(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103218206 A
(43)申请公布日 2013.07.24

(21)申请号 201210015287.6

(22)申请日 2012.01.18

(71)申请人 上海算芯微电子有限公司
地址 201203 上海市浦东新区盛夏路 560 号 2 幢 1004-1005 室

(72)发明人 沙力 兰军强 朱磊

(74)专利代理机构 北京戈程知识产权代理有限公司 11314
代理人 程伟 孙向民

(51)Int.Cl.
G06F 9/38(2006.01)

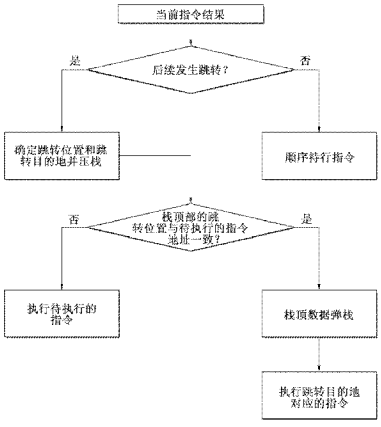
权利要求书1页 说明书6页 附图2页

(54)发明名称

指令分支的预跳转方法和系统

(57)摘要

一种指令分支的预跳转方法和系统，该方法包括：当根据当前指令判断出后续指令中包含跳转指令时，根据当前指令的结果，判断所述跳转指令是否执行跳转；若判断结果为执行跳转，则确定跳转指令的跳转位置和跳转目的地，并将所述跳转位置和所述跳转目的地压入栈，若判断结果为不执行跳转，则按原有顺序执行之后的指令；当所述栈顶部的跳转位置与待执行的指令地址一致时，则弹出所述栈顶部的跳转位置和跳转目的地，以弹出的所述跳转目的地作为下一条指令的地址，当所述栈顶部的跳转位置与待执行的指令地址不一致时，则以待执行的指令地址作为下一条指令的地址；根据所述下一条指令的地址读取指令并执行。



CN 103218206 A

1. 一种指令分支的预跳转方法，该方法包括：

步骤 S11：当根据当前指令判断出后续指令中包含跳转指令时，根据当前指令的结果，判断所述跳转指令是否执行跳转；

步骤 S12：若判断结果为执行跳转，则确定跳转指令的跳转位置和跳转目的地，并将所述跳转位置和所述跳转目的地压入栈，若判断结果为不执行跳转，则按原有顺序执行之后的指令；

步骤 S13：当所述栈顶部的跳转位置与待执行的指令地址一致时，则弹出所述栈顶部的跳转位置和跳转目的地，以弹出的所述跳转目的地作为下一条指令的地址，当所述栈顶部的跳转位置与待执行的指令地址不一致时，则以待执行的指令地址作为下一条指令的地址；

步骤 S14，根据所述下一条指令的地址读取指令并执行。

2. 根据权利要求 1 所述的指令分支的预跳转方法，其中，步骤 S12 通过预跳转指令来实现，该预跳转指令基于条件生成指令的结果进行操作，若条件生成指令的结果指示为执行跳转，则确定跳转指令的跳转位置和跳转目的地，并将所述跳转位置和所述跳转目的地压入栈，若判断结果为不执行跳转，则顺序执行之后的指令。

3. 根据权利要求 2 所述的指令分支的预跳转方法，其中所述与跳转指令包括条件字段、跳转位置字段和跳转目的地字段。

4. 根据权利要求 1 所述的指令分支的预跳转方法，其中待执行的指令的地址由程序计数器指示。

5. 一种指令分支的预跳转系统，该系统包括：

用于存储预跳转数据的栈；

预跳转模块，其输入端连接指令处理模块的输出，其输入端连接所述栈的输入，当指令处理模块的输出结果指示后面的跳转指令将执行跳转时，将所述跳转指令的跳转位置和跳转目的地作为跳转数据压入所述栈；

指令地址比较模块，其比较所述栈的顶部的跳转位置和待执行的指令地址，当所述栈顶部的跳转位置与待执行的指令地址一致时，则弹出所述栈顶部的跳转位置和跳转目的地作为指令地址比较模块的输出，当所述栈顶部的跳转位置与待执行的指令地址不一致时，则以所述待执行的指令地址作为指令地址比较模块的输出；

指令读取模块，根据指令地址比较模块的输出读取指令；

指令处理模块，对指令读取模块读取的指令进行处理，输出结果。

6. 根据权利要求 5 所述的指令分支的预跳转系统，其中由程序计数器指示所述待执行的指令地址。

7. 根据权利要求 5 所述的指令分支的预跳转系统，其中指令处理器模块进行指令译码、读操作数、运算操作。

指令分支的预跳转方法和系统

技术领域

[0001] 本发明涉及一种指令分支的预跳转方法和系统，特别涉及一种基于栈模式的预跳转方法和系统。

背景技术

[0002] 在微处理器的运行过程中，跳转指令的出现十分频繁，因此对跳转指令的处理方式显著的影响着以流水线方式工作的微处理器的性能。跳转指令的一般功能是：如果满足某一条件，则跳转到某一目的地，而该条件通常是由该跳转指令之前的用于生成跳转发生的条件的条件生成指令（例如比较指令 CMP）产生的。由于流水线的工作方式是：在流水线的起点读取指令，在流水线的终点产生指令的结果，因此，如果按照指令的自然顺序依次执行指令，就会出现这样的问题：当条件生成指令在流水线的最后一级 S_n 产生结果时，跳转指令处于倒数第二级 S_{n-1}，跳转指令之后的按自然顺序排列的各条指令也已经被读出并依次处于流水线的 S_1-S_{n-2} 级，此时，如果跳转指令根据条件生成指令的结果判断出跳转条件已经满足，需要发生跳转，也就是说不再按照自然顺序执行指令，而是要执行跳转指令的跳转目的地的指令，这样一来，之前读出的各条指令就必须取消执行，造成了流水线资源的浪费。

[0003] 目前，解决这一问题的方法主要包括延迟槽技术和跳转预测技术。

[0004] 延迟槽技术的原理是，将跳转指令提前，将跳转指令之前与跳转无关的一定数量（即延迟槽长度，一般为 2 或 3 的固定长度）的指令放在跳转指令之后，这样，无论跳转指令判断的结果如何，被转移到跳转指令之后的与跳转无关的指令都可以顺序执行。延迟槽技术的主要问题是，使得指令排序不自然，并且，这种定长延迟槽十分死板，当跳转指令附近缺少合适的可调换的指令时，只能通过填入空操作 NOP 来满足延迟槽长度，事实上，大多数时候只能通过 NOP 来构成延迟槽，这同样造成流水线资源的浪费。

[0005] 跳转预测技术的原理是，根据很多历史记录猜测跳转指令后应该执行哪一条指令，并按照猜测的结果读取指令，如果猜测错误，则将已经读取的错误指令全部取消。这种跳转预测技术的问题是，其逻辑十分复杂，硬件成本显著，且某些已经读出甚至已经执行的指令是很难干净的取消的，只能被迫推迟执行，因此大大影响了微处理器的性能。

发明内容

[0006] 本发明提出了一种指令分支的预跳转方法和系统，其克服了现有技术中的上述问题，不受定长延迟槽的限制，也无需引入取消执行，提高了编译灵活性和流水线效率，降低了设计复杂度。根据本发明的方法和系统基于栈模式，特别有利于循环嵌套语句的优化。

[0007] 根据本发明的一方面，提出了一种指令分支的预跳转方法，该方法包括：

[0008] 步骤 S11：当根据当前指令判断出后续指令中包含跳转指令时，根据当前指令的结果，判断所述跳转指令是否执行跳转；

[0009] 步骤 S12：若判断结果为执行跳转，则确定跳转指令的跳转位置和跳转目的地，并

将所述跳转位置和所述跳转目的地压入栈，若判断结果为不执行跳转，则按原有顺序执行之后的指令；

[0010] 步骤 S13：当所述栈顶部的跳转位置与待执行的指令地址一致时，则弹出所述栈顶部的跳转位置和跳转目的地，以弹出的所述跳转目的地作为下一条指令的地址，当所述栈顶部的跳转位置与待执行的指令地址不一致时，则以待执行的指令地址作为下一条指令的地址；

[0011] 步骤 S14，根据所述下一条指令的地址读取指令并执行。

[0012] 优选地，步骤 S12 通过预跳转指令来实现，该预跳转指令基于条件生成指令的结果进行操作，若条件生成指令的结果指示为执行跳转，则确定跳转指令的跳转位置和跳转目的地，并将所述跳转位置和所述跳转目的地压入栈，若判断结果为不执行跳转，则顺序执行之后的指令。

[0013] 优选地，所述与跳转指令包括条件字段、跳转位置字段和跳转目的地字段。

[0014] 优选地，待执行的指令的地址由程序计数器指示。

[0015] 根据本发明的另一方面，提出了一种指令分支的预跳转系统，该系统包括：

[0016] 用于存储预跳转数据的栈；

[0017] 预跳转模块，其输入端连接指令处理模块的输出，其输入端连接所述栈的输入，当指令处理模块的输出结果指示后面的跳转指令将执行跳转时，将所述跳转指令的跳转位置和跳转目的地作为跳转数据压入所述栈；

[0018] 指令地址比较模块，其比较所述栈的顶部的跳转位置和待执行的指令地址，当所述栈顶部的跳转位置与待执行的指令地址一致时，则弹出所述栈顶部的跳转位置和跳转目的地作为指令地址比较模块的输出，当所述栈顶部的跳转位置与待执行的指令地址不一致时，则以所述待执行的指令地址作为指令地址比较模块的输出；

[0019] 指令读取模块，根据指令地址比较模块的输出读取指令；

[0020] 指令处理模块，对指令读取模块读取的指令进行处理，输出结果。

[0021] 优选地，由程序计数器指示所述待执行的指令地址。

[0022] 优选地，指令处理器模块进行指令译码、读操作数、运算操作。

附图说明

[0023] 图 1 是根据本发明的一个实施例的一种指令分支的预跳转方法的流程图；

[0024] 图 2 是一条预跳转指令的示意性结构图；

[0025] 图 3 是根据本发明的一个实施例的一种指令分支的预跳转系统的结构图；

[0026] 图 4 是本发明的预跳转系统的一个优选实施例。

具体实施方式

[0027] 本发明提出了一种指令分支的预跳转方法及系统，其基本原理是：在判断出后面将要发生跳转时，就确定将要发生的跳转的跳转位置和跳转目的地，从而在指令进行到该跳转位置时完成跳转操作。

[0028] 事实上，程序中的任何一个有条件的跳转都是在条件生成时就能够知道后面将会有跳转发生的。例如对于标准的“for”循环，在循环起点就可以知道，在后面肯定会面临是

否跳转的判断，甚至根据循环变量的当前数值，就能够知道未来要不要跳转，根据本发明的方法在得知未来要跳转的时候就指明在哪里跳转，要跳到哪里去，在指令进行到之前预计跳转的位置时，就执行之前预计的跳转操作。

[0029] 本发明的预跳转方法基于栈模式，其基本原理是可预先判断后面将要发生的若干条跳转，将这些跳转按顺序压栈，当到达各个跳转位置时将这些跳转逐次弹栈，这种栈模式（即，先进后出的模式）与嵌套循环的模式是匹配的，因此特别适用于多重嵌套循环。

[0030] 图 1 是根据本发明的一个实施例的一种指令分支的预跳转方法的流程图，其包括：

[0031] 步骤 S11：当根据当前指令判断出后续指令中包含跳转指令时，根据当前指令的结果，确定所述跳转指令是否执行跳转；

[0032] 步骤 S12：若判断结果为执行跳转，则确定跳转指令的跳转位置和跳转目的地，并将所述跳转位置和所述跳转目的地压入栈，若判断结果为不执行跳转，则按原有顺序执行之后的指令；

[0033] 步骤 S13：当所述栈顶部的跳转位置与待执行的指令地址一致时，则弹出所述栈顶部的跳转位置和跳转目的地，以弹出的所述跳转目的地作为下一条指令的地址，当所述栈顶部的跳转位置与待执行的指令地址不一致时，则以待执行的指令地址作为下一条指令的地址；

[0034] 步骤 S14，根据所述下一条指令的地址读取指令并执行。

[0035] 其中，步骤 S12 可通过预跳转指令 PJ 来实现，该预跳转指令 PJ 的功能是：基于条件生成指令的结果进行操作，若条件生成指令的结果指示为执行跳转，则确定跳转指令的跳转位置和跳转目的地，并将所述跳转位置和所述跳转目的地压入栈，若判断结果为不执行跳转，则顺序执行之后的指令。

[0036] 图 2 是一条预跳转指令的示意性结构图。其中该预跳转指令主要包括条件字段 #C、跳转位置字段 B 和跳转目的地字段 A。条件字段 #C 规定了跳转发生的条件，#C 通常为常数，可对应于条件生成指令产生的标示位 FLAG。跳转位置字段 B 指示了在所执行的指令中，“跳转”这一动作所在的位置，即指明“在哪里跳转”的问题，该位置与执行“跳转”操作的指令的指令地址相关联。跳转目的地字段 A 指示了执行“跳转”操作时，指令的分支地址，即指明“跳转到哪里去”的问题。

[0037] 步骤 S13 可以通过将栈顶部的跳转位置与程序计数器 PC 指示的指令地址相比较来实现，其中程序技术器 PC 指示的地址就是待执行的指令的地址，若栈顶的跳转位置与 PC 指示的指令地址一致，则标志着指令已经运行到了应该发生跳转的位置，则弹出栈顶部的跳转位置和跳转目的地，该跳转目的地就是执行跳转时应“跳转”到的指令分支地址，按照这个指令分支地址读取指令，就完成了此次跳转。

[0038] 由于栈的工作机制是“先进后出”，因此在多层嵌套循环指令中，外层循环的跳转先进栈，内层循环的跳转后进栈，此后指令必然先运行至内层循环的跳转位置，从而使顶部的内层循环跳转出栈。当指令运行到外层循环的跳转位置时，再使外层循环跳转出栈，从而完成整个多层嵌套循环。

[0039] 下面以标准的二重嵌套循环为例详细描述本发明的方法。

[0040] 以下是标准的二重循环的 c 语言代码：

[0041]

```
for(i=0; i<4; i++) {
    for(j=0; j<2; j++) {
    *p++=q;
    }
}
```

[0042] 其中 *p++ = q ;是示意性的循环体。

[0043] 按照传统的定长（假定为 3）延迟槽的方法进行编译，得到如下的汇编指令，总执行指令数为 65 条（内循环 5x2 条，外循环 16x4 条）：

```
[0044]  MOV    i,0        //i = 0
[0045]  LOOPi :           // 外循环起点
[0046]  MOV    j,0        //j = 0
[0047]  LOOPj :           // 内循环起点
[0048]  CMP    J,1        // 比较 j 和 2-1
[0049]  JLT    LOOPj      // 在 j = 0 时跳转
[0050]  ST     q,[p],1    // 延迟槽 1 :*p++ = q
[0051]  ADD    j,1        // 延迟槽 2 :j++
[0052]  NOP               // 延迟槽 3 :
[0053]                    // 内循环终点
[0054]  CMP    i,3        // 比较 i 和 4-1
[0055]  JLT    LOOPi      // 在 i = 0,1,2 时循环
[0056]  ADD    i,1        // 延迟槽 1 :i++
[0057]  NOP               // 延迟槽 2
[0058]  NOP               // 延迟槽 3
[0059]                    // 外循环终点
```

[0060] 由于在采用了长度为 3 的定长延迟槽，在内循环跳转指令 JLT 之后，将循环体指令 ST 和循环变量 j 加 1 指令 ADD 作为延迟槽 1 和延迟槽 2，由于 JLT 附近没有其他适合作为延迟槽的指令，因此插入 NOP 指令作为延迟槽 3，在外循环中，则需要两条 NOP 指令作为延迟槽 2 和延迟槽 3，造成了指令的冗长和流水线资源的浪费。

[0061] 相对比地，根据本发明所述的方法，基于上述 c 语言程序生成的指令如下 :其中总执行指令条数为 49 条（内循环 4x2 条，外循环 12x4 条）：

```
[0062]  MOV    i,0              //i = 0
[0063]  LOOPi :                 // 外循环起点
[0064]  CMP    i,3              // 比较 i 和 4-1
[0065]  PJ     [LT]ENDi,LOOPi   // 在 i = 0,1,2 时循环
[0066]  MOV    j,0              //j = 0 ;
[0067]  LOOPj :                 // 内循环起点
[0068]  CMP    j,1              // 比较 j 和 2-1
[0069]  PJ     [LT]ENDj,LOOPj   // 在 j = 0 时循环
```

[0070] ST q,[p],1 //*p++ = q ;

[0071] ADD j,1 //j++ ;

[0072] ENDj : // 内循环终点

[0073] ADD i,1 //i++ ;

[0074] ENDj : // 外循环终点

[0075] 在外循环起点处，由于 i = 0，CMP 指令判断出在外循环终点处将发生跳转，跳转到外循环起点，这里 CMP 指令起到条件生成指令的作用，“外循环终点”就是跳转位置，“外循环起点”就是跳转目的地，因此，在 CMP 指令后插入一条预跳转指令 PJ，其作用是当满足条件 [LT] 时（这里是指 i = 0,1,2），将跳转位置“外循环终点”ENDi，和跳转目的地“外循环起点”LOOPi 压栈。

[0076] 在内循环起点处，由于 j = 0，CMP 指令判断出在内循环终点处将发生跳转，跳转到内循环起点，因此，在 CMP 指令后插入一条预跳转指令 PJ，其作用是当满足条件 [LT] 时（这里是指 j = 0），将跳转位置“内循环终点”ENDj，和跳转目的地外循环起点 LOOPj 压栈。

[0077] 与此同时，程序计数器 PC 实时地将所指示的待执行的指令地址与栈顶点的跳转位置进行比较，当进行到内循环终点 ENGj 时，PC 指示待执行的指令地址对应于 ENGj，而栈顶的跳转位置也对应于 ENGj，二者一致，因此将 ENGj 和 LOOPj 弹栈，指令处理系统读取该跳转目的地，即指令分支地址 LOOPj，指令序列跳转到内循环起点，完成了内循环中的一次跳转。

[0078] 此时，由于内循环跳转位置 ENGj 和内循环跳转目的地 LOOPj 已经弹栈，栈顶存放的数据为外循环跳转位置 ENGi 和外循环跳转目的地 LOOPi。

[0079] 接下来，由于 j = 1，不再满足预跳转指令的条件，因此不再进行压栈操作，指令顺序进行至外循环终点，此时，程序计数器 PC 指示的指令地址为外循环终点 ENDi，与栈顶的跳转位置 ENGi 一致，指令处理系统读取该跳转目的地，即指令分支地址 LOOPi，指令序列跳转到外循环起点，完成了外循环中的一次跳转。

[0080] 接下来，在 i = 1,2 时重复上述外循环过程，直到 i = 3，不再满足预跳转指令 PJ 的条件，因此不发生压栈操作，指令顺序进行，直到整个嵌套循环结束。

[0081] 编译器可以根据对跳转的预测，自由的选择预跳转指令的位置。这种基于栈的机制非常利于嵌套循环这一常见的循环模式的优化，从而大大提高指令的效率。根据本发明的方法相当于使用了可选长度的延迟槽，相对于固定长度的延迟槽来说，给程序带来了极大的灵活性，跳转条件判断和实际分支地点不必相邻，且完全无需引入取消执行而增加设计难度。在不使用传统的跳转预测技术的情况下，本发明的预跳转方法就能够获得很高的效率，同时，本发明的预跳转技术与传统的跳转预测技术并不矛盾，根据应用的需要，也可以引入取消执行的机制作进一步的优化。

[0082] 图 3 是根据本发明的一个实施例的一种指令分支的预跳转系统的结构图，其包括 :

[0083] 用于存储预跳转数据的栈 301 ;

[0084] 预跳转模块 302，其输入端连接指令处理模块 303 的输出，其输入端连接所述栈 301 的输入，当指令处理模块 303 的输出结果指示后面的跳转指令将执行跳转时，将所述跳转指令的跳转位置和跳转目的地作为跳转数据 304 压入所述栈 ;

[0085] 指令地址比较模块 305，其比较所述栈的顶部的跳转位置和待执行的指令地址 306，当所述栈顶部的跳转位置与待执行的指令地址一致时，则弹出所述栈 301 顶部的跳转位置和跳转目的地作为指令地址比较模块 305 的输出，当所述栈 301 顶部的跳转位置与待执行的指令地址 306 不一致时，则以所述待执行的指令地址 306 作为指令地址比较模块的输出；

[0086] 指令读取模块 307，根据指令地址比较模块 305 的输出读取指令；

[0087] 指令处理模块 303，对指令读取模块 307 读取的指令进行处理，输出结果。

[0088] 图 4 是本发明的预跳转系统的一个优选实施例，在典型的微处理器架构下，由程序计数器 PC 指示待执行的指令的指令地址，然后将该指令地址保存在地址暂存器 ITCM 中，指令寄存器 INS 根据该指令地址读取待执行的指令，并在指令处理系统中进行相应的译码、读操作数、运算等操作，并获得结果，如果该结果指示后面会执行跳转指令的跳转操作，则预跳转分支模块 PJM 就将所述跳转指令的跳转位置 JD 和跳转目的地 JA 压入栈 STA 中，当指令地址比较模块比较出程序计数器 PC 指示的指令地址与栈的顶部的跳转位置 JD 一致的时候，则将栈顶部的跳转位置 JD 和跳转目的地 JA 弹栈，其中跳转目的地 JA，即跳转后的指令地址进入地址寄存器 ITCM，指令寄存器根据该跳转后的地址读取指令，并在后续的指令处理器系统中进行处理。

[0089] 上述实施例是用于例示性说明本发明的原理及其功效，而非用于限制本发明。任何本领域技术人员均可在不违背本发明的精神及范畴下，对上述实施例进行修改。因此本发明的保护范围，应如本发明的权利要求书所列。

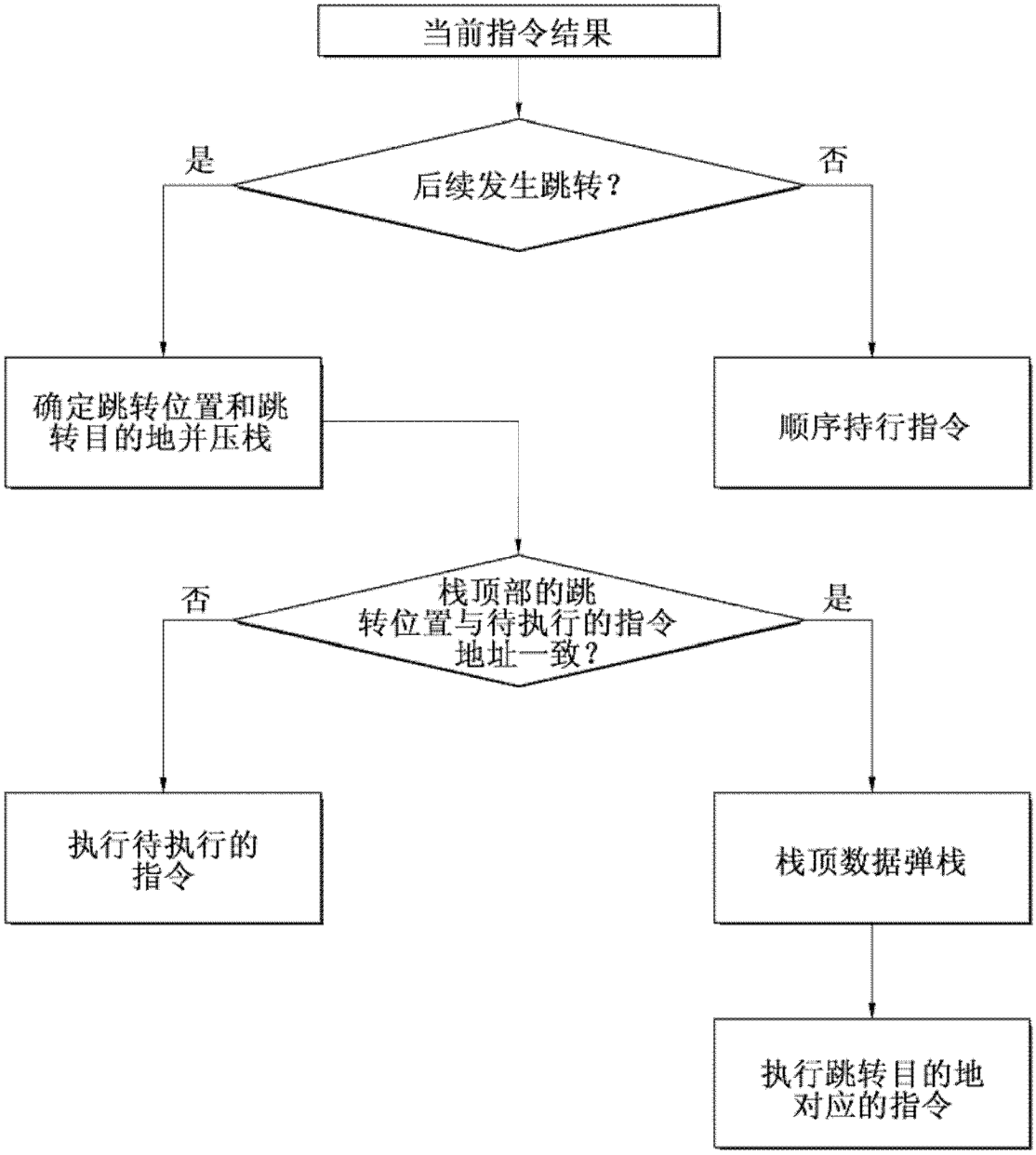

当前指令结果
后续发生跳转？
是
否
确定跳转位置和跳转目的地并压栈
顺序持行指令
栈顶部的跳转位置与待执行的指令地址一致？
否
是
执行待执行的指令
栈顶数据弹栈
执行跳转目的地对应的指令

图 1

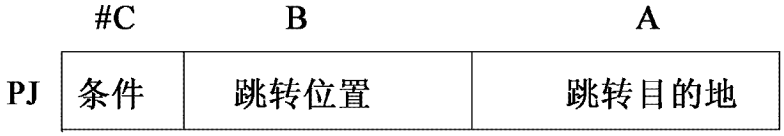

#C
B
A
PJ
条件
跳转位置
跳转目的地

图 2

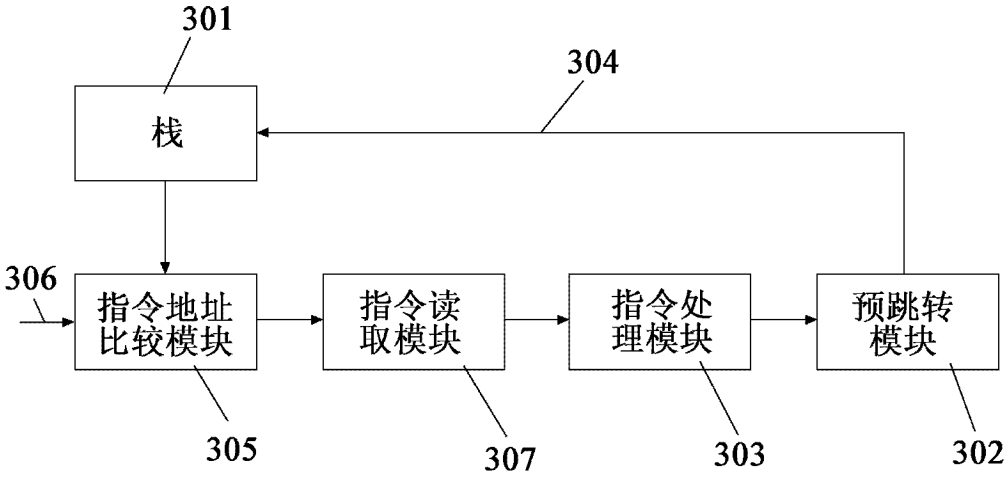

301
304
栈
306
指令地址
比较模块
指令读
取模块
指令处
理模块
预跳转
模块
305
307
303
302

图 3

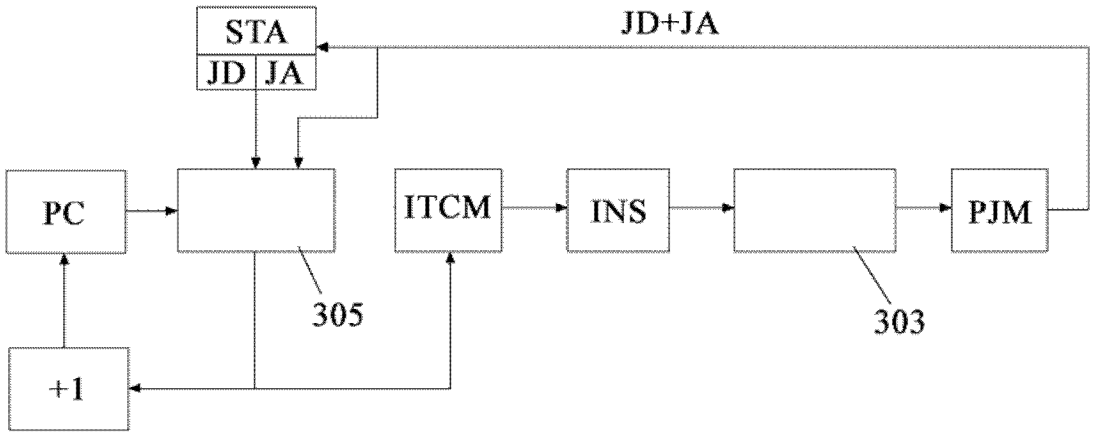

STA
JD JA
JD+JA
PC
ITCM
INS
PJM
305
303
+1

图 4