



# [12] 发明专利说明书

[21] ZL 专利号 01131569.5

[45] 授权公告日 2004 年 3 月 17 日

[11] 授权公告号 CN 1142485C

[22] 申请日 2001. 11. 28 [21] 申请号 01131569. 5

[71] 专利权人 中国人民解放军国防科学技术大学  
地址 410073 湖南省长沙市砚瓦池正街 47 号  
中国人民解放军国防科学技术大学计算机学院

[72] 发明人 戴 葵 王志英 沈 立 王蓉晖  
王 蕾 张春元 王明仕 王 进  
审查员 陈 谦

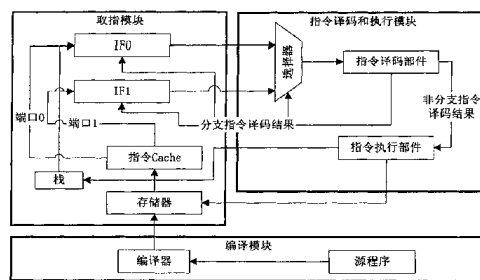
[74] 专利代理机构 湖南兆弘专利事务所  
代理人 赵 洪

权利要求书 3 页 说明书 10 页 附图 2 页

[54] 发明名称 流水线控制相关延迟消除方法

[57] 摘要

本发明公开了一种流水线控制相关延迟消除方法，其目的是在满足硬件实现简单、功耗低的前提下，高效消除流水线控制相关延迟，提高微处理器性能。技术方案是：由编译器确定分支指令所有可能的转移目标地址并插入预取指令；由两个取指令部件提前读入当前分支指令的所有后继指令，并由选择器根据对当前分支指令的译码结果选择一个取指令部件提供的指令进行译码和执行。预取指令有 fetch addr1, addr2、fetch addr、fetch stack 三条，它们对应不同的分支指令，由取指令部件执行。指令预取以基本块为单位。本发明硬件实现复杂度低、功耗低，控制相关延迟消除率高，无效预取指令数少。采用本发明设计的微处理器有极高性能价格比。



1. 一种流水线控制相关延迟消除方法，其总体逻辑结构包括编译模块、取指令模块、指令译码和执行模块，其特征在于由编译模块中的编译器负责确定分支指令所有可能的转移地址并根据分支指令的类型插入预取指令；取指令模块除了负责为指令译码和执行模块提供指令，还要进行指令预取，其功能通过两个取指令部件 IF<sub>0</sub> 和 IF<sub>1</sub> 完成；指令译码和执行模块中有一个选择器，负责根据当前分支指令的译码结果选择一个取指令部件提供的指令交给指令译码部件和指令执行部件进行译码和执行；其执行过程是：

- 1) 由编译模块中编译器确定分支指令所有可能的转移目标地址并插入预取指令；
- 2) 当程序开始运行时，取指令模块中一个取指令部件读入第一个基本块，另一个取指令部件空闲；对于程序中的每一个基本块：
  - (a) 负责读入该基本块的取指令部件依次读入基本块中的每一条指令，并判断它是否为预取指令；如果当前被读入的指令为预取指令，则将预取指令发送给另一个取指令部件，由它执行该预取指令；否则将指令发送给译码部件进行译码；
  - (b) 当基本块的最后一条指令即分支指令译码结束后，指令译码和执行模块中的选择器根据分支指令译码结果选择一个取指令部件中的基本块作为当前基本块的后继基本块：
    - I. 如果当前正在执行 IF<sub>0</sub> 中的指令，IF<sub>1</sub> 则执行预取指令，预取当前基本块的后继目标指令；当指令顺序执行，即结束译码的指令不是分支指令或转移条件为 False 的分支指令时，选

择  $IF_0$  中的指令进行译码；当执行分支指令时，即结束译码的指令是转移条件为 True 的分支指令时，选择  $IF_1$  中的指令进行译码；

II. 如果当前流水线正在执行  $IF_1$  中的指令，选择策略正好相反，即：如果指令依照程序顺序执行，即结束译码的指令不是分支指令或转移条件为 False 的分支指令时，选择  $IF_1$  中的指令进行译码；当执行分支指令时，即结束译码的指令是转移条件为 True 的分支指令时，选择  $IF_0$  中的指令进行译码。

2. 根据权利要求 1 所述的流水线控制相关延迟消除方法，其特征在于所述编译器插入预取指令的流程是：编译程序依次读出程序代码中的每一条指令，当遇到分支指令时，表示到达当前基本块的末尾，根据分支指令的类型在该分支指令所在基本块的第一条指令之后插入相应的预取指令。

3. 根据权利要求 1 所述的流水线控制相关延迟消除方法，其特征在于所述预取指令是对应不同的分支指令设计的，它们包括 `fetch addr1, addr2`、`fetch addr`、`fetch stack` 三种，不同的分支指令要使用不同的预取指令：

- (1) 条件分支指令：转移可能成功也可能失败，有两个后继基本块，应同时预取当前基本块的两个后继基本块；有两个可能的转移地址，一个保存在指令中，另一个是该指令之后那条指令的地址；此时在该分支指令所在基本块的第一条指令之后插入预取指令 `fetch addr1, addr2`，预取从地址 `addr1` 和 `addr2` 开始的两个基本块；`addr1` 由该分支指令译码得到，

addr2 是该指令之后的指令地址，其值为分支指令地址加指令长度；

(2) 直接无条件分支指令：转移总是成功，只有一个后继基本块，在编译时可以获得转移目标地址，预取目标基本块即可；只有一个可能的转移地址，保存在指令中；此时在该分支指令所在基本块的第一条指令之后插入预取指令 `fetch addr`，预取从地址 `addr` 开始的基本块；`addr` 由该分支指令译码得到；

(3) 间接无条件分支指令：转移总是成功，但由于转移目标地址保存在寄存器中，通常在编译时无法得到，对于这类分支指令，不进行处理；

(4) 过程返回语句：转移总是成功，只有一个后继基本块，这类语句通常出现在过程调用返回时，由于过程调用可能出现嵌套，本发明用一个栈来保存过程调用的返回地址—即预取地址，每次过程调用时将返回地址保存在栈顶单元，预取时从栈顶单元中获得预取地址；此时在该过程返回指令所在基本块的第一条指令之后插入预取指令 `fetch stack`，预取从栈顶单元地址开始的基本块。

4. 根据权利要求 1 所述的流水线控制相关延迟的消除方法，其特征在于所述指令预取以基本块为单位进行，除了条件分支指令转移失败时预取的一部分后继指令外，所预取的指令都将被执行。

## 流水线控制相关延迟消除方法

**技术领域：**本发明涉及微处理器设计中流水线控制相关延迟的消除方法，尤其是要求功耗低、硬件实现简单的嵌入式微处理器设计中流水线控制相关延迟的消除方法。

**背景技术：**目前，微处理器设计中消除流水线控制相关延迟的方法大致可以分为两类，即分支预测和延迟分支方法。分支预测方法主要利用程序运行时的局部性原理，根据分支指令执行结果的统计信息预测下次分支转移是否成功。分支预测的效果不仅取决于其准确性，而且与分支预测时的开销密切相关。流水线控制相关延迟取决于流水线的结构、预测的方法和预测错误后恢复所采取的策略。这种方法的不足之处在于，预测需要大量的硬件支持，例如预测表和预测错误后的恢复部件等，实现开销大、功耗高。延迟分支方法的主要思想是在控制相关暂停周期内执行与分支指令无关的指令，从而掩盖该暂停周期。这种方法的不足之处在于，无法填充所有的分支延迟槽，同时无法保证被调度的指令总是必须要执行的，如果不是必须的，那么性能得不到真正提高。

嵌入式微处理器主要应用于家电、手机、微控制器等领域，要求功耗低，以上方法要么无法满足其功耗低、复杂度低的要求，要么控制相关延迟消除率低，无法充分提高其性能。即使在通用微处理器设计中，这些方法也无法高效率地消除流水线中的控制相关延迟。

**发明内容：**本发明所要解决的技术问题是在满足嵌入式微处理器

硬件实现简单、功耗低的前提下，高效消除流水线控制相关延迟，提高微处理器性能。

本发明的技术方案是：由编译模块中的编译器确定分支指令所有可能的转移目标地址并插入预取指令；在取指令模块中设计两个取指令部件提前读入当前分支指令的所有可能的后继指令；在指令译码和执行模块中设计一个选择器，由选择器根据对当前分支指令的译码结果选择一个取指令部件提供的指令交给指令译码部件和指令执行部件进行译码和执行，从而消除流水线控制相关延迟。目前国内外尚无采用这种方法进行流水线控制相关延迟消除的报道。

本发明涉及到八个名词：流水线、分支指令、流水线相关、控制相关、控制相关延迟、指令预取、预取指令、基本块，它们的定义是：

- (1)流水线：将微处理器的指令执行过程分解成若干个子过程，每个子过程都可以有效地在其专用功能段上与其他子过程同时执行，这就是指令执行的流水技术。流水线是流水技术的具体实现。在不同的微处理器设计中，流水线具体被分为多少个阶段也不相同。一般流水线都包括5个阶段：取指、译码、执行、访存、写回。
- (2)分支指令：改变程序计数器值的所有指令统称为分支指令，它包括四类：条件转移指令、直接无条件转移指令、间接无条件转移指令和函数、过程返回指令（例如 return 语句）。
- (3)流水线相关：由于指令间的相互依赖关系造成的流水线停顿称为流水线相关。

- (4)控制相关：由于分支指令引起的流水线相关叫控制相关。
- (5)控制相关延迟：因为控制相关引起的流水线暂停时钟周期数就是控制相关延迟。
- (6)指令预取：在指令被执行之前提前将指令从存储器中取出的操作称为指令预取。
- (7)预取指令：负责完成预取的指令称为预取指令。
- (8)基本块：程序的基本组成单元，它只有一个入口（即基本块的第一条语句）和一个出口（即基本块的最后一条语句），一个基本块至少含有两条指令，基本块的出口为分支指令，一个程序总是可以划分成若干个基本块。

本发明的执行过程是：

1. 由编译模块中的编译器确定分支指令所有可能的转移目标地址并插入预取指令；
2. 当程序开始运行时，取指令模块中，一个取指令部件读入第一个基本块，另一个取指令部件空闲。

对于程序中的每一个基本块：

- (a) 负责读入该基本块的取指令部件依次读入基本块中的每一条指令，并判断它是否为预取指令。如果当前被读入的指令为预取指令，则将预取指令发送给另一个取指令部件，由它执行该预取指令；否则将指令发送给指令译码和执行模块中的译码部件进行译码。
- (b) 当基本块的最后一条指令即分支指令译码结束后，指令译码和执行模块中的选择器根据译码结果选择一个取指令部件中的基

本块作为当前基本块的后继基本块：

- 1) 假设两个取指令部件分别为  $IF_0$  和  $IF_1$ ，当前正在执行  $IF_0$  中的指令， $IF_1$  则执行预取指令，预取当前基本块的后继目标指令。当指令顺序执行，即结束译码的指令不是分支指令或转移条件为 `False` 的分支指令时，选择  $IF_0$  中的指令进行译码；当执行分支指令时，即结束译码的指令是转移条件为 `True` 的分支指令时，选择  $IF_1$  中的指令进行译码。
- 2) 如果当前流水线正在执行  $IF_1$  中的指令，选择策略正好相反，即：如果指令依照程序顺序执行，即结束译码的指令不是分支指令或转移条件为 `False` 的分支指令时，选择  $IF_1$  中的指令进行译码；当执行分支指令时，即结束译码的指令是转移条件为 `True` 的分支指令时，选择  $IF_0$  中的指令进行译码。

因此，在程序运行过程中，一个取指令部件负责为指令执行部件提供指令，另一个取指令部件负责完成指令预取，两个取指令部件并行工作，使得当分支指令译码结束时，所有可能的分支目标指令已经分别保存在两个取指令部件中。

与一般编译器相比，本发明的编译器增加了两项特殊功能：确定分支指令所有可能的转移地址并根据不同类型的分支指令向基本块中插入预取指令。编译器插入预取指令的流程是：编译程序依次读出程序代码中的每一条指令，当遇到分支指令时，表示到达当前基本块的末尾，根据分支指令的类型在该分支指令所在基本块的第一条指令之后插入相应的预取指令。

分支指令分为四类，本发明根据它们的不同情况设计了三条预取



指令对应不同的分支指令。三条预取指令是 `fetch addr1, addr2`、`fetch addr`、`fetch stack`。

- (1) 条件分支指令：转移可能成功也可能失败，有两个后继基本块，应同时预取当前基本块的两个后继基本块。有两个可能的转移地址，一个保存在指令中，另一个是该指令之后那条指令的地址。此时在该分支指令所在基本块的第一条指令之后插入预取指令 `fetch addr1, addr2`，预取从地址 `addr1` 和 `addr2` 开始的两个基本块。`addr1` 由该分支指令译码得到，`addr2` 是该指令之后的指令地址，其值为分支指令地址加指令长度（单位为字节）。
- (2) 直接无条件分支指令：转移总是成功，只有一个后继基本块，在编译时可以获得转移目标地址，预取目标基本块即可。只有一个可能的转移地址，保存在指令中。此时在该分支指令所在基本块的第一条指令之后插入预取指令 `fetch addr`，预取从地址 `addr` 开始的基本块。`addr` 由该分支指令译码得到。
- (3) 间接无条件分支指令：转移总是成功，但由于转移目标地址保存在寄存器中，通常在编译时无法得到，对于这类分支指令，不进行处理。
- (4) 过程返回语句：转移总是成功，只有一个后继基本块，这类语句通常出现在过程调用返回时，由于过程调用可能出现嵌套，本发明用一个栈来保存过程调用的返回地址（即预取地址），每次过程调用时将返回地址保存在栈顶单元，预取时从栈顶单元中获得预取地址。此时在该过程返回指令所在基本块

的第一条指令之后插入预取指令 `fetch stack`，预取从栈顶单元地址开始的基本块。

与其他指令不同，预取指令由取指令部件执行。不同的 RISC（精简指令集计算）指令集对应的预取指令的编码可能有所不同，但只要实现本发明相同的功能，都属于本发明保护范围。预取以基本块为单位进行，除了当条件分支指令转移失败时预取的一部分后继指令外，所预取的指令都将被执行。

如果在分支指令译码结束前所有可能的目标指令已被读入，就可以根据分支指令译码结果选择正确的后继指令进行译码和执行。采用系统性能评测协会（System Performance Evaluation Cooperative Consortium）提供的 SPECint95 基准程序组进行测试，在 FastDLX 模拟器（标准的 CPU 模拟器）中实现本发明时，如果不考虑间接无条件分支指令（这类指令在程序中所占的比率比较低），在分支指令译码结束后分支目标指令已被读入的机率达 99.3%；如果考虑间接无条件分支指令，在分支指令译码结束后分支目标指令已被读入的机率达 93%。

本发明具有以下优点：

- (1) 硬件实现复杂度低、功耗低。与分支预测技术相比，本发明省去了复杂的分支预测硬件和预测错误时的恢复硬件，仅仅使用了简单的控制逻辑部件，大大降低了硬件实现的难度和复杂度。
- (2) 控制相关延迟消除率高。本发明直接根据分支指令的译码结果选择分支目标指令，指令预取保证了当分支指令译码结束

后绝大多数指令都已被读入，从而消除了绝大多数控制相关延迟。

- (3) 预取以基本块为单位进行，在执行当前基本块的同时预取它的所有后继基本块，使得预取操作发出时间早，保证了有充足时间完成预取；同时，除了条件分支指令转移失败时预取的一部分后继指令外，所预取的指令都是将要被执行的，有效地减少了无效预取数。

本发明在满足嵌入式微处理器硬件实现简单、功耗低的前提下，实现了高效消除流水线控制相关延迟，提高微处理器性能的目的。本发明也可应用在通用微处理器设计中。

附图说明：

图 1 是本发明编译器插入预取指令的流程图；

图 2 是本发明总体逻辑结构图；

图 3 是一般微处理器非分支指令在 5 级流水线中的时空图；

图 4 是一般微处理器分支指令在 5 级流水线中的时空图；

图 5 是采用本发明后分支指令在 5 级流水线中的时空图；

图 6 是采用本发明针对 SPECint95 基准程序的测试结果；

图 7 是采用本发明和其它控制相关延迟消除方法的性能比较。

具体实施方式：

图 1 为本发明编译器插入预取指令的流程图。编译程序依次读出程序代码中的每一条指令，当遇到分支指令时，表示到达当前基本块的末尾，根据分支指令的类型在该分支指令所在基本块的第一条指令之后插入相应的预取指令，程序最后总是一条过程返回语句，因此编

译时总能遇到分支指令。

图2是本发明的总体逻辑结构图。它由编译模块、取指令模块、指令译码和执行模块组成：

编译模块主要负责确定分支指令所有可能的转移地址并根据分支指令的类型插入预取指令。编译器依次读取源程序中的每一条指令，如果该指令为分支指令，则根据分支指令的类型向它所在基本块的第一条指令之后插入相应的预取指令。具体方法为：对于条件分支指令，有两个后继基本块，相应的预取地址有两个，插入预取指令 `fetch addr1, addr2`，其中 `addr1` 由该分支指令译码得到，`addr2` 是该指令之后的指令地址，其值为分支指令地址加指令长度（单位为字节）；对于直接无条件分支指令，只有一个后继基本块，相应的预取地址有一个，插入预取指令 `fetch addr`，其中 `addr` 由指令译码得到；过程返回语句，有一个后继基本块，相应的预取地址有一个，保存在栈顶单元中，插入预取指令 `fetch stack`。编译后的程序代码保存在存储器中。

取指令模块主要负责为指令译码和执行模块提供指令，并进行指令预取，其功能通过两个取指令部件完成。取指令部件 `IF0` 通过端口 0 从指令 Cache 中读指令，取指令部件 `IF1` 则通过端口 1 从指令 Cache 中读指令。在程序开始运行时，选择 `IF0` 提供的指令进行译码和执行，`IF1` 空闲，当 `IF0` 读入预取指令后，将其转发给 `IF1`，由 `IF1` 完成预取；在程序运行过程中，哪个取指令部件进行取指令，哪个取指令部件进行指令预取应根据分支指令的译码结果确定：如果 `IF0` 负责取指令，`IF1` 负责指令预取，结束译码的指令不是分支指令或者是转移失败的分支指令，那么二者操作不变，否则 `IF1` 负责取指令，`IF0` 负责指令预

取；如果  $IF_1$  负责取指令， $IF_0$  负责指令预取，结束译码的指令不是分支指令或者是转移失败的分支指令，那么二者操作不变，否则  $IF_0$  负责取指令， $IF_1$  负责指令预取。

指令译码和执行模块主要负责指令的译码和执行，被译码和执行的指令由取指令模块中的一个取指令部件提供，其中分支指令的译码结果被送往选择器和两个取指令部件，非分支指令的译码结果被送往指令执行部件进行执行。选择器负责根据分支指令的译码结果选择一个取指令部件提供的指令进行译码和执行，当程序开始运行时，它选择  $IF_0$  中的指令译码和执行，在程序运行过程中，它根据分支指令的译码结果进行选择：如果正在使用  $IF_0$  中的指令，结束译码的指令不是分支指令或者是转移失败的分支指令，那么继续使用  $IF_0$  的指令，否则使用  $IF_1$  中的指令；如果正在使用  $IF_1$  中的指令，结束译码的指令不是分支指令或者是转移失败的分支指令，那么继续使用  $IF_1$  中的指令，否则使用  $IF_0$  中的指令。在指令执行过程中，如果发生过程调用，则将过程返回地址保存在栈顶单元中以便进行预取。

如图 3，假设流水线分为 5 个阶段：取指（IF）、译码（ID）、执行（EX）、访存（MEM）和写回（WB），指令  $p$  为在指令  $i$  之后执行的第一条指令，指令  $p+1$  为在指令  $i$  之后执行的第二条指令，依次类推。由于  $i$  不是分支指令，那么在对它译码的同时，取指令部件将读取指令  $p$ ，因此指令  $i$  的 ID 阶段与指令  $p$  的 IF 阶段重合，指令  $i$  的 EX 阶段与指令  $p$  的 ID 阶段重合，此时没有控制相关延迟。

如图 4，指令  $i$  为分支指令，在与图 3 相同的流水线中，只有在指令  $i$  译码结束后，才能确定指令  $p$  的地址，指令  $i$  的 EX 阶段与指令  $p$

的 IF 阶段重叠，此时流水线有一个时钟周期的控制相关延迟。

如图 5，假设指令 I 为分支指令，指令  $i_1$  是分支转移失败时执行的指令，指令  $i_2$  是分支转移成功时执行的指令，指令 q 为在分支指令后执行的第二条指令，指令 q+1 为在分支指令后执行的第三条指令，依此类推。采用本发明后，在指令 i 译码的同时，两个取指令部件将分别读取指令  $i_1$  和  $i_2$ ，在指令 i 译码结束后即可根据译码结果选择指令  $i_1$  或  $i_2$  进行译码，指令 i 的 EX 阶段与指令  $i_1$  或  $i_2$  的 ID 阶段重合，消除了原有的控制相关延迟。

本发明已成功地在银河 TS-1 嵌入式微处理器 IP 核的流水线中实现，能有效消除流水线控制相关延迟，图 6 为采用 SPECint95 基准程序组测试得到的结果。图中，纵轴表示 SPECint95 基准程序组中的每个基准程序，横轴表示控制相关延迟消除率，gcc 为 88.80%，jpeg 为 83.32%，compress 为 88.55%，perl 为 88.69%，m88ksim 为 92.99%，li 为 85.09%，vertex 为 87.47%，go 为 86.16%，流水线控制相关延迟的平均消除率达到 87.8%。

图 7 出了采用不同的控制相关延迟消除方法所得到的性能，其中 CPI (cycle per instruction) 表示执行一条指令所需的平均时钟周期数，条件分支延迟、无条件分支延迟和平均分支延迟的单位都是时钟周期 (cycle)。其中，流水线暂停代表没有采用任何分支延迟消除技术的情况，双取指部件表示采用本发明的情况。采用本发明后，条件分支延迟降低到 0.05，无条件分支延迟降低到 0.09，平均分支延迟降低到 0.06，有效 CPI 值降低到 1.01，均远远低于采用其他方法得到的结果。

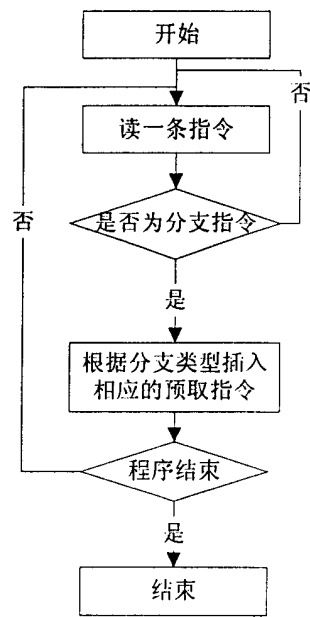


图 1

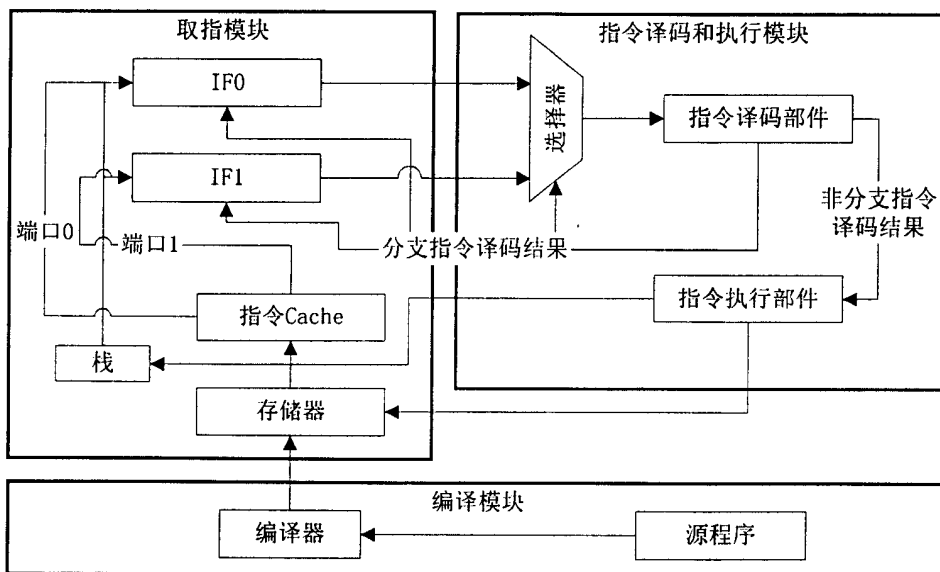


图 2

指令 i	IF	ID	EX	MEM	WB				
指令 p		IF	ID	EX	MEM	WB			
指令 p+1			IF	ID	EX	MEM	WB		
指令 p+2				IF	ID	EX	MEM	WB	
指令 p+3					IF	ID	EX	MEM	WB

图 3

指令 i	IF	ID	EX	MEM	WB					
暂停										
指令 p			IF	ID	EX	MEM	WB			
指令 p+1				IF	ID	EX	MEM	WB		
指令 p+2					IF	ID	EX	MEM	WB	
指令 p+3						IF	ID	EX	MEM	WB

图 4

指令 i	IF	ID	EX	MEM	WB					
指令 i <sub>1</sub>		IF	ID	EX	MEM	WB				
指令 i <sub>2</sub>		IF	ID	EX	MEM	WB				
指令 q				IF	ID	EX	MEM	WB		
指令 q+1					IF	ID	EX	MEM	WB	
指令 q+2						IF	ID	EX	MEM	WB

图 5

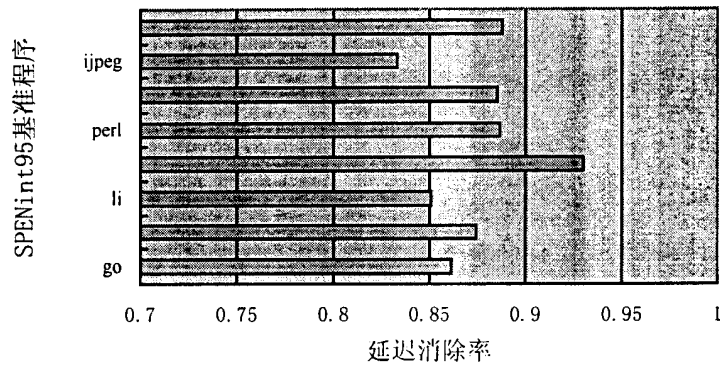


图 6

调度策略	条件分支延迟	无条件分支延迟	平均分支延迟	有效 CPI
暂停流水线	1.00	1.00	1.00	1.17
预测转移成功	1.00	1.00	1.00	1.17
预测转移不成功	0.62	0.70	0.69	1.12
延迟分支	0.25	0.00	0.21	1.04
双取指部件	0.05	0.09	0.06	1.01

图 7