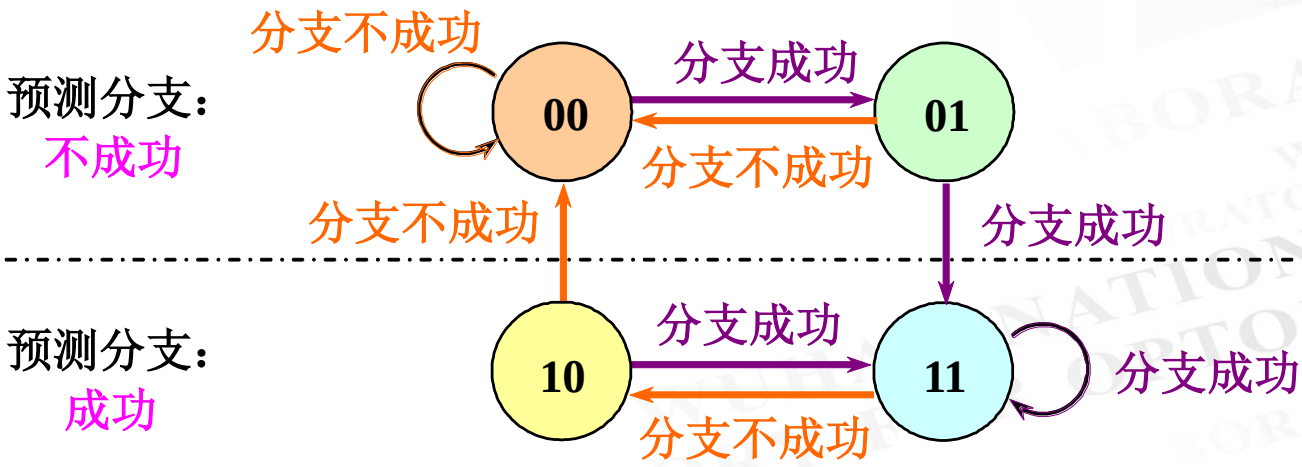
5.4 动态分支预测

- 1位预测机制的缺点

只要预测出错，往往是连续两次而不是一次。

- 解决方法：采用两个预测位的预测机制。

在两个预测位的分支预测中，更改对分支的预测必须有两次连续预测错误。



具有两个分支预测位的分支预测缓冲状态转换机制

5.4 动态分支预测

➢ n位分支预测缓冲

✓ 采用n位计数器，则计数器的值在0到2^n-1之间：

当计数器的值大于或等于最大值的一半（2^{n-1}）时， 预测下一次分支成功；否则预测下一次分支不成功。

✓ 预测位的修改和两位预测时相同：

当分支成功时计数器的值加1，不成功时减1。

✓ 研究表明：

n位分支预测的性能和两位分支预测差不多，因而大多数处理器都只采用两位分支预测。

但是BHT方法对于改进后的MIPS流水线没有帮助

5.4.2 分支目标缓冲器BTB

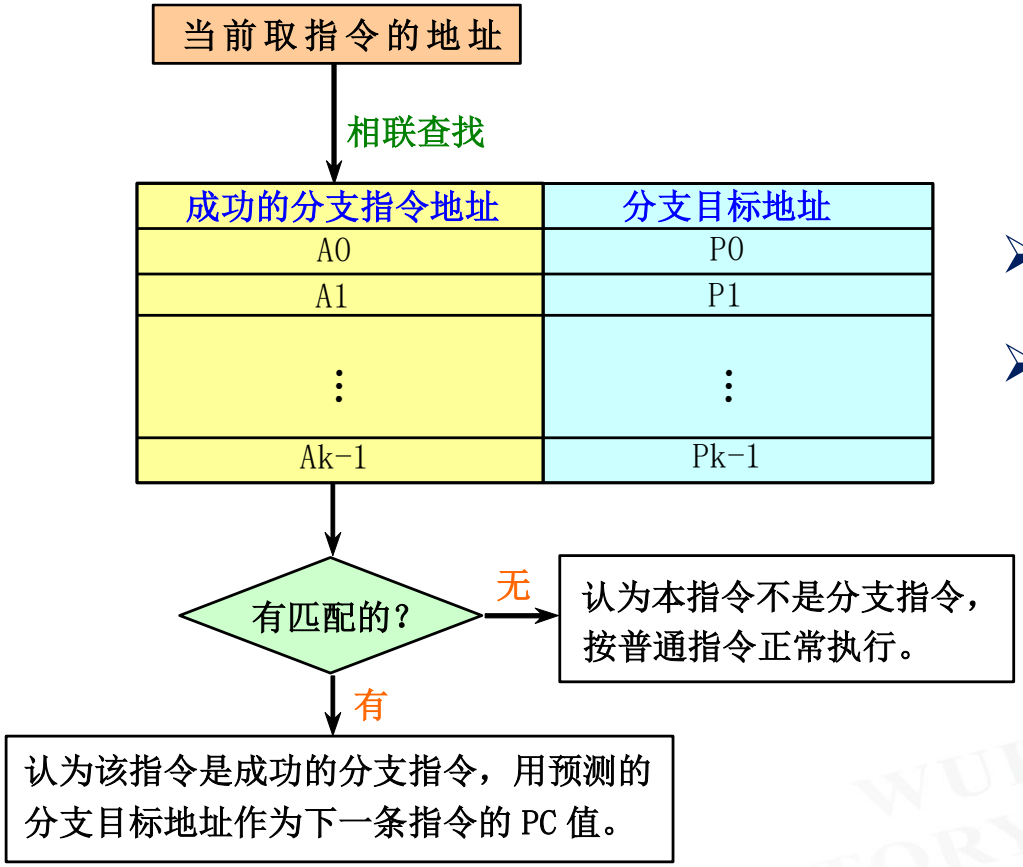
目标：将分支的开销降为零。

方法：分支目标缓冲

- 将分支成功的分支指令的地址和它的分支目标地址都放到一个缓冲区中保存起来，缓冲区以分支指令的地址作为标识；取指令阶段，所有指令地址都与保存的标识作比较，一旦相同，我们就认为本指令是分支指令，且认为它转移成功，并且它的分支目标（下一条指令）地址就是保存在缓冲区中的分支目标地址。
- 这个缓冲区就是分支目标缓冲区（**Branch-Target Buffer**，简记为**BTB**，或者**Branch-Target Cache**）。

5.4 动态分支预测

1. 分支目标缓冲结构和工作过程

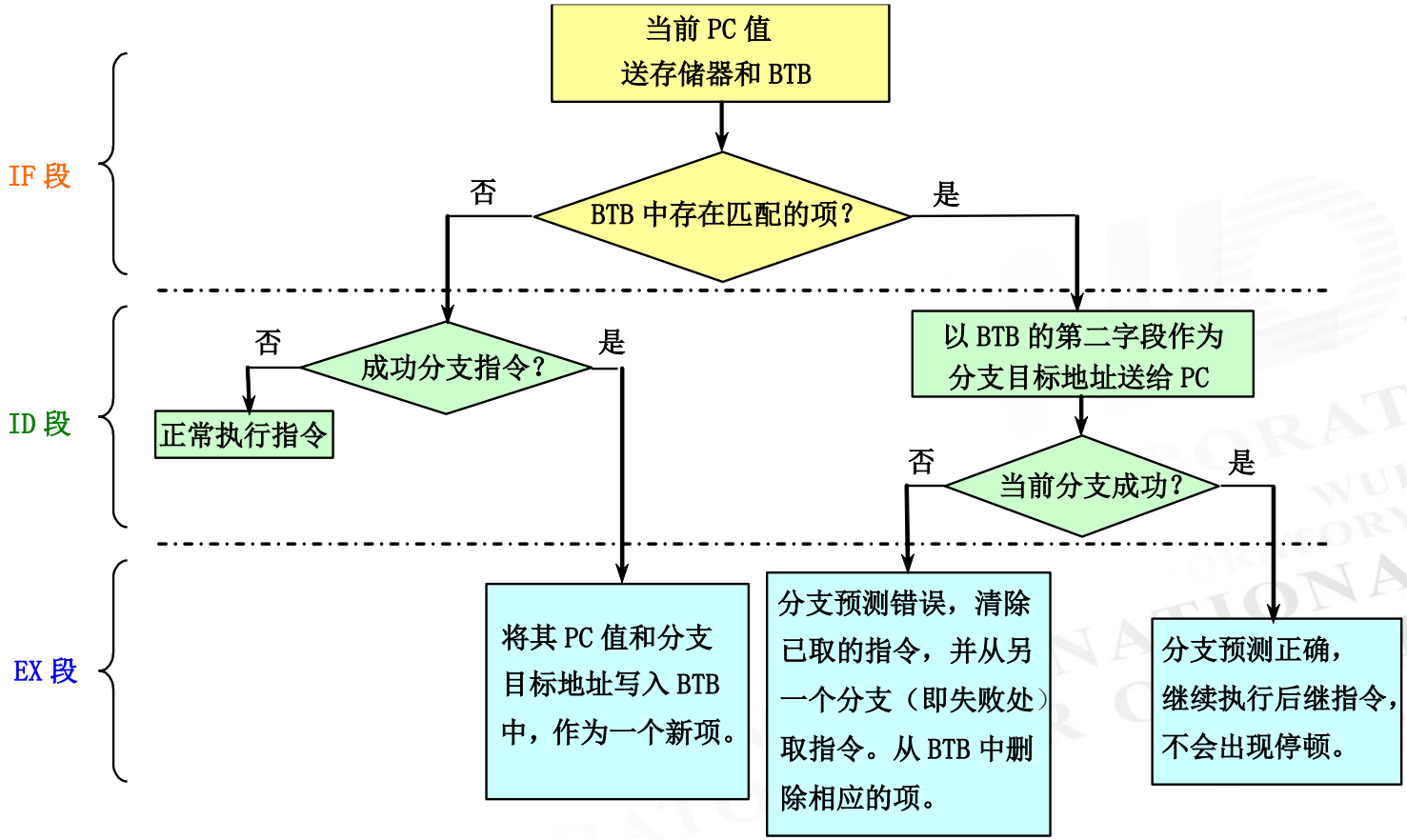


- 用专门的硬件实现的一张表格
- 表格中的每一项至少有两个字段
  - ✓ 执行过的成功分支指令的地址
  - ✓ 预测的分支目标地址

分支目标缓冲的结构和工作过程

5.4 动态分支预测

2.采用BTB后，在流水线各个阶段所进行的相关操作：



问题：解决预测错误或不命中的延迟。

3. 采用BTB技术时指令在各种情况下的延迟

指令在BTB中？	预测结果	实际的动作	延迟周期
是	成功	成功	0
是	成功	不成功	2
不是		成功	2
不是		不成功	0

表5.4.1 采用BTB技术时指令在各种情况下的延迟

5.4 动态分支预测

例 按表5.4.1计算分支转移总的延迟，根据下面的假设，计算分支目标缓冲的性能。

(1) 对于BTB中的指令，预测准确率90%

(2) 缓冲区命中率90%

(3) 不在BTB中分支转移成功的比例为60%

5.4 动态分支预测

解：根据表5.4.1的分类，性能计算包括4个部分：

(1) 在BTB中，预测成功，实际成功，此时的延迟为0。

(2) 在BTB中，预测成功，实际不成功，此时的延迟为：

$$BTB\text{命中率}\times\text{预测错误率}\times 2$$

$$=90\%\times 10\%\times 2=0.18\text{（时钟周期）}$$

(3) 不在BTB中，实际成功，此时的延迟为：

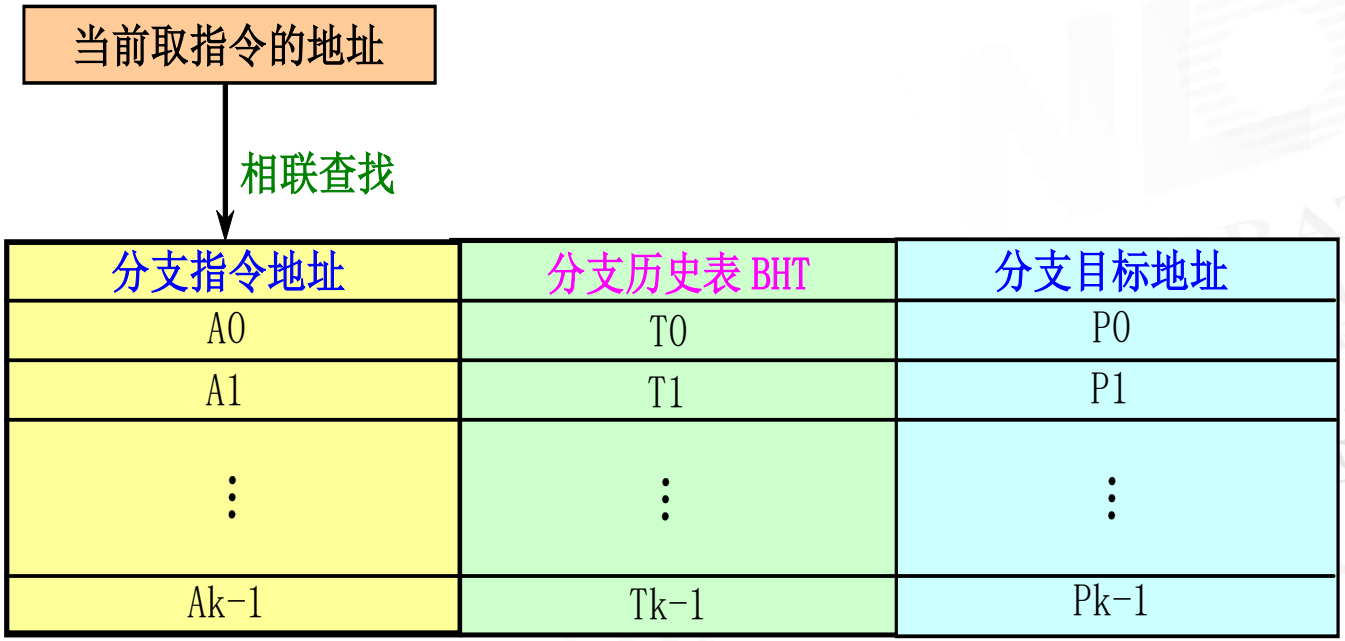
$$(1-BTB\text{命中率})\times\text{不在BTB中分支转移成功率}\times 2$$

$$=10\%\times 60\%\times 2=0.12\text{(时钟周期)}$$

(4) 不在BTB中，实际不成功，此时的延迟为0。

4. 对分支预测机制的一种改进

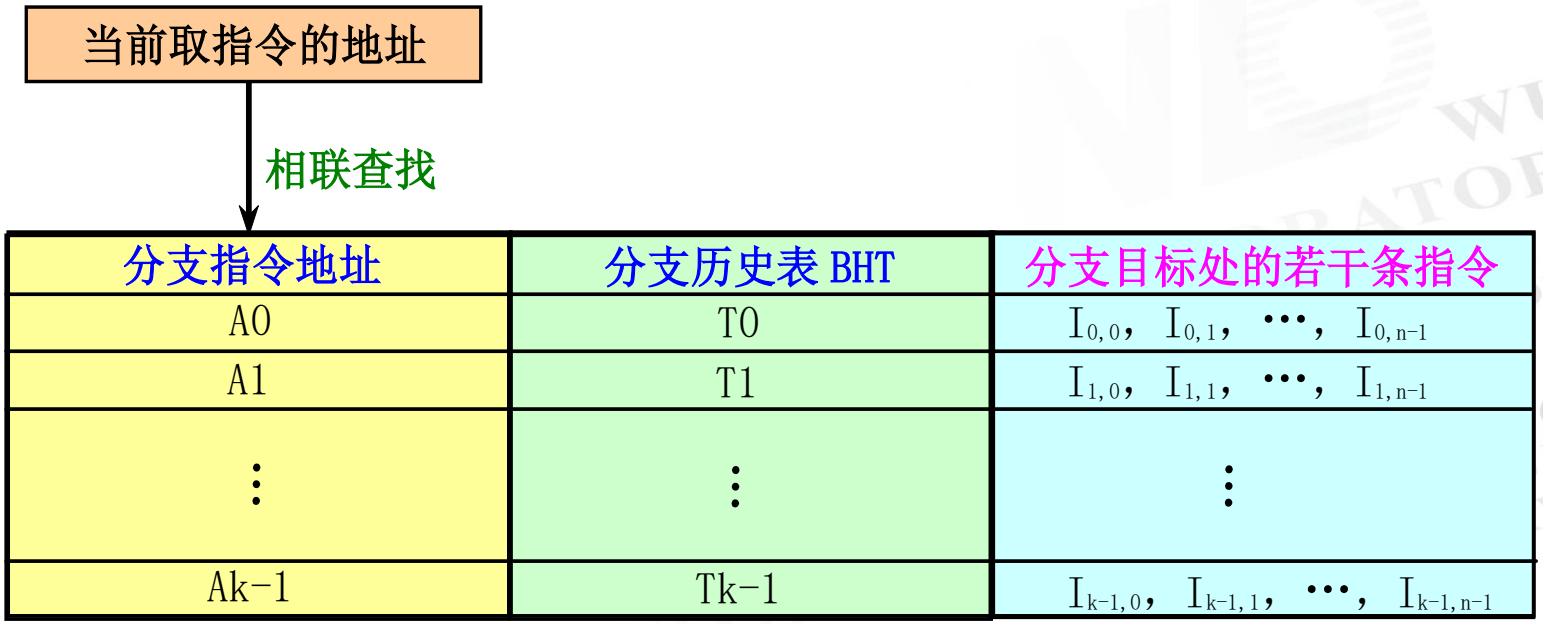
在分支目标缓冲器中增设一个至少是两位的“分支历史表”字段



分支指令地址	分支历史表 BHT	分支目标地址
A0	T0	P0
A1	T1	P1
⋮	⋮	⋮
Ak-1	Tk-1	Pk-1

5. 对分支预测机制的更进一步的改进

在表中对于每条分支指令都存放若干条分支目标处的指令，就形成了分支目标指令缓冲器，对于多流出处理器很有帮助



分支指令地址	分支历史表 BHT	分支目标处的若干条指令
A0	T0	$I_{0,0}$，$I_{0,1}$，…，$I_{0,n-1}$
A1	T1	$I_{1,0}$，$I_{1,1}$，…，$I_{1,n-1}$
⋮	⋮	⋮
Ak-1	Tk-1	$I_{k-1,0}$，$I_{k-1,1}$，…，$I_{k-1,n-1}$

5.4 动态分支预测

4. 对分支预测机制的一种改进

在缓冲区中不仅存入目的地址，而且还存入一个或多个目标指令。

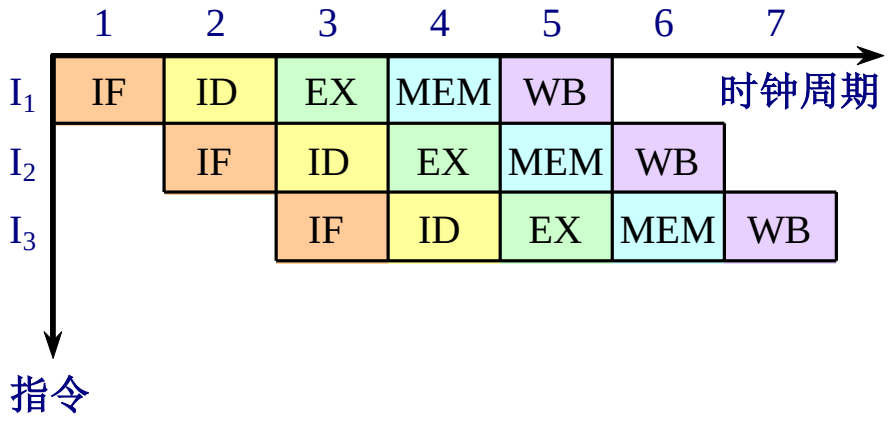
有两种潜在的好处：

- 在连续取指令之前，可以较长时间的访问缓冲区，这时的分支目的缓冲区较大。
- 对目的指令进行缓冲，构成称为分支目标指令缓冲（**branch folding**）的结构，它可使无条件分支的延迟达到零，甚至有的条件分支也可达到零延迟 。

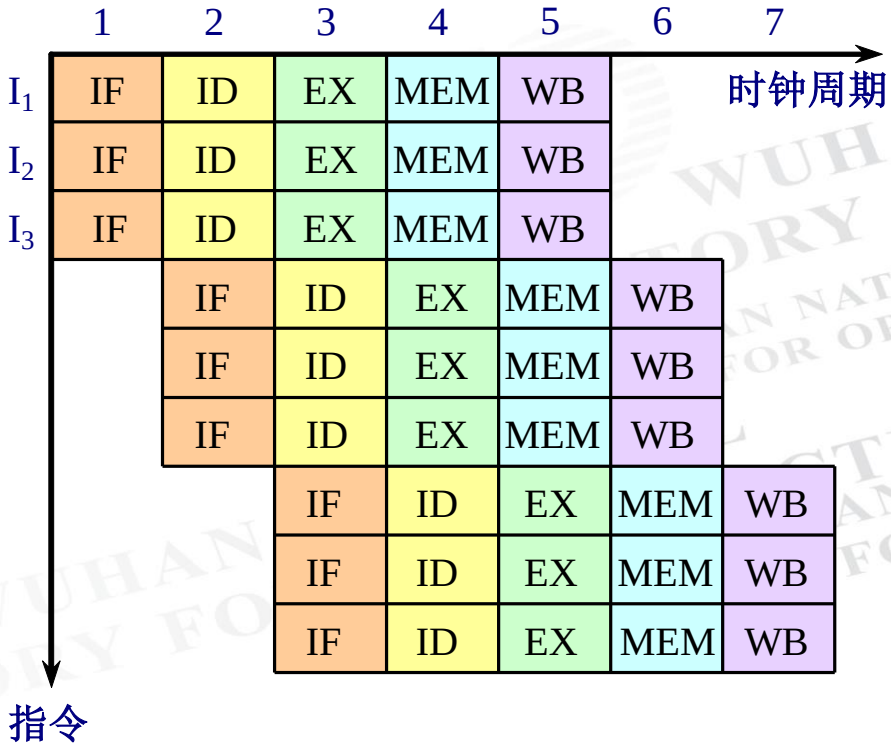
5.5 多指令流出技术

- 一个时钟周期内流出多条指令， CPI＜1
- 单流出和多流出处理机执行指令的时空图对比

单流出时空图



多流出时空图



5.5 多指令流出技术

多流出处理机有两种基本风格：

- 超标量（Superscalar）
  - ✓ 在每个时钟周期流出的指令条数不固定，依代码的具体情况而定。（有个上限）
  - ✓ 设这个上限为n，就称该处理机为n-流出。
  - ✓ 可以通过编译器进行静态调度，也可以使用硬件动态调度。
- 超长指令字VLIW（Very Long Instruction Word）
  - ✓ 在每个时钟周期流出的指令条数是固定的，这些指令构成一条长指令或者一个指令包。
  - ✓ 指令包中，指令之间的并行性是通过指令显式地表示出来的。
  - ✓ 指令调度是由编译器静态完成的。

5.5 多指令流出技术

基本的多流出技术、特点以及实例

技 术	流出结构	冲突检测	调 度	主要特点	处理机实例
超标量（静态）	动态	硬件	静态	按序执行	Sun UltraSPARCII/III
超标量（动态）	动态	硬件	动态	部分乱序执行	IBM Power2
超标量（猜测）	动态	硬件	带有前瞻的动态调度	带有前瞻的乱序执行	Pentium III/4，MIPS R10K，Alpha 21264，HP PA 8500，IBM RS64III
VLIW /LIW	静态	软件	静态	流出包之间没有冲突	Trimedia，i860
EPIC	主要是静态	主要是软件	主要是静态	相关性被编译器显式地标记出来	Itanium

EPIC: Explicitly Parallel Instruction Computer

5.5.3 超长指令字技术

1. 把能并行执行的多条指令组装成一条很长的指令；

   （100多位到几百位）

2. 设置多个功能部件；

3. 指令字被分割成一些字段，每个字段称为一个操作槽，直接独立地控制一个功能部件；

4. 在VLIW处理机中，在指令流出时不需要进行复杂的冲突检测，而是依靠编译器全部安排好了。

5.5.3 超长指令字技术

VLIW存在的一些问题

- 程序代码长度增加了
  - ✓ 提高并行性而进行的大量的循环展开；
  - ✓ 指令字中的操作槽并非总能填满。

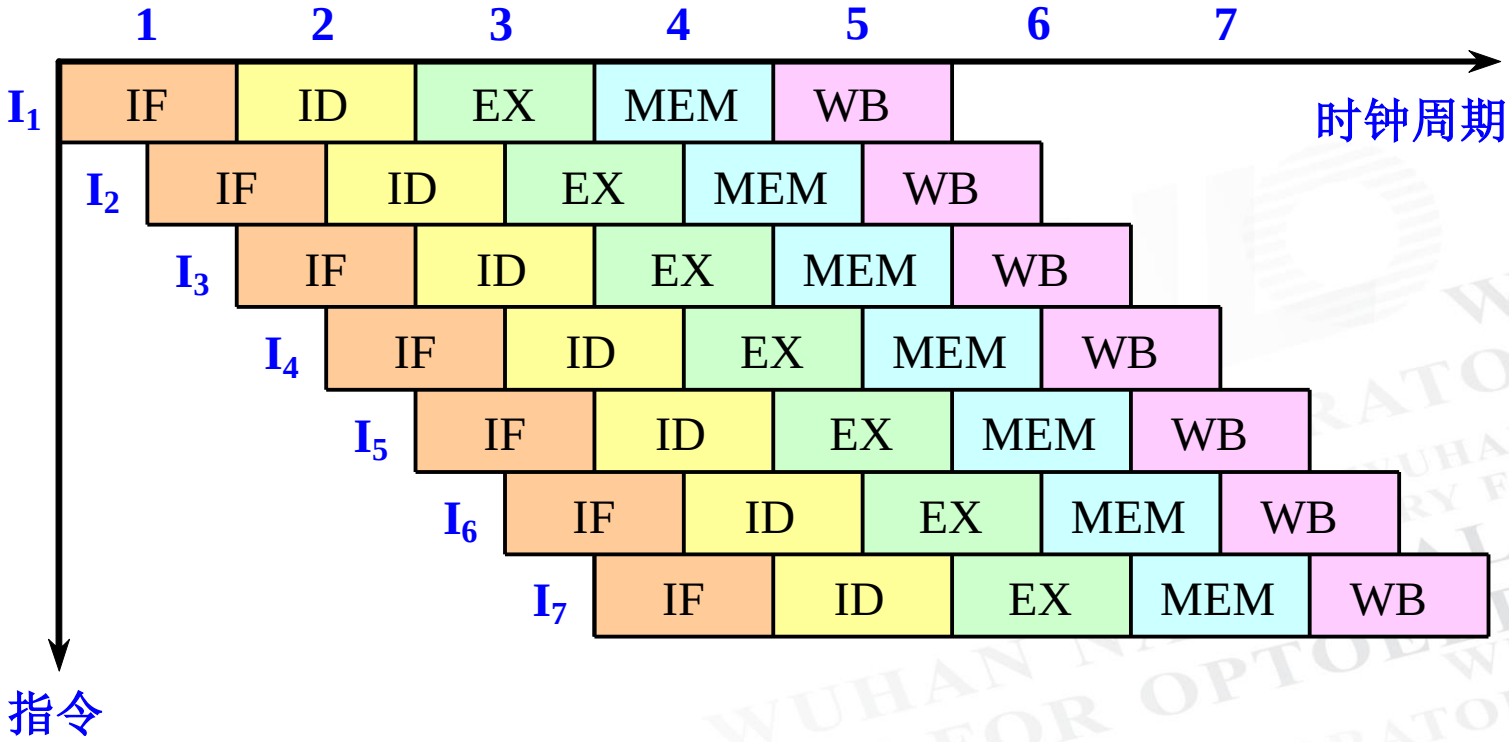
  解决：采用指令共享立即数字段的方法，或者采用指令压缩存储、调入Cache或译码时展开的方法。
- 采用了锁步机制

  任何一个操作部件出现停顿时，整个处理机都要停顿。
- 机器代码的不兼容性

5.5.5 超流水线处理机

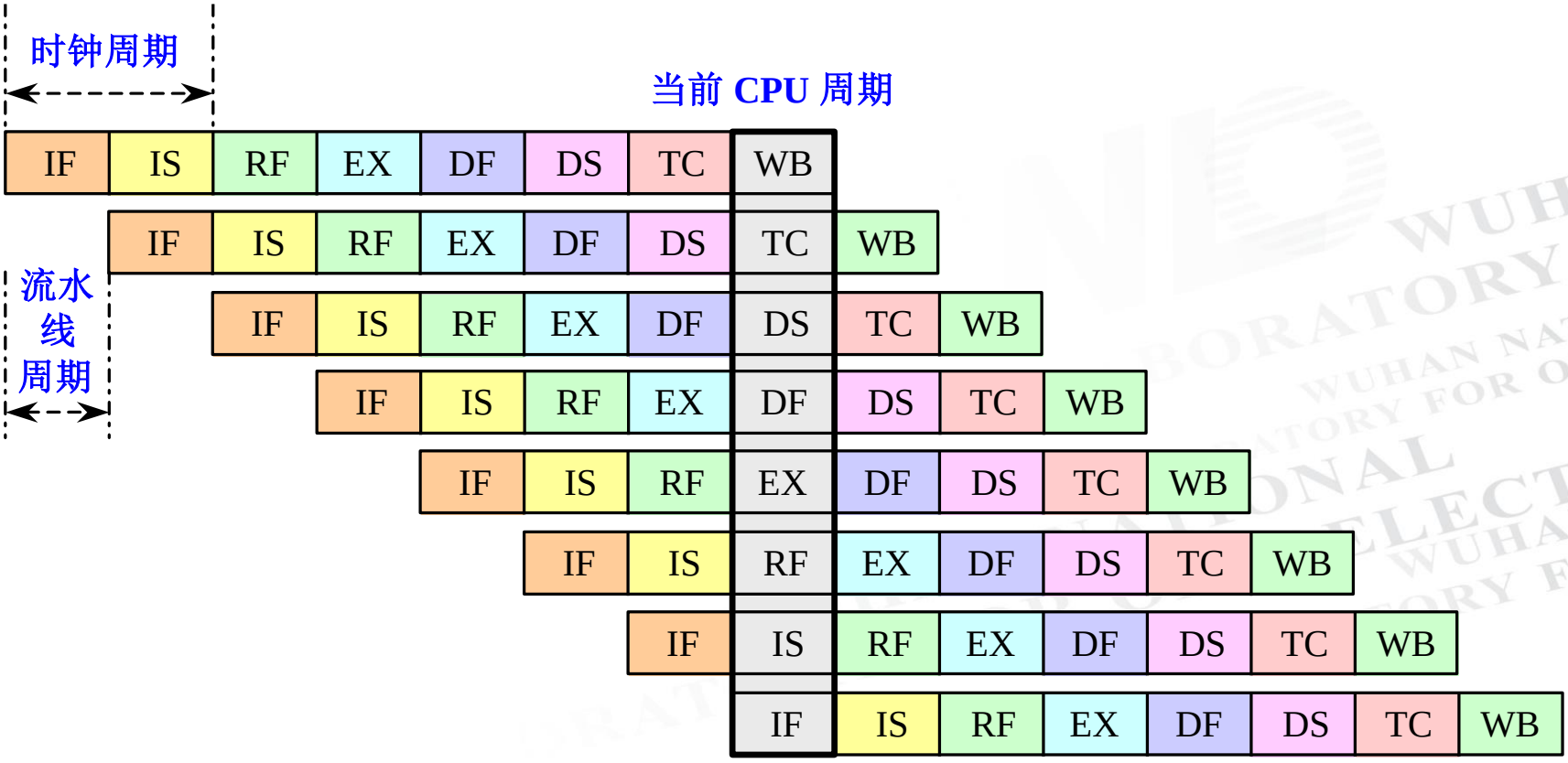
1. 将每个流水段进一步细分，这样在一个时钟周期内能够分时流出多条指令。这种处理机称为超流水线处理机。
2. 对于一台每个时钟周期能流出n条指令的超流水线计算机来说，这n条指令不是同时流出的，而是每隔1/n个时钟周期流出一条指令。
   - 实际上该超流水线计算机的流水线周期为1/n个时钟周期。
3. 一台每个时钟周期分时流出两条指令的超流水线计算机的时空图。

5.5.5 超流水线处理机



5.5.5 超流水线处理机

MIPS R4000指令流水线时空图



4.5 本章小结

本章在介绍指令级并行概念的基础上，重点讨论开发指令级并行的技术和方法，包括解决控制相关技术和多指令流出等内容。

(1) 分支预测缓冲和分支目标缓冲；

(2) 超长指令字技术。

作业： 5.8、5.9、5.11