(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 119166315 A
(43)申请公布日 2024.12.20

(21)申请号 202411650887.9

(22)申请日 2024.11.19

(71)申请人 上海芯力基半导体有限公司
地址 201203 上海市浦东新区亮秀路112号Y1座901B

(72)发明人 葛小燕 陈家棋

(74)专利代理机构 北京锦辉智通专利代理事务所(普通合伙) 16160
专利代理师 马红

(51)Int.Cl.
G06F 9/48(2006.01)
G06F 9/50(2006.01)

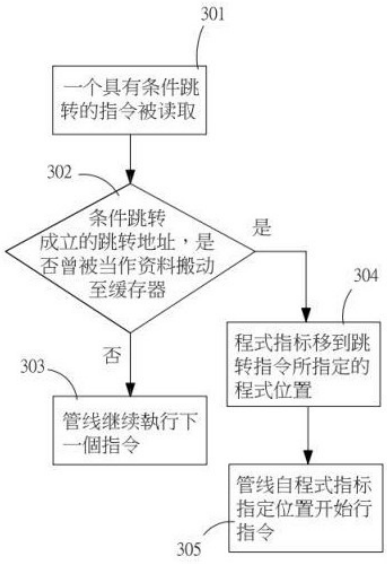
权利要求书2页 说明书9页 附图2页

(54)发明名称

一种具管线处理设计的处理器及执行路径预测方法

(57)摘要

本发明提出了一种具管线处理设计的处理器及执行路径预测方法。所述具管线处理设计的处理器包括内存控制器、指令快取模块、发送单元、排程器和执行单元;执行路径预测方法包括:将指令的执行作为条件跳转的预测满足条件;当有跳转请求的指令被读取时,则检测条件跳转预测指令对应的条件跳转的满足条件是否发生;当所述条件跳转的满足条件没有发生时,则管线继续执行下一指令;当所述条件跳转的满足条件发生时,则程序指针移到满足条件的条件跳转指令所请求的指定位置,管线自所述条件跳转指令所请求的指定位置继续执行指令,本发明通过快取列具有跳转地址的设计,可以降低因为不正确的预测的效果损失,进而提高程序的执行效率。



CN 119166315 A

1.一种具管线处理设计的处理器的执行路径预测方法,其特征在于,所述具管线处理设计的处理器的执行路径预测方法包括:

将指令的执行作为条件跳转的预测满足条件;

当有跳转请求的指令被读取时,则检测条件跳转预测指令对应的条件跳转的满足条件是否发生;

当所述条件跳转的满足条件没有发生时,则管线继续执行下一指令;

当所述条件跳转的满足条件发生时,则程序指针移到满足条件的条件跳转指令所请求的指定位置,管线自所述条件跳转指令所请求的指定位置继续执行指令;

获取指令的历史执行数据,计算指令的指令权重系数,生成执行预测路径,对指令的历史执行数据进行更新,获得指令更新数据,计算指令的权重更新系数,对指令的指令权重系数进行更新,获得执行变化路径,计算权重调节系数,对对应指令的权重进行调节。

2.根据权利要求1所述一种具管线处理设计的处理器的执行路径预测方法,其特征在于,具有跳转请求的该指令于第一次执行,无论指令有无执行过,视为该条件跳转的条件不满足。

3.根据权利要求1所述一种具管线处理设计的处理器的执行路径预测方法,其特征在于,在所述条件跳转的条件被满足的情况下,具有跳转请求的该指令再次被执行时,则该程序指针移动到该跳转请求所指定的位置。

4.根据权利要求1所述一种具管线处理设计的处理器的执行路径预测方法,其特征在于,所述条件跳转所对应的位置记录于一条件跳转记录表中,且每一所述条件跳转所对应的位置可被标注为有效的或无效的。

5.根据权利要求1所述一种具管线处理设计的处理器的执行路径预测方法,其特征在于,获取指令的历史执行数据,计算指令的指令权重系数,生成执行预测路径,对指令的历史执行数据进行更新,获得指令更新数据,计算指令的权重更新系数,对指令的指令权重系数进行更新,获得执行变化路径,计算权重调节系数,对对应指令的权重进行调节,包括:

获取指令的历史执行数据,根据所述指令的历史执行数据计算指令的指令权重系数;

所述指令权重系数的计算公式为:

$$Q_{zli}=\frac{S_i*C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}$$

其中,Q_{zli}为第i个指令的指令权重系数,j为数据采集时间段的指令执行记录次数,S_i为第i次记录的时间衰减因子,用于减少旧执行记录对当前权重的影响,C_i 为第i次记录中指令的成功执行次数,L_{vi}为第i次记录中指令的预设权重,Z_i为第i次记录中指令的总执行次数;

根据指令执行顺序结合指令权重顺序生成执行预测路径;

将每个指令的指令权重系数与预设权重范围进行对比,获得权重对比结果;

所述权重对比结果包括范围外指令和范围内指令;

将执行预测路径中的范围外指令进行剔除,获得执行更新路径;

获取指令执行动态变化数据,根据所述指令执行动态变化数据对指令的历史执行数据进行更新,获得指令更新数据;

根据所述权重对比结果、指令权重系数结合指令更新数据计算指令的权重更新系数；

所述权重更新系数的计算公式为：

$$G_{zli}=\frac{S_i*C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}*\left[\frac{S_i*\Delta C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}-\frac{S_i*\nabla C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}\right]$$

其中，G_{zli}为第i个指令的权重更新系数，ΔC_i为第i次记录中指令的成功执行次数的增加变化量，∇C_i为第i次记录中指令的成功执行次数；

根据所述权重更新系数对指令的指令权重系数进行更新，进而对执行更新路径进行更新，获得执行变化路径；

当指令权重系数在预设权重范围外，且权重更新系数在预设权重范围内时，根据所述指令权重系数、权重更新系数结合预设权重范围计算权重调节系数；

所述权重调节系数的计算公式为：

$$T_{ys}=\left|\frac{X_x-(Q_{zli}+G_{zli})}{Q_{zli}+G_{zli}-S_x}\right|$$

其中，T_{ys}为权重调节系数，X_x为预设权重范围下限值，S_x为预设权重范围上限值；

根据所述权重调节系数对对应指令的权重进行调节，直至指令的权重在预设权重范围内。

6.一种具管线处理设计的处理器，所述具管线处理设计的处理器包括内存控制器、指令快取模块、发送单元、排程器和执行单元；其中，所述内存控制器与指令快取模块进行电性连接；所述指令快取模块与发送单元进行电性连接；所述发送单元与排程器进行电性连接；所述排程器与执行单元进行电性连接，所述处理器包括如权利要求1-5任一所述执行路径预测方法。

7.根据权利要求6所述具管线处理设计的处理器，其特征在于，所述排程器与执行单元进行电性连接包括第一电性连接通道和第二电性连接通道；其中，所述第一电性连接通道用于运作数据发送至所述排程器；所述第二电性连接通道用于确认运作数据重新发送至所述排程器。

8.根据权利要求6所述具管线处理设计的处理器，其特征在于，所述内存控制器与处理器所处系统的系统内存进行电性连接。

9.根据权利要求6所述具管线处理设计的处理器，其特征在于，所述具管线处理设计的处理器还包括分支预测单元和数据快取模块；其中，所述分支预测单元与所述执行单元进行电性连接，并且，所述分支预测单元与所述指令快取模块进行电性连接；所述数据快取模块分别与执行单元和内存控制器进行电性连接。

10.根据权利要求6所述具管线处理设计的处理器，其特征在于，所述具管线处理设计的处理器还包括总线；其中，所述总线与所述执行单元进行电性连接。

一种具管线处理设计的处理器及执行路径预测方法

技术领域

[0001] 本发明提出了一种具管线处理设计的处理器及执行路径预测方法,涉及数据预测技术领域,特别涉及具管线处理设计的处理器及执行路径预测技术领域。

背景技术

[0002] 一般在处理器(CPU)内部对于一个指令的执行,通常需要经过包含以下步骤:fetch(获取指令)、decode(译码指令)、execute(执行指令)及write-back(写回数据)。当完全的执行完各阶段的步骤,则表示完成一条指令。

[0003] 处理器的流水线(Pipeline)处理,其可以同时且分段的执行上述步骤,例如前一指令A执行至译码指令时,次一指令B可以开始执行获取指令。如此一来,通过此种执行方式可以缩短执行完所有指令的所需时间。

[0004] 然而对于特定指令,例如「if」指令,其下一条指令必需根据「if」式中的结果是「true(真)」还是「false(假)」来决定,所以必须等到「if」指令执行完,才能知道下一条指令是什么。此时,无疑降低处理器的执行效率,于是,现今处理器对于这种情况,都设计有一个分支预测(Branch Prediction)机制,就是针对这种if指令,不等它执行完毕,先预测一下执行的结果可能是true还是false,然后将对应条件的指令放进流水线。

[0005] 分支预测的使用范围从每次单纯地产生相同的预测的方法至在该程序中维持先前分支的复杂记录以产生记录型预测的方法。分支预测可以透过硬件优化、编译程序优化或两者皆使用而变得容易。基于通过分支预测机制所提供的预测,可以预测地存取及执行指令。当对分支指令做最终评估时,可以确认分支预测。若预测是不正确时,可撤销基于不正确的预测而预测执行的任何指令。但是分支预测准确率高代表更复杂的算法,同时还会影响CPU的周期时间。

发明内容

[0006] 本发明提供了一种具管线处理设计的处理器及执行路径预测方法,用以解决不正确的预测会造成效果损失等问题:

本发明提出的一种具管线处理设计的处理器及执行路径预测方法,所述具管线处理设计的处理器包括内存控制器、指令快取模块、发送单元、排程器和执行单元;其中,所述内存控制器与指令快取模块进行电性连接;所述指令快取模块与发送单元进行电性连接;所述发送单元与排程器进行电性连接;所述排程器与执行单元进行电性连接。

[0007] 进一步地,所述排程器与执行单元进行电性连接包括第一电性连接通道和第二电性连接通道;其中,所述第一电性连接通道用于运作数据发送至所述排程器;所述第二电性连接通道用于确认运作数据重新发送至所述排程器。

[0008] 进一步地,所述内存控制器与处理器所处系统的系统内存进行电性连接。

[0009] 进一步地,所述具管线处理设计的处理器还包括分支预测单元和数据快取模块;其中,所述分支预测单元与所述执行单元进行电性连接,并且,所述分支预测单元与所述指

令快取模块进行电性连接;所述数据快取模块分别与执行单元和内存控制器进行电性连接。

[0010] 进一步地,所述具管线处理设计的处理器还包括总线;其中,所述总线与所述执行单元进行电性连接。

[0011] 进一步地,所述具管线处理设计的处理器的执行路径预测方法包括:

将指令的执行作为条件跳转的预测满足条件;

当有跳转请求的指令被读取时,则检测条件跳转预测指令对应的条件跳转的满足条件是否发生;

当所述条件跳转的满足条件没有发生时,则管线继续执行下一指令;

当所述条件跳转的满足条件发生时,则程序指针移到满足条件的条件跳转指令所请求的指定位置,管线自所述条件跳转指令所请求的指定位置继续执行指令。

[0012] 进一步地,具有跳转请求的该指令于第一次执行,无论该先指令有无执行过,视为该条件跳转的条件不满足。

[0013] 进一步地,在所述条件跳转的条件被满足的情况下,具有跳转请求的该指令再次被执行时,则该程序指针移动到该跳转请求所指定的位置。

[0014] 进一步地,所述条件跳转所对应的位置记录于一条件跳转记录表中,且每一所述条件跳转所对应的位置可被标注为有效的或无效的。

[0015] 进一步地,计算指令的指令权重系数和权重更新系数,进而计算权重调节系数,对指令进行权重调节,包括:

获取指令的历史执行数据,根据所述指令的历史执行数据计算指令的指令权重系数;

所述指令权重系数的计算公式为:

$$Q_{zli}=\frac{S_i*C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}$$

[0016] 其中,Q_{zli}为第i个指令的指令权重系数,j为数据采集时间段的指令执行记录次数,S_i为第i次记录的时间衰减因子,用于减少旧执行记录对当前权重的影响,C_i 为第i次记录中指令的成功执行次数,L_{vi}为第i次记录中指令的预设权重,Z_i为第i次记录中指令的总执行次数;

根据指令执行顺序结合指令权重顺序生成执行预测路径;

将所述每个指令的指令权重系数与预设权重范围进行对比,获得权重对比结果;

所述权重对比结果包括范围外指令和范围内指令;

将执行预测路径中的范围外指令进行剔除,获得执行更新路径;

获取指令执行动态变化数据,根据所述指令执行动态变化数据对指令的历史执行数据进行更新,获得指令更新数据;

根据所述权重对比结果、指令权重系数结合指令更新数据计算指令的权重更新系数;

所述权重更新系数的计算公式为:

$$G_{zli}=\frac{S_i*C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}*\left[\frac{S_i*\Delta C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}-\frac{S_i*\nabla C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}\right]$$

[0017] 其中,G_{zli}为第i个指令的权重更新系数,ΔC_i为第i次记录中指令的成功执行次数的增加变化量,∇C_i为第i次记录中指令的成功执行次数;

根据所述权重更新系数对指令的指令权重系数进行更新,进而对执行更新路径进行更新,获得执行变化路径;

当指令权重系数在预设权重范围外,且权重更新系数在预设权重范围内时,根据所述指令权重系数、权重更新系数结合预设权重范围计算权重调节系数;

所述权重调节系数的计算公式为:

$$T_{ys}=\left|\frac{X_x-(Q_{zli}+G_{zli})}{Q_{zli}+G_{zli}-S_x}\right|$$

[0018] 其中,T_{ys}为权重调节系数,X_x为预设权重范围下限值,S_x为预设权重范围上限值;

根据所述权重调节系数对对应指令的权重进行调节,直至指令的权重在预设权重范围内。

[0019] 本发明有益效果:通过快取列具有跳转地址的设计,可以降低因为不正确的预测的效果损失,进而提高程序的执行效率。通过指令快取和分支预测技术,减少了对系统内存的访问次数和延迟,提高了指令的执行效率。管线化处理使得指令可以并行执行,进一步提高了处理器的吞吐量。双重电性连接通道(第一通道用于正常发送,第二通道用于重试)的设计,使得处理器在处理异常情况时更加灵活和可靠。总线接口提供了与外部设备通信的能力,增强了处理器的可扩展性和兼容性。排程器根据指令的依赖关系和资源可用性进行智能调度,确保了资源的有效利用和避免了不必要的冲突。数据快取模块减少了数据访问的延迟和带宽消耗,提高了系统整体的性能。高效的指令执行和数据管理能力使得处理器能够更快地响应用户的请求,提升了用户的使用体验。可靠的错误处理和重试机制保证了系统的稳定性和可靠性,减少了因硬件故障导致的系统崩溃和数据丢失的风险。

附图说明

[0020] 图1为本发明的预测方法流程图;

图2是本发明的处理器架构的方块示意图;

图3是本发明的处理器架构的分支预测单元的构成示意图。

具体实施方式

[0021] 以下结合附图对本发明的优选实施例进行说明,应当理解,此处所描述的优选实施例仅用于说明和解释本发明,并不用于限定本发明。

[0022] 本发明的一个实施例,本发明提出的一种具管线处理设计的处理器及执行路径预测方法,所述具管线处理设计的处理器包括内存控制器、指令快取模块、发送单元、排程器和执行单元;其中,所述内存控制器与指令快取模块进行电性连接;所述指令快取模块与发送单元进行电性连接;所述发送单元与排程器进行电性连接;所述排程器与执行单元进行电性连接。

[0023] 所述排程器与执行单元进行电性连接包括第一电性连接通道和第二电性连接通道;其中,所述第一电性连接通道用于运作数据发送至所述排程器;所述第二电性连接通道用于确认运作数据重新发送至所述排程器。

[0024] 所述内存控制器与处理器所处系统的系统内存进行电性连接。

[0025] 所述具管线处理设计的处理器还包括分支预测单元和数据快取模块;其中,所述分支预测单元与所述执行单元进行电性连接,并且,所述分支预测单元与所述指令快取模块进行电性连接;所述数据快取模块分别与执行单元和内存控制器进行电性连接。

[0026] 所述具管线处理设计的处理器还包括总线;其中,所述总线与所述执行单元进行电性连接。

[0027] 上述技术方案的工作原理为:本发明实施例所展示的处理器100包含发送单元101可以配置成接收来自指令快取102的指令,且发送单元101能将指令发送至排程器(scheduler)103,其次指令快取102连接内存控制器104。排程器103可以连接执行单元105,且排程器103用以储存等待决定的运作信息。执行单元105可配置成用以执行存取数据快取106的加载/储存单元,且执行单元105所产生的结果可以输出至总线107,以及来自总线107的数据流可以进入执行单元105;进一步,数据快取106连接内存控制器104,以及执行单元105可以提供重演指示108用以确认运作重新发至排程器103。

[0028] 其次,处理器100内具有分支预测单元109连接执行单元105及指令快取102。内存控制器104连接系统内存110。

[0029] 在一个实施例中,处理器100可以通过设计以兼容于x86架构。需要注意的是处理器100包括分支预测单元109。分支预测单元109用以暂存包含程序位置的信息。

[0030] 本实施例指出一个具体实施例,如图3所示,分支预测单元109包含条件跳转记录表120。条件跳转记录表120的字段包含但不限于标签(Tag)栏121、地址(address)栏122及状态(status)栏123。可理解的是,条件跳转记录表120用以记该条件跳转指令的地址的高位(high byte)或卷标 (tag)、跳转指令的目标跳转位置,以及数据或程序目前是有效的(valid)还是无效的(invalid)。

[0031] 条件跳转记录表120内的信息是可以被自动加载及更新的。例如,当某一跳转指令被第一次执行,则其仍依程序行而依序执行,同时将跳转指令中所要求的跳转地址记录于条件跳转记录表120内,并且被记录成无效的(invalid)。经过一次或数次的执行结果,特别是,在满足预设的跳转条件下,跳转指令再被执行时,即可依所要求的跳转地址,让程序指针(program counter)移动到跳转位置,且在跳转记录表120内的状态,被更改为有效的(valid)。另外,当跳转位置过久没被执行,则跳转记录表120内的状态,也会被更改为无效的(invalid)。

[0032] 上述的预设跳转条件包含但不限于是一个指令的执行。满足预设跳转条件的情形,可以是若有一个非跳转的先指令被执行,且非跳转的先指令所执行的内容与被置于跳转指令的默认跳转条件相符,则可以视为为满足预设跳转条件,因此当跳转指令被执行时,则程序指针(program counter)移动到跳转位置。

[0033] 是以,在跳转指令执行结束之前猜测哪一分支将会被执行,可用以提高处理器的指令管线的效能,本发明的一个实施例揭示一种执行预测的方法,如图1所示,执行程序时,逐一且依序的读取指令列,且一个具有条件跳转的指令被读取(如方块301);随之,执行一

个判断,该判断用以查看是跳转条件成立的跳转地址,是否曾被当作资料移动至缓存器(如方块302);若该指令自条件跳转记录表取得的资料数据位置为无效的,则管线继续执行下一个指令,如方块303。若指令自条件跳转记录表取得的资料位置为有效的,则程式指标(program counter)移动到默认的跳转位置,如方块304;随后程序的执行则自程式指标(program counter)所指位置,接续执行指令,如方块305。

[0034] 根据以上说明,本发明实施例所揭示的处理器及预测方法,可以降低因为不正确的预测之效果损失,进而提高程序的执行效率。

[0035] 当处理器需要执行新的指令时,内存控制器首先从系统内存中读取指令数据,并将其传递给指令快取模块。指令快取模块负责缓存这些指令,以便快速访问。指令快取模块中的指令随后被发送到发送单元,发送单元负责将这些指令按照一定顺序和格式准备好,以便后续处理。发送单元将准备好的指令发送到排程器。排程器根据指令的依赖关系和资源可用性,对指令进行排序和调度,确保其能够按照正确的顺序和时机被执行。排程器与执行单元之间通过第一电性连接通道进行运作数据的发送。执行单元接收来自排程器的指令,并执行相应的操作。如果在执行过程中遇到需要重试或重新发送的情况(如缓存未命中、资源冲突等),执行单元会通过第二电性连接通道向排程器发送确认信号,请求重新发送相关指令或数据。分支预测单元通过分析历史执行模式和当前指令流,预测即将执行的分支路径,以减少因分支跳转带来的延迟。其与指令快取模块和执行单元相连,以便及时获取和更新预测信息。数据快取模块用于缓存执行单元所需的数据,以减少对系统内存的访问次数。其与执行单元和内存控制器相连,确保数据能够快速、准确地传输到执行单元。总线与执行单元相连,为处理器与外部设备或系统其他部分提供通信接口。通过总线,执行单元可以与其他处理器、内存或其他I/O设备交换数据和控制信息。

[0036] 上述技术方案的效果为:通过指令快取和分支预测技术,减少了对系统内存的访问次数和延迟,提高了指令的执行效率。管线化处理使得指令可以并行执行,进一步提高了处理器的吞吐量。双重电性连接通道(第一通道用于正常发送,第二通道用于重试)的设计,使得处理器在处理异常情况时更加灵活和可靠。总线接口提供了与外部设备通信的能力,增强了处理器的可扩展性和兼容性。排程器根据指令的依赖关系和资源可用性进行智能调度,确保了资源的有效利用和避免了不必要的冲突。数据快取模块减少了数据访问的延迟和带宽消耗,提高了系统整体的性能。高效的指令执行和数据管理能力使得处理器能够更快地响应用户的请求,提升了用户的使用体验。可靠的错误处理和重试机制保证了系统的稳定性和可靠性,减少了因硬件故障导致的系统崩溃和数据丢失的风险。

[0037] 本发明的一个实施例,所述具管线处理设计的处理器的执行路径预测方法包括:

将指令的执行作为条件跳转的预测满足条件;

当有跳转请求的指令被读取时,则检测条件跳转预测指令对应的条件跳转的满足条件是否发生;

当所述条件跳转的满足条件没有发生时,则管线继续执行下一指令;

当所述条件跳转的满足条件发生时,则程序指针移到满足条件的条件跳转指令所请求的指定位置,管线自所述条件跳转指令所请求的指定位置继续执行指令。

[0038] 上述技术方案的工作原理为:在指令执行过程中,处理器将某些指令(特别是条件跳转指令)的执行结果作为后续跳转预测的依据。这种预测基于历史执行模式和当前上下

文信息,试图提前确定条件跳转是否会发生。当指令流中遇到条件跳转指令时,处理器会读取该指令,并识别出这是一个需要跳转判断的指令。此时,处理器会检查与该条件跳转指令相关联的预测信息。处理器会根据条件跳转指令的具体条件,检查当前的状态或寄存器值等,以确定条件是否满足。这一步骤是判断是否需要实际执行跳转的关键。如果条件不满足(即预测未发生或实际检测未发生),则处理器会继续沿当前管线执行下一指令,不改变执行路径。如果条件满足(即预测发生且实际检测也发生),则处理器会更新程序指针(PC),将其指向条件跳转指令所请求的指定位置。随后,管线将从该新位置开始继续执行指令,实现执行路径的跳转。

[0039] 上述技术方案的效果为:通过条件跳转预测,处理器能够提前做出是否跳转的决策,从而减少了因等待条件判断结果而产生的延迟。这对于提高程序的执行效率和响应速度至关重要。预测准确时,处理器能够连续执行指令流中的多个指令,而无需因条件跳转而中断。这有助于提高处理器的指令吞吐量,使其能够处理更多的工作负载。预测和跳转机制使得处理器能够更有效地利用管线资源。当预测到条件跳转将发生时,处理器可以提前准备跳转目标处的指令和数据,从而减少因资源冲突或等待而造成的浪费。总体上,该方法通过优化条件跳转的处理流程,提高了程序的执行效率和性能。这对于需要处理大量条件分支和复杂控制流的程序来说尤为重要。对于用户来说,更快的程序响应和更高的执行效率代表更好的使用体验。无论是在处理日常任务还是在运行大型应用程序时,用户都能感受到明显的性能提升。

[0040] 本发明的一个实施例,具有跳转请求的该指令于第一次执行,无论该先指令有无执行过,视为该条件跳转的条件不满足。

[0041] 上述技术方案的工作原理为:处理器在解析指令流时,会跟踪每条指令的执行状态。当遇到具有跳转请求的指令,并且这是该指令在当前上下文中的第一次执行时,处理器会特别标记这一点。由于缺乏历史执行数据或上下文信息来准确预测该条件跳转是否会真正发生,处理器采取了一种保守的假设,即认为该条件跳转的条件不满足。这代表,在没有足够证据支持条件跳转会发生的情况下,处理器不会冒险改变当前的执行路径。基于上述假设,处理器会继续沿当前的管线执行后续的指令,而不执行跳转操作。这样做可以确保程序流的稳定性,并避免因错误的跳转预测而导致的执行错误或性能下降。当该条件跳转指令实际上被执行,并且其条件被确定满足或不满足时,处理器会更新与该指令相关联的预测信息。这些信息将在未来的执行中用于更准确地预测条件跳转的行为。

[0042] 上述技术方案的效果为:通过在首次执行时采取保守的预测策略,处理器减少了因错误预测而导致的执行路径错误的风险。这有助于提高程序的执行稳定性和可靠性。在没有足够信息支持复杂预测的情况下,选择简单的保守策略可以简化处理器的预测逻辑,降低实现难度和成本。随着程序的执行和预测信息的积累,处理器能够逐渐提高条件跳转的预测准确性。这种逐步优化的过程有助于在保持稳定性的同时提高性能。通过在首次执行时避免冒险的跳转预测,处理器减少了因预测错误而导致的性能波动。这对于需要稳定性能的应用程序来说尤为重要。在程序或系统启动阶段,许多指令都是首次执行。采用保守的预测策略可以确保这些指令能够平稳地执行,从而支持快速启动和初始化过程。

[0043] 本发明的一个实施例,在所述条件跳转的条件被满足的情况下,具有跳转请求的该指令再次被执行时,则该程序指针移动到该跳转请求所指定的位置。

[0044] 上述技术方案的工作原理为:当处理器遇到条件跳转指令时,会首先评估与该指令相关联的条件表达式。这通常涉及检查特定的寄存器、内存位置或先前指令的执行结果。处理器根据条件表达式的评估结果来判断条件是否满足。如果条件为真(即满足),则处理器会决定执行跳转操作。一旦决定执行跳转,处理器会查找该条件跳转指令中指定的跳转目标地址。这个地址是程序中的一个具体位置,通常是一个标签或内存地址,标识了跳转后应该开始执行的指令序列。处理器将当前的程序指针(PC)更新为跳转目标地址。这一步骤是改变执行路径的关键,因为指示了接下来应该从哪个位置开始读取和执行指令。由于程序指针的更新,处理器的指令管线会重新定向以从新的地址加载指令。后续的指令获取、解码和执行操作都将基于新的执行路径进行。一旦指令管线重新定向完成,处理器就会从跳转目标地址开始继续执行指令。跳过了一些原本在当前执行路径上的指令,直接跳转到另一个代码块执行。

[0045] 上述技术方案的效果为:条件跳转允许程序根据条件选择性地执行代码块,从而避免了执行不必要的指令。这有助于减少程序的执行时间和资源消耗,提高整体效率。通过条件跳转,程序可以更加灵活地控制执行流程,实现复杂的控制逻辑和决策过程。这对于开发具有复杂功能和行为的应用程序至关重要。条件跳转使得程序员能够组织代码以更自然和直观的方式表达逻辑结构。例如,循环和条件语句等控制流结构可以基于条件跳转指令来实现,从而使代码更加清晰和易于理解。在模块化编程中,条件跳转可以帮助实现函数或子程序的调用和返回。通过跳转到特定的代码块并执行该块中的指令,程序可以重用和共享代码,减少冗余并提高可维护性。在需要快速响应的应用程序中,条件跳转可以减少处理时间并加快决策速度。

[0046] 本发明的一个实施例,所述条件跳转所对应的位置记录于一条件跳转记录表中,且每一所述条件跳转所对应的位置可被标注为有效的或无效的。

[0047] 上述技术方案的工作原理为:当编译器或汇编器处理源代码或汇编代码时,其会为每个条件跳转指令生成一个条目,并将其添加到条件跳转记录表中。这个条目至少包含两个关键信息:条件跳转指令的标识符(如标签或地址)和目标位置的地址。在某些情况下,跳转目标的位置可能会变得无效。例如,如果跳转目标位于一个被删除或重构的代码块中,或者由于条件逻辑的变化而不再需要该跳转。在这些情况下,处理器或相关软件会更新条件跳转记录表,将相应的跳转位置标注为无效。当处理器执行到条件跳转指令时,它会首先查找条件跳转记录表以获取目标地址,并检查该地址的有效性。如果目标地址被标注为有效,则处理器会正常执行跳转操作;如果目标地址被标注为无效,则处理器可能会采取特定的错误处理措施,如抛出异常、执行备用代码路径或继续执行下一条指令。在程序执行期间,如果检测到跳转目标位置的有效性发生变化(例如,通过运行时分析或动态代码生成),处理器或相关软件可以动态地更新条件跳转记录表,以反映这些变化。

[0048] 上述技术方案的效果为:通过在条件跳转记录表中标注跳转位置的有效性,处理器可以避免执行到无效或已删除的代码区域,从而提高程序的稳定性和可靠性。条件跳转记录表提供了一种集中管理跳转信息的方式,使得在修改或重构代码时更容易跟踪和更新跳转目标的有效性。这有助于减少因遗漏或错误更新跳转信息而导致的错误。在运行时,处理器或相关软件可以根据程序的实际执行情况动态地更新条件跳转记录表,以优化执行路径和性能。例如,如果某个条件跳转在大多数情况下都不满足,处理器可能会选择跳过该跳

转的预测和检查过程,以减少开销。条件跳转记录表提供了一种结构化的方式来组织和表示跳转信息,使得代码的逻辑结构和控制流更加清晰和易于理解。这有助于其他开发人员更快地熟悉和修改代码。在模块化编程中,条件跳转记录表可以帮助实现模块之间的解耦和独立更新。通过更新跳转记录表而不是修改模块内部的跳转逻辑,可以更容易地替换或升级模块,而不会影响其他部分的程序。

[0049] 本发明的一个实施例,计算指令的指令权重系数和权重更新系数,进而计算权重调节系数,对指令进行权重调节,包括:

获取指令的历史执行数据,根据所述指令的历史执行数据计算指令的指令权重系数;

所述指令权重系数的计算公式为:

$$Q_{zli}=\frac{S_i*C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}$$

[0050] 其中,Q_{zli}为第i个指令的指令权重系数,j为数据采集时间段的指令执行记录次数,S_i为第i次记录的时间衰减因子,用于减少旧执行记录对当前权重的影响,C_i 为第i次记录中指令的成功执行次数(满足条件跳转),L_{vi}为第i次记录中指令的预设权重,Z_i为第i次记录中指令的总执行次数(无论是否满足条件跳转);

根据指令执行顺序结合指令权重顺序生成执行预测路径;

将所述每个指令的指令权重系数与预设权重范围进行对比,获得权重对比结果;

所述权重对比结果包括范围外指令和范围内指令;

将执行预测路径中的范围外指令进行剔除,获得执行更新路径;

获取指令执行动态变化数据,根据所述指令执行动态变化数据对指令的历史执行数据进行更新,获得指令更新数据;

根据所述权重对比结果、指令权重系数结合指令更新数据计算指令的权重更新系数;

所述权重更新系数的计算公式为:

$$G_{zli}=\frac{S_i*C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}*\left[\frac{S_i*\Delta C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}-\frac{S_i*\nabla C_i*L_{vi}}{\sum_{i=1}^{j}(S_i*Z_i)}\right]$$

[0051] 其中,G_{zli}为第i个指令的权重更新系数,ΔC_i为第i次记录中指令的成功执行次数(满足条件跳转)的增加变化量,∇C_i为第i次记录中指令的成功执行次数(满足条件跳转);

根据所述权重更新系数对指令的指令权重系数进行更新,进而对执行更新路径进行更新,获得执行变化路径;

当指令权重系数在预设权重范围外,且权重更新系数在预设权重范围内时,根据所述指令权重系数、权重更新系数结合预设权重范围计算权重调节系数;

所述权重调节系数的计算公式为:

$$T_{ys}=\left|\frac{X_x-(Q_{zli}+G_{zli})}{Q_{zli}+G_{zli}-S_x}\right|$$

[0052] 其中,T_{ys}为权重调节系数,X_x为预设权重范围下限值,S_x为预设权重范围上限值;

根据所述权重调节系数对对应指令的权重进行调节,直至指令的权重在预设权重范围内。

[0053] 上述技术方案的工作原理为:系统获取每条指令的历史执行数据,包括指令的执行记录次数、每次记录的时间衰减因子(减少旧数据的影响)、每次记录的成功执行次数、预设权重以及总执行次数。使用这些历史数据计算每条指令的指令权重系数。本考虑了时间衰减、成功执行次数、预设权重和总执行次数,确保旧数据的影响逐渐减弱,同时突出了成功执行次数的重要性。系统根据指令的执行顺序和指令权重顺序生成一个执行预测路径。这个路径反映了根据当前权重系数预测的指令执行顺序。将每条指令的指令权重系数与预设的权重范围进行对比,将权重在范围外的指令视为异常或需调整的指令,从而剔除这些指令,获得更新后的执行路径。系统获取指令执行的动态变化数据,更新历史执行数据,并计算每条指令的权重更新系数。这个系数考虑了成功执行次数的增加变化量,以反映指令性能的改善或恶化。使用权重更新系数对指令权重系数进行更新,并据此更新执行路径,获得执行变化路径。这一步确保了路径能够根据指令性能的最新变化进行调整。对于权重在预设范围外的指令,如果其权重更新系数在预设范围内,系统计算权重调节系数,并使用该系数对指令权重进行调节,直至权重回到预设范围内。这一步骤确保了指令权重的稳定性和合理性。

[0054] 上述技术方案的效果为:通过动态调整指令权重,系统能够更准确地预测和执行指令,减少不必要的执行开销,提高整体执行效率。系统能够根据指令执行的动态变化实时更新权重,适应不同的执行环境和条件,增强系统的适应性和鲁棒性。通过权重调节,系统能够更合理地分配资源给重要的指令,确保关键任务得到优先处理,提高资源利用率。指令执行的准确性和效率的提升,以及资源分配的优化,最终将提升用户的整体体验,使用户能够更高效地完成任务。系统通过自动化的权重计算和调节,减少了人工干预的需要,降低了系统的维护成本。

[0055] 显然,本领域的技术人员可以对本发明进行各种改动和变型而不脱离本发明的精神和范围。这样,倘若本发明的这些修改和变型属于本发明权利要求及其等同技术的范围之内,则本发明也意图包含这些改动和变型在内。

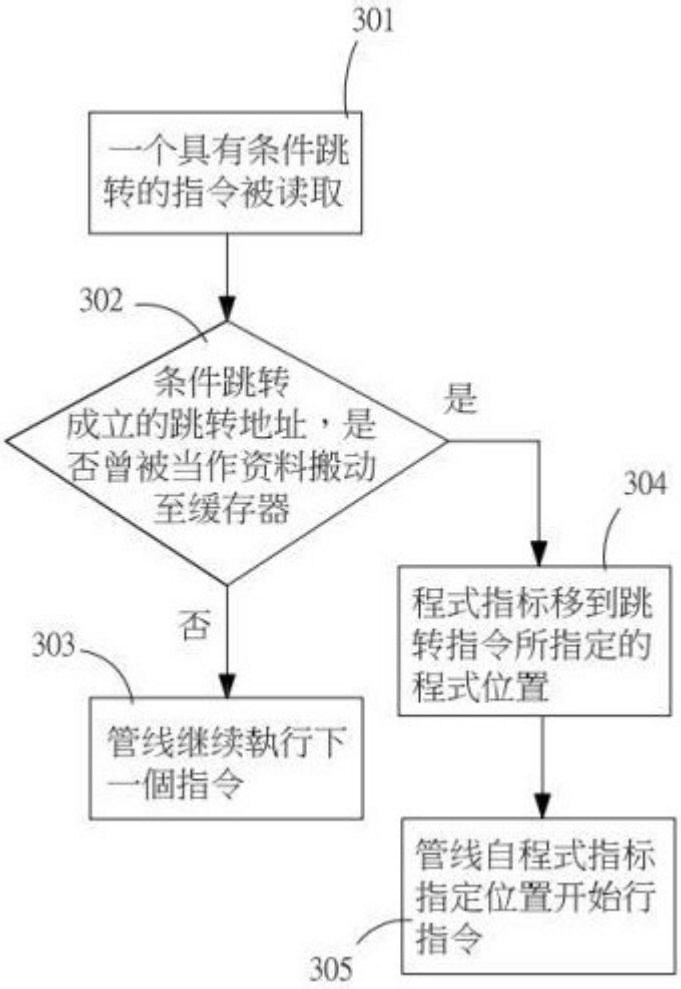

301
一个具有条件跳转的指令被读取
302
条件跳转成立的跳转地址，是否曾被当作资料搬动至缓存器
是
否
303
管线继续執行下一個指令
304
程式指标移到跳转指令所指定的程式位置
管线自程式指标指定位置开始行指令
305

图 1

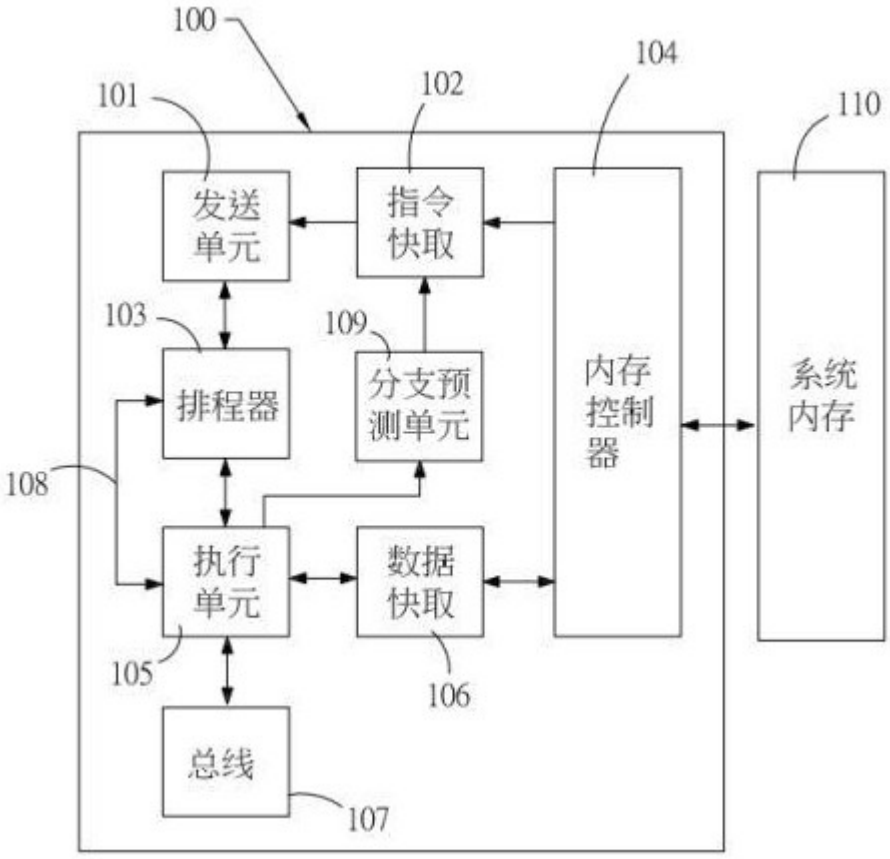

100
101
102
104
110
发送单元
指令快取
103
109
排程器
分支预测单元
内存控制器
系统内存
108
执行单元
数据快取
105
106
总线
107

图 2

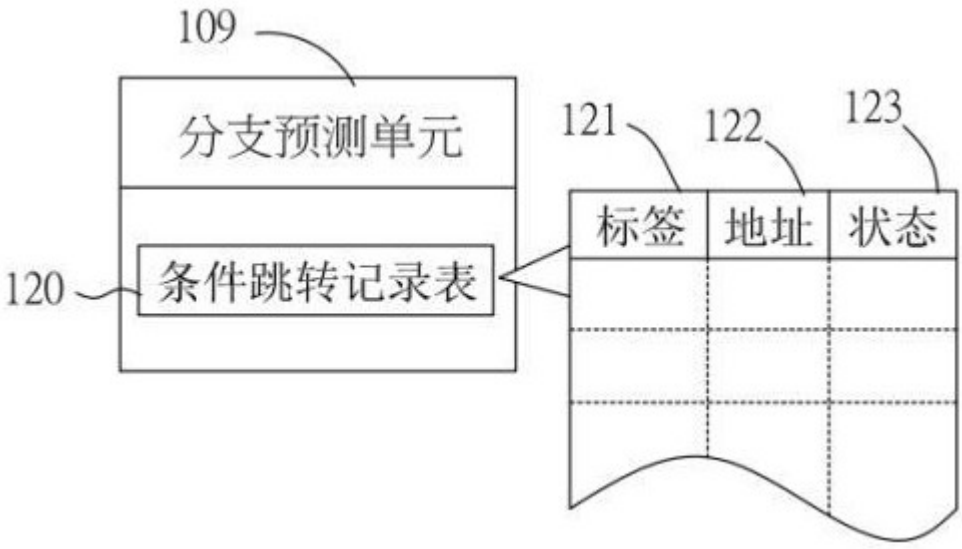

109
分支预测单元
121
122
123
标签
地址
状态
120
条件跳转记录表

图 3