(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 114647447 A
(43)申请公布日 2022.06.21

(21)申请号 202111367114.6

(22)申请日 2021.11.18

(30)优先权数据
17/128,814 2020.12.21 US

(71)申请人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 孙科 布兰科·罗德里戈 胡科开 杰森·布兰特

(74)专利代理机构 北京东方亿思知识产权代理有限责任公司 11258
专利代理师 杨佳婧

(51)Int.Cl.
G06F 9/38(2006.01)
G06F 9/32(2006.01)

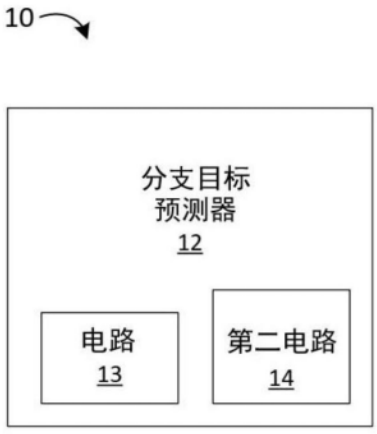
权利要求书3页 说明书25页 附图18页

(54)发明名称

基于上下文的存储器间接分支目标预测

(57)摘要

本申请公开了基于上下文的存储器间接分支目标预测。一种集成电路的实施例可包括分支目标预测器，来为一个或多个指令提供分支目标预测，该分支目标预测器包括电路来识别该一个或多个指令中的存储器间接分支，并且基于存储器间接分支的上下文提供存储器间接分支的预测目标。公开并且要求保护了其他实施例。



CN 114647447 A

1.一种集成电路,包括:

分支目标预测器,用于为一个或多个指令提供分支目标预测,所述分支目标预测器包括用于执行以下操作的电路:

识别所述一个或多个指令中的存储器间接分支,并且

基于所述存储器间接分支的上下文来提供所述存储器间接分支的预测目标。

2.根据权利要求1所述的集成电路,其中,所述存储器间接分支的上下文对应于所述存储器间接分支的目标指针。

3.根据权利要求2所述的集成电路,其中,所述分支目标预测器还包括第二电路,以用于:

在基于所述存储器间接分支的上下文的所述存储器间接分支的预测目标和来自一个或多个其他间接分支目标预测器的预测之间进行选择;并且

相对于所述来自一个或多个其他间接分支目标预测器的预测,向基于所述存储器间接分支的上下文的所述存储器间接分支的预测目标提供更高的优先级。

4.根据权利要求2至3中任一项所述的集成电路,其中,所述电路还用于:

维持数据结构以将目标指针信息与目标信息相关联;

确定所述存储器间接分支的目标指针是否匹配存储在所述数据结构中的目标指针信息;并且如果确定是,

则基于所述数据结构中的与匹配的目标指针信息相关的目标信息来提供所述存储器间接分支的预测目标。

5.根据权利要求4所述的集成电路,其中,所述数据结构包括条目的阵列,每个条目包括由相应的目标指针信息标记的目标信息,以缓存所述目标指针和分支目标之间的相关性,并且其中所述电路还用于:

基于所述存储器间接分支的目标指针的一个或多个预定位为所述存储器间接分支计算标签值;

确定计算出的标签值是否匹配所述阵列的有效条目的标签;并且如果确定是,

则基于来自所述阵列的具有匹配标签的相应条目的目标信息来提供所述存储器间接分支的预测目标。

6.根据权利要求5所述的集成电路,其中,所述阵列的条目包括:

从存储器间接分支的目标指针计算出的标签字段;

具有与所述目标指针相关的目标信息的目标字段;

有效性字段,用于指示出所述条目的有效性;以及

使用字段,用于存储所述条目的使用信息以用于替换选择。

7.根据权利要求5所述的集成电路,其中,所述电路还用于:

基于所述存储器间接分支的目标指针的一个或多个预定位和控制寄存器的预定位来计算所述存储器间接分支的标签值。

8.一种方法,包括:

由分支预测单元为一个或多个指令提供分支目标预测;

识别所述一个或多个指令中的存储器间接分支;以及

基于所述存储器间接分支的上下文来提供所述存储器间接分支的预测目标。

9.根据权利要求8所述的方法,其中,所述存储器间接分支的上下文对应于所述存储器间接分支的目标指针。

10.根据权利要求9所述的方法,还包括:

在基于所述存储器间接分支的上下文的所述存储器间接分支的预测目标和一个或多个其他间接分支目标预测之间进行选择;以及

相对于所述一个或多个其他间接分支目标预测,向基于所述存储器间接分支的上下文的所述存储器间接分支的预测目标提供更高的优先级。

11.根据权利要求9至10中任一项所述的方法,还包括:

维持数据结构以将目标指针信息与目标信息相关联;

确定待预测的存储器间接分支的目标指针是否匹配存储在所述数据结构中的目标指针信息;并且如果确定是,

则基于所述数据结构中的与匹配的目标指针信息相关的目标信息来提供所述存储器间接分支的预测目标。

12.根据权利要求11所述的方法,其中,所述数据结构包括条目的阵列,每个条目包括由相应的目标指针信息标记的目标信息,以缓存所述目标指针和分支目标之间的相关性,所述方法还包括:

基于所述存储器间接分支的目标指针的一个或多个预定位为所述存储器间接分支计算标签值;

确定计算出的标签值是否匹配所述阵列的有效条目的标签;并且如果确定是,

则基于来自具有匹配的标签值的条目的目标信息来提供所述存储器间接分支的预测目标。

13.根据权利要求12所述的方法,其中,所述阵列的条目包括:

从所述存储器间接分支的目标指针计算出的标签字段;

具有与所述目标指针相关的目标信息的目标字段;

有效性字段,用于指示出所述条目的有效性;以及

使用字段,用于存储所述条目的使用信息以用于替换选择。

14.根据权利要求12所述的方法,还包括:

基于所述存储器间接分支的目标指针的一个或多个预定位和控制寄存器的预定位来计算所述存储器间接分支的标签值。

15.一种电子装置,包括:

前端单元,用于提取和解码一个或多个指令;以及

与所述前端单元通信耦合的执行单元,用于执行解码的一个或多个指令并且向所述前端单元提供信息,

其中,所述前端单元包括:

分支预测单元,用于为所述一个或多个指令提供分支预测和分支目标预测信息,以及

所述分支预测单元内的基于上下文的存储器间接分支预测器,所述基于上下文的存储器间接分支预测器包括电路,以用于识别所述一个或多个指令中的存储器间接分支,并且基于所述存储器间接分支的目标指针来提供所述存储器间接分支的预测目标。

16.根据权利要求15所述的装置,其中,所述前端单元还包括:

所述分支预测单元中的一个或多个其他间接分支预测器，用于提供其他预测目标，其中，所述分支预测单元被配置为在来自所述基于上下文的存储器间接分支预测器的预测目标和来自所述一个或多个其他间接分支目标预测器的其他预测目标之间进行选择，

并且其中，所述分支预测单元被配置为：相对于来自所述一个或多个其他间接分支目标预测器的其他预测目标，向来自所述基于上下文的存储器间接分支预测器的预测目标提供更高的优先级。

17.根据权利要求15至16中任一项所述的装置，其中，所述基于上下文的存储器间接分支预测器的电路还用于：

维持数据结构以将目标指针信息与目标信息相关联；

确定所述存储器间接分支的目标指针是否匹配存储在所述数据结构中的目标指针信息；并且如果确定是，

则基于所述数据结构中的与匹配的目标指针信息相关的目标信息来提供所述存储器间接分支的预测目标。

18.根据权利要求17所述的装置，其中，所述数据结构包括条目的阵列，每个条目包括由相应的目标指针信息标记的目标信息，以缓存所述目标指针和分支目标之间的相关性，并且其中，所述基于上下文的存储器间接分支预测器的电路还用于：

基于所述存储器间接分支的目标指针的一个或多个预定位为所述存储器间接分支计算标签值；

确定计算出的标签值是否匹配所述阵列的有效条目的标签；并且如果确定是，

则基于来自计算出的标签值所匹配的条目的目标信息来提供所述存储器间接分支的预测目标。

19.根据权利要求18所述的装置，其中，所述阵列的条目包括：

从所述存储器间接分支的目标指针计算出的标签字段；

具有与所述目标指针相关的目标信息的目标字段；

有效性字段，用于指示出所述条目的有效性；以及

使用字段，用于存储所述条目的使用信息以用于替换选择。

20.根据权利要求18所述的装置，其中，所述基于上下文的存储器间接分支预测器的电路还用于：

基于所述存储器间接分支的目标指针的一个或多个预定位和控制寄存器的预定位来计算所述存储器间接分支的标签值。

基于上下文的存储器间接分支目标预测

技术领域

[0001] 本公开总体上涉及处理器技术、分支预测技术和分支目标预测技术。

背景技术

[0002] 一些中央处理器单元(central processor unit,CPU)核心可以利用推测性执行来避免管线停滞并且实现更好的性能,这使得执行可以继续进行,而不必等待对分支目标或方向的架构解决。分支预测技术利用数字电路,该数字电路在分支指令被执行之前猜测分支的走向。正确的预测/猜测改善了指令管线中的流程。

[0003] 一般而言,对于分支的推测性执行存在两种预测:条件分支的分支预测,这可被理解为对分支被“采取”与“不采取”的预测;以及无条件或已采取的条件分支的分支目标预测,包括直接分支和间接分支两者。间接分支预测是整个分支目标预测的一个重要部分,因为间接分支在解决其目标时通常涉及更高的时延,尤其是对于需要从特定存储器位置提取其目标的存储器间接分支。分支预测单元(branch prediction unit,BPU)可以通过基于诸如以下信息向CPU的前端(frond-end,FE)提供分支预测和分支目标预测两者来支持推测性执行,例如:分支指令指针(instruction pointer,IP)、分支类型以及预测点之前的控制流历史(也称为分支历史)。

发明内容

[0004] 根据本公开的第一方面,提供了一种集成电路,包括:分支目标预测器,用于为一个或多个指令提供分支目标预测,所述分支目标预测器包括用于执行以下操作的电路:识别所述一个或多个指令中的存储器间接分支,并且基于所述存储器间接分支的上下文来提供所述存储器间接分支的预测目标。

[0005] 根据本公开的第二方面,提供了一种方法,包括:由分支预测单元为一个或多个指令提供分支目标预测;识别所述一个或多个指令中的存储器间接分支;以及基于所述存储器间接分支的上下文来提供所述存储器间接分支的预测目标。

[0006] 根据本公开的第三方面,提供了一种电子装置,包括:前端单元,用于提取和解码一个或多个指令;以及与所述前端单元通信耦合的执行单元,用于执行解码的一个或多个指令并且向所述前端单元提供信息,其中,所述前端单元包括:分支预测单元,用于为所述一个或多个指令提供分支预测和分支目标预测信息,以及所述分支预测单元内的基于上下文的存储器间接分支预测器,所述基于上下文的存储器间接分支预测器包括电路,以用于识别所述一个或多个指令中的存储器间接分支,并且基于所述存储器间接分支的目标指针来提供所述存储器间接分支的预测目标。

附图说明

[0007] 在附图中以示例方式而非限制方式图示了本发明的各种实施例,在附图中:

[0008] 图1是根据一个实施例的集成电路的示例的框图;

[0009] 图2A至2C是根据一个实施例的方法的示例的流程图；

[0010] 图3是根据一个实施例的电子装置的示例的框图；

[0011] 图4是根据一个实施例的控制流的示例的流程图；

[0012] 图5是根据一个实施例的控制流的另一示例的流程图；

[0013] 图6是根据一个实施例的基于上下文的存储器间接分支预测阵列的示例的框图；

[0014] 图7是根据一个实施例的集成电路的另一示例的框图；

[0015] 图8A至8C是根据一个实施例的方法的另一示例的流程图；

[0016] 图9是根据一个实施例的电子装置的另一示例的框图；

[0017] 图10是根据一个实施例的循环分支预测阵列的示例的框图；

[0018] 图11A是根据本发明的实施例图示出示例性有序管线和示例性寄存器重命名、无序发出/执行管线两者的框图。

[0019] 图11B是根据本发明的实施例图示出要被包括在处理器中的有序架构核心的示例性实施例和示例性寄存器重命名、无序发出/执行架构核心两者的框图；

[0020] 图12A-12B图示出更具体的示例性有序核心架构的框图，该核心将是芯片中的若干个逻辑块(包括相同类型和/或不同类型的其他核心)之一；

[0021] 图13是根据本发明的实施例的处理器的框图，该处理器可以具有多于一个核心，可以具有集成的存储器控制器，并且可以具有集成的图形；

[0022] 图14-图17是示例性计算机架构的框图；并且

[0023] 图18是根据本发明的实施例与使用软件指令转换器来将源指令集中的二进制指令转换成目标指令集中的二进制指令相对比的框图。

具体实施方式

[0024] 本文论述的实施例以各种方式提供了用于分支预测或分支目标预测的技术和机制。本文描述的技术可被实现在一个或多个电子设备中。可以利用本文描述的技术的电子设备的非限制性示例包括任何种类的移动设备和/或固定设备，例如相机、蜂窝电话、计算机终端、桌面型计算机、电子阅读器、传真机、一体机(kiosk)、膝上型计算机、上网本计算机、笔记本计算机、互联网设备、支付终端、个人数字助理、媒体播放器和/或记录器、服务器(例如，刀片式服务器、机架安装式服务器、其组合，等等)、机顶盒、智能电话、平板个人计算机、超便携个人计算机、有线电话、其组合，等等。更一般而言，本文描述的技术可用于包括集成电路的各种电子设备的任何一种中，该集成电路可操作来预测分支目标或者分支指令是被采取还是不采取。

[0025] 在接下来的描述中，论述了许多细节以提供对本公开的实施例的更透彻说明。然而，本领域技术人员将会清楚，可以在没有这些具体细节的情况下实践本公开的实施例。在其他实例中，以框图形式而不是详细示出公知的结构和设备，以避免模糊本公开的实施例。

[0026] 注意，在实施例的相应附图中，以线条来表示信号。一些线条可能更粗，以指示出更大数目的构成信号路径，和/或在一端或多端具有箭头，以指示出信息流的方向。这种指示并不旨在是限制性的。更确切地说，这些线条与一个或多个示例性实施例结合使用来帮助更容易理解电路或逻辑单元。由设计需要或偏好决定的任何所表示的信号可实际上包括可在任一方向上行进并且可利用任何适当类型的信号方案来实现的一个或多个信号。

[0027] 在整个说明书中，以及在权利要求中，术语“连接”意指直接连接，例如连接的事物之间的电连接、机械连接或磁连接，而没有任何中间设备。术语“耦合”意指直接或间接连接，例如连接的事物之间的直接电连接、机械连接或磁连接，或者通过一个或多个无源或有源中间设备的间接连接。术语“电路”或“模块”可以指被布置为彼此协作以提供期望的功能的一个或多个无源和/或有源组件。术语“信号”可以指至少一个电流信号、电压信号、磁信号、或者数据/时钟信号。“一”、“一个”和“该”的含义包括复数引用。“在……中”的含义包括“在……中”和“在……上”。

[0028] 术语“设备”通常可以指的是根据该术语的使用的上下文的装置。例如，设备可以指层或结构的堆叠、单个结构或层、具有有源和/或无源元件的各种结构的连接，等等。一般而言，设备是三维结构，该三维结构具有x-y-z笛卡儿坐标系中沿着x-y方向的平面和沿着z方向的高度。设备的平面也可以是包括该设备的装置的平面。

[0029] 术语“缩放”一般是指将某个设计(图解和布局)从一个工艺技术转换到另一个工艺技术并随后减小布局面积。术语“缩放”一般也指在同一技术节点内减小布局和器件的大小。术语“缩放”还可以指相对于另一参数(例如，电力供应水平)调整信号频率(例如，减慢或加速——即分别是缩小或放大)。

[0030] 术语“基本上”、“接近”、“大致”、“近似”和“大约”一般指在目标值的+/-10％内。例如，除非在其使用的明确上下文中另有指明，否则术语“基本上等于”、“大约等于”和“大致等于”意指在这样描述的事物之间没有超过偶然的差异。在本领域中，这种差异通常不超过预定目标值的+/-10％。

[0031] 要理解，这样使用的术语在适当的情况下是可互换的，使得本文描述的发明的实施例例如能够按与本文所示或以其他方式描述的那些不同的其他朝向来操作。

[0032] 除非另有指明，否则使用序数形容词“第一”、“第二”和“第三”等等来描述共同对象只是表明相似对象的不同实例被引用，而并不是要暗示这样描述的对象必须在时间上、空间上、排名上或者以任何其他方式处于给定的序列中。

[0033] 说明书中和权利要求中的术语“左”、“右”、“前”、“后”、“顶部”、“底部”、“之上”、“之下”等等(如果有的话)是用于描述性目的的，而并不一定用于描述永久的相对位置。例如，本文使用的术语“之上”、“之下”、“前侧”、“后侧”、“顶部”、“底部”、“上方”、“下方”和“在……上”指的是一个组件、结构或材料相对于设备内的其他引用的组件、结构或材料的相对位置，其中这种物理关系是值得注意的。这些术语在本文中仅用于描述性目的，并且主要是在设备z轴的上下文中使用的，因此可相对于设备的朝向。因此，如果该设备相对于提供的附图的上下文被上下颠倒的话，在本文提供的附图的上下文中在第二材料“之方”的第一材料也可以在第二材料的“之下”。在材料的上下文中，将一种材料设置在另一材料之上或之下可以是直接接触或者可以具有一种或多种居间的材料。另外，将一种材料设置在两种材料之间可以与这两个层直接接触或者可以具有一个或多个居间的层。相比之下，在第二材料“上”的第一材料是与该第二材料直接接触的。在组件装配的上下文中要做出类似的区分。

[0034] 术语“在……之间”可以用于设备的z轴、x轴或y轴的上下文中。在两种其他材料之间的材料可以与这些材料的一者或两者接触，或者该材料可以通过一种或多种居间的材料与其他两种材料间隔开。在两种其他材料“之间”的材料因此可以与其他两种材料的任一者

接触，或者该材料可以通过居间的材料耦合到其他两种材料。在两个其他设备之间的设备可以与这些设备的一者或两者直接连接，或者该设备可以通过一个或多个居间的设备与其他两个设备间隔开。

[0035] 如在整个本说明书中和权利要求中所使用的，由术语“……中的至少一个”或者“……中的一个或多个”联接的项目的列表可以意指列出的术语的任何组合。例如，短语“A、B或C中的至少一个”可以意指A；B；C；A和B；A和C；B和C；或者A、B和C。要指出，附图的具有与任何其他图中的元素相同的标号(或名称)的那些元素可以按与所描述的方式相似的任何方式来操作或工作，但不限于此。

[0036] 此外，本公开中论述的组合逻辑和时序逻辑的各种元素既可涉及物理结构(例如，与门、或门或者异或门)，也可涉及实现逻辑结构(是所论述的逻辑的布尔等同物)的设备的合成的或者以其他方式优化的集合。

[0037] 基于上下文(context-based)的存储器间接分支目标预测示例

[0038] 一些实施例可以为基于上下文的存储器间接分支目标预测提供有利的技术。常规的分支目标预测技术可能主要基于将当前控制流与BPU内部的各种分支预测阵列中缓存的所记录的控制流历史(例如，就最近采取的分支而言)进行匹配。间接分支在指令本身中不包含其目标，而是包含分支目标的位置。可以在特定的寄存器中(又称寄存器间接分支)或者在特定的存储器位置(又称存储器间接分支)中指定间接分支的目标。后者的分支目标预测尤其重要，这不仅因为它在实践中被广泛使用，而且还因为存储器间接分支通常具有更高的时延，因为需要基于目标的指针(本文称为“目标指针”)从存储器位置提取分支目标。

[0039] 处理器的常规BPU可以包括若干不同的分支目标预测器，它们以不同的速度、准确度和优先级处理不同的分支类型。它们可包括缓存过去采取的分支(例如，包括间接分支)的信息的阵列，每个条目通常包括分支IP(分支指令的地址)、分支类型、分支目标、以及有时还包括分支历史向量(也可称为全局历史向量)，该向量是根据最后N个(N为整数)采取的分支的信息计算出来的，作为捕捉的分支之前的控制流历史的数字表示。

[0040] 常规的间接分支目标预测是基于分支目标与控制流历史和/或BPU中记录的先前采取分支的分支IP之间的所缓存的相关性的。然而，在许多情况下，控制流历史(包括分支IP)可能不是预测分支目标的最佳方式。在情况(1)中，具有相同控制流历史/分支IP的间接分支可能不会导致相同的分支目标。例如，充当聚合调度器调用多个子例程的间接分支，在每次其被执行时可能具有相同的最近控制流历史导向这同一个分支，但实际上调用的是具有不同目标的不同函数指针。在情况(2)中，具有不同控制流历史/分支IP的间接分支在其使用相同目标指针时可能导致相同的分支目标，例如，不同代码位置处的两个间接分支调用相同的函数指针，非常有可能去到同一个目标(例如，从C代码中的不同位置通过动态链接的调用表调用同一个外部函数)。基于使用相同目标指针但具有不同分支历史/IP的其他分支，常规的间接分支预测技术可能会在情况(1)中导致误预测，并且在情况(2)中导致无法提供对分支的预测。

[0041] 与使用分支目标与控制流历史和/或分支IP之间的相关性作为预测标准的常规间接分支目标预测技术不同，一些实施例可提供基于上下文的存储器间接分支目标预测技术，该技术可利用分支目标与目标指针之间的相关性来预测存储器间接分支的目标。因为首先，鉴于存储在存储器中的分支目标(大多以函数指针的形式)在实践中不被频繁更新

(例如,在许多情况下,函数指针在初始化之后是只读的),目标指针(例如,持有目标的存储器地址)与目标本身具有非常强的相关性;其次,目标指针在提取目标之前总是可用的,并且不依赖于目标取得的高时延存储器访问。仅解决目标指针涉及的时延比解决间接分支目标要少得多,这为在目标指针可用的时刻预测分支目标提供了良好的性能激励。因此,当常规的基于控制流历史的分支目标预测失败(例如,不提供预测或提供误预测)时,这样的实施例可有利地有效预测存储器间接分支的目标。鉴于常规的间接分支目标预测技术无法妥善处理的上述情况在实践中非常常见,从而基于上下文的存储器间接分支预测技术的一些实施例可通过提高间接分支目标预测的准确性和覆盖范围来改善处理器性能。在一些实施例中,基于上下文的存储器间接分支预测器可作为补充预测器,并且与常规预测器一起并行工作。在一些实施例中,当有多于一个预测时,鉴于实践中目标指针与存储器间接分支的目标值之间的强相关性,基于上下文的预测器可能比其他预测器具有更高的优先级。

[0042] 参考图1,集成电路10的一个实施例可以包括分支目标预测器12,以为一个或多个指令提供分支目标预测。分支目标预测器12包括电路13,该电路被配置为识别一个或多个指令中的存储器间接分支,并且基于存储器间接分支的上下文来提供存储器间接分支的预测目标。例如,存储器间接分支的上下文可以对应于存储器间接分支的目标指针。分支目标预测器12的一些实施例还可以包括第二电路14,该第二电路14被配置为在基于存储器间接分支的上下文的预测和来自一个或多个其他间接分支目标预测器的预测之间选择存储器间接分支的预测目标。例如,相对于来自一个或多个其他间接分支目标预测器的预测,第二电路14可以向基于存储器间接分支的上下文的存储器间接分支的预测目标提供更高的优先级。

[0043] 在一些实施例中,电路13可以被配置为维持数据结构,以基于先前采取的存储器间接分支的上下文将目标指针信息与目标信息相关联,以确定待预测的间接分支的目标指针是否与存储在该数据结构中的任何目标指针信息相匹配,并且如果确定是,则基于该数据结构中的与匹配的目标指针信息相关的目标信息来提供待预测的存储器间接分支的预测目标。例如,该数据结构可以包括条目的阵列,每个条目包括由相应的目标指针信息标记的分支目标信息,以缓存目标指针和分支目标之间的相关性。作为待预测的间接分支的上下文和阵列中的条目之间的匹配标准的标签,可以是目标指针的位的全部或一部分,或者是从目标指针的位的全部或一部分计算出的散列值。电路13还可以被配置为计算待预测的存储器间接分支的标签值,并且确定所计算的标签值是否与阵列的有效条目的标签相匹配,并且如果确定是,则基于来自具有匹配标签的阵列的相应条目的目标信息来提供存储器间接分支的预测目标。例如,阵列的条目可包括诸如以下字段:从存储器间接分支的目标指针信息计算出的标签字段,具有与目标指针相关的目标信息的目标字段,指示条目的有效性的有效性字段,以及存储条目的使用信息以用于替换选择的使用字段(例如,实现“最近最少使用”(Least Recently Used,LRU)算法的“年龄位”的字段)。例如,当电路13在其阵列中没有发现匹配,并且未能为存储器间接分支提供目标预测时,将基于这个存储器间接分支执行和引退之后的上下文在阵列中填充新的条目,如果没有空的或无效的条目可用则替换基于使用信息选择的条目。当电路13发现匹配并且为存储器间接分支提供目标预测,但在从存储器提取实际目标之后被确定为误预测时,现有条目中的分支目标也可被更新。在一些实施例中,当启用虚拟寻址以提供不同虚拟地址空间之间的隔离时,电路13还可以

被配置为基于其目标指针和与地址转化有关的控制寄存器(例如,英特尔x86架构中的CR3)的预定位两者来为存储器间接分支计算条目的标签值。

[0044] 分支目标预测器12、电路13和/或第二电路14的实施例可以被包含到处理器中,该处理器包括例如核心990(图11B)、核心1102A-N(图13、图17)、处理器1210(图14)、协处理器1245(图14)、处理器1370(图15-图16)、处理器/协处理器1380(图15-图16)、协处理器1338(图15-图16)、协处理器1520(图17)、和/或处理器1614、1616(图18)。具体地,分支目标预测器12、电路13和/或第二电路14的实施例可以被包含到分支预测单元932(图11B)中。

[0045] 参考图2A至2C,方法20的一个实施例可以包括在方框21处由分支预测单元为一个或多个指令提供分支目标预测,在方框22处识别该一个或多个指令中的存储器间接分支,并且在方框23处基于存储器间接分支的上下文提供该存储器间接分支的预测目标。例如,在方框24,存储器间接分支的上下文可以对应于方框24的存储器间接分支的目标指针。方法20的一些实施例还可以包括在方框25处在基于存储器间接分支的上下文的存储器间接分支的预测目标和一个或多个其他间接分支目标预测之间进行选择,以及在方框26处相对于一个或多个其他间接分支目标预测,向基于存储器间接分支的上下文的存储器间接分支的预测目标提供更高的优先级。

[0046] 方法20的一些实施例还可以包括在方框27处维持数据结构以将目标指针信息与目标信息相关联,在方框28处确定待预测的存储器间接分支的目标指针是否与存储在数据结构中的目标指针信息相匹配,并且如果确定是,则在方框29处基于数据结构中的与匹配的目标指针信息相关的目标信息来提供存储器间接分支的预测目标。例如,在方框30处,该数据结构可以包括条目的阵列,其中每个条目包括由相应的目标指针信息标记的目标信息,以缓存目标指针和分支目标之间的相关性,并且该方法20还可以包括在方框31处基于其目标指针的一个或多个预定位为存储器间接分支计算标签值,在方框32处确定所计算的标签值是否与阵列的有效条目的标签相匹配,并且如果确定是,则在方框33处基于来自具有匹配标签值的条目的目标信息提供存储器间接分支的预测目标。例如,在方框34处,阵列的条目可以包括从存储器间接分支的目标指针信息计算出的标签字段,具有与目标指针相关的目标信息的目标字段,指示条目的有效性的有效性字段,以及存储条目的使用信息以用于替换选择的使用字段。方法20的一些实施例还可以包括在方框35处,当虚拟寻址被启用时,基于其目标指针和与地址转化有关的控制寄存器的预定位两者来为存储器间接分支计算标签值。

[0047] 参考图3,电子装置40的一个实施例可以包括前端单元41,以提取和解码一个或多个指令,以及与前端单元41通信耦合的执行单元42,以执行解码的一个或多个指令并且向前端单元41提供信息。前端单元41可以包括分支预测单元43,以为一个或多个指令提供分支预测和分支目标预测信息,分支预测单元43包括与分支预测单元43通信耦合的基于上下文的存储器间接分支预测器44,基于上下文的存储器间接分支预测器44包括以下电路,该电路识别一个或多个指令中的存储器间接分支,并且基于存储器间接分支的目标指针提供存储器间接分支的预测目标。在一些实施例中,分支预测单元43还可包括一个或多个其他间接分支预测器,以为间接分支提供目标预测。分支预测单元43可以被配置为在来自基于上下文的存储器间接分支预测器44的预测目标和来自一个或多个其他间接分支目标预测器的其他预测目标之间进行选择。例如,分支预测单元43可以被配置为:相对于来自一个或

多个其他间接分支目标预测器的其他预测目标,向来自基于上下文的存储器间接分支预测器44的预测目标提供更高的优先级。

[0048] 在一些实施例中,基于上下文的存储器间接分支预测器44的电路还可以被配置为维持数据结构以将目标指针信息与目标信息相关联,确定待预测的存储器间接分支的目标指针是否与存储在数据结构中的目标指针信息相匹配,并且如果确定是,则基于数据结构中的与匹配的目标指针信息相关的目标信息来提供存储器间接分支的预测目标。例如,数据结构可以包括条目的阵列,其中每个条目包括由相应的目标指针信息标记的目标信息,以缓存目标指针和分支目标之间的相关性,并且基于上下文的存储器间接分支预测器44的电路还可以被配置为基于其目标指针的一个或多个预定位为存储器间接分支计算标签值,确定所计算的标签值是否与阵列的有效条目的标签相匹配,并且如果确定是,则基于来自相应条目的目标信息来提供存储器间接分支的预测目标。例如,阵列的条目可以包括从存储器间接分支的目标指针信息计算出的标签字段,具有与目标指针相关的目标信息的目标字段,指示条目的有效性的有效性字段,以及存储条目的使用信息以用于替换选择的使用字段。在一些实施例中,基于上下文的存储器间接分支预测器44的电路还可以被配置为在虚拟寻址被启用时基于其目标指针和与地址转化有关的控制寄存器的预定位两者来为存储器间接分支计算标签值。

[0049] 前端单元41、执行单元42、分支预测单元43和/或基于上下文的存储器间接分支预测器44的实施例可以被包含到处理器中,该处理器包括例如核心990(图11B)、核心1102A-N(图13、图17)、处理器1210(图14)、协处理器1245(图14)、处理器1370(图15-图16)、处理器/协处理器1380(图15-图16)、协处理器1338(图15-图16)、协处理器1520(图17)、和/或处理器1614、1616(图18)。具体地,基于上下文的存储器间接分支预测器44的实施例可以被包含到分支预测单元932(图11B)中。

[0050] 参考图4,控制流48的一个实施例图示了调遣器例程通过调用表来调用不同子例程的示例情况。在这种情况下,存储器间接分支(例如本示例中的“call QWORD PTR[rax]”一行)可以使用不同的目标指针,并且在不同的调用中前往不同的分支目标。例如,在一次调用中,间接分支可以基于目标指针从调用表的第M个条目(M是整数)提取其目标,并且分支到子例程M(用实线图示),而在另一次调用中,间接分支可以基于目标指针从调用表的第N个条目(N是与M不同的整数)提取其目标,并且分支到子例程N(用虚线图示)。因为在调遣器的不同调用中,间接分支具有相同的分支IP,并且很可能在间接分支之前具有相同的分支历史,所以基于分支历史和/或分支IP的常规间接分支预测器在间接分支去到与上次调用不同的目标时,几乎总是会误预测。基于上下文的存储器间接分支预测器的一些实施例可有利地处理这种情况,因为目标预测不是基于分支历史或分支IP的,而是基于目标指针和分支目标之间的相关性的,这可有效地在不同的调用中基于每次的目标指针来预测不同的目标。

[0051] 参考图5,控制流50的一个实施例图示了一种示例情况,其中同一个外部函数被不同的存储器间接分支通过导入地址表(import address table,IAT)从不同的代码位置调用。在这种情况下,对于指令“call QWORD PTR[IAT_entry_N]”,在两个不同的行有两个间接分支。第一个将被称为分支#1,第二个将被称为分支#2。显然,分支#1和分支#2具有不同的分支IP和控制流历史(分支历史),而它们使用相同的目标指针并且去往同一目标。虽然

分支#1和分支#2具有相同的目标指针和相同的分支目标,但基于分支历史和/或分支IP的常规间接分支预测器不能基于分支#1的执行来预测分支#2的目标,因为它们的分支历史和分支IP不匹配。另一方面,基于上下文的存储器间接分支预测器的一些实施例可有利地处理这种情况,基于目标指针和分支#1创建的目标之间的相关性来预测分支#2的目标。类似于这个示例的情况在实践中可能非常普遍,包括对作为存储器间接分支执行的动态链接函数(例如,应用程序接口(application program interface,API)函数)的大多数调用。

[0052] 可以利用任何适当的数据结构来缓存目标指针和分支目标之间的适当相关性。在一些实施例中,基于上下文的存储器间接分支预测器包含阵列,该阵列缓存目标指针和存储器间接分支的目标之间的相关性。该阵列可以使用目标指针的全部或部分位,或者从目标指针的全部或部分位计算出的散列值,作为其用于预测匹配的入口标签。以位为单位的标签字段的大小可取决于不同的资源和性能考虑而变化。阵列条目还应当包含带有缓存的分支目标的目标字段,其形式是目标位的全部或一部分,这取决于不同的资源和性能考虑。阵列还应当包含有效性字段/位,以指示出该条目的有效性,以及使用字段,来存储该条目的使用信息,以用于替换选择目的。整个阵列可以被设计成具有简单标签匹配的完全关联阵列,或者可以被设计成集合关联阵列以优化查找/匹配过程。在后一种情况下,目标指针的一些位可被用作集合索引。

[0053] 参考图6,基于上下文的存储器间接分支预测阵列的一个实施例可以包括被组织为集合关联阵列的多个条目。在图示的示例中,目标指针的位范围Bit[17:12]为阵列提供六(6)位集合索引。基于上下文的存储器间接分支预测器条目的示例可以包含但不限于以下字段:

[0054] 标签字段:从目标指针计算

[0055] 有效位:条目有效性的指示符

[0056] LRU字段:条目使用的指示符

[0057] 目标字段:缓存的分支目标

[0058] 如图所示,该示例提供了64集合乘8路的集合关联阵列,总共有五百一十二(512)个六十二(62)位条目。目标指针的位范围Bit[17:12]被用作集合索引,而目标指针的位范围Bit[28:19]和位范围Bit[11:2]的XOR(二进制操作“异或”)结果被用作10位标签值。因为在大多数情况下,函数指针在存储器中应当至少是4字节对齐的,所以在预测中可以忽略目标指针的两个最低有效位。在一些实施例中,当启用虚拟寻址时,标签计算还可以包括与地址转化有关的控制寄存器的某些位(例如,INTEL x86架构中的CR3),以提供虚拟地址空间之间的隔离,以满足性能和安全性考虑。

[0059] 基于上下文的循环分支预测

[0060] 一些实施例可以为基于上下文的循环分支预测提供有利的技术。循环是一种基本的代码结构,在所有种类的计算机程序和编程语言中都是非常常用的。在汇编代码级别,循环通常是由条件分支指令实现的。这里的“循环分支”是指决定循环是将继续下一个循环迭代还是从循环中退出的条件分支指令。

[0061] 与分支目标预测不同,分支预测以“采取”或“不采取”的形式预测条件分支(例如,条件跳转指令)的方向。在大多数情况下,循环分支(其本身也是条件分支)的分支预测的实现方式可以与任何其他条件分支(例如,实现“if/then”语句的条件分支)的方式相同。BPU

可以利用所有条件分支的一般分支预测器来预测循环分支。

[0062] 常规的分支预测器主要基于其来自过去调用的执行历史,以“采取”或“不采取”的形式来预测条件分支的方向。通常,计数器或状态机可用于跟踪条件分支的执行,并且还生成其预测结果。计数器或状态机可在“(强烈)采取”和“(强烈)不采取”之间具有多个状态,这些状态可在每次根据所解决的分支方向执行相应的条件分支时被更新:假设计数器或状态机不处于“饱和”状态,那么如果分支被解决为采取,则计数器或状态机将朝着“(强烈)采取”被更新到邻近状态,如果分支被解决为不采取,则计数器或状态机将朝着“(强烈)不采取”被更新到邻近状态。未解决的条件分支的预测是从跟踪该分支的相应计数器或状态机的当前状态生成的(利用将状态映射到二进制“采取”或“不采取”预测的某个算法),如果预测器中有这样的信息可用的话。

[0063] 分支预测器通常包含一个或多个阵列,其每个条目存储了相应条件分支的分支预测信息。条目通常包括:(1)标签字段,用于将阵列中的条目与待预测的条件分支相匹配;(2)预测字段,包含按照数字表示形式的“采取”或“不采取”而言的预测分支方向(例如,上述的计数器或状态机可被实现在这个字段中);以及其他字段,用于保存有效性、使用信息,等等。

[0064] 不同类型的分支预测器可以使用不同的方式来标记其阵列中的条目。一种常见类型的分支预测器(下文将称为类型I)可简单地使用分支IP位的全部或一部分(或者从分支IP位的全部或一部分计算出的散列值)作为条目标签,基本上使用分支指令的地址作为预测输入条件。另一种常见类型的分支预测器(下文将称为类型II)可以使用分支历史向量(也可称为全局历史向量、全局分支历史向量)来标记条目。分支历史向量是基于最近采取的分支的信息的最近控制流的数字表示。其通常被实现为从最近N个(N为整数)采取的分支的分支信息(例如,分支地址、分支目标地址,等等)计算出的滚动散列。类型II分支预测器可以只使用分支历史向量,或者使用分支历史向量和分支IP两者,以用于标记阵列条目。

[0065] 两种类型的常规分支预测器在预测循环分支方面都存在问题。类型I分支预测器总是会错误预测循环退出时的循环分支,因为循环退出时的循环分支的方向与所有先前循环迭代不同,这将诱发性能惩罚,而如果正确预测循环分支,则可避免这种惩罚。使用或包括分支历史向量作为预测输入条件的类型II分支预测器在处理循环分支时也有困难:(1)类型II分支预测器是基于历史的,从而当循环被第一次执行时,不能正确预测循环退出时的循环分支;(2)类型II分支预测器是基于历史的,从而当最终循环计数(即,循环将执行的迭代的总数)在一次执行与另一次执行之间变化时,则不能正确预测循环退出时的循环分支;(3)类型II分支预测器中使用的分支历史向量涵盖最近N个采取的分支的历史(例如,其中N是整数常数,通常小于100),因此当最终循环计数大于N时,类型II分支预测器不能正确地预测循环退出时的循环分支;(4)由于在大多数情况下,循环分支在每次循环迭代时具有不同的分支历史向量,所以程序循环可能会过度填充使用分支历史向量作为条目标签的类型II分支预测器(在每次迭代中添加新的条目),清除那些对预测其他分支有用的较旧条目,从而降低预测效率。

[0066] 除了一般对于条件分支预测最常使用的分支预测器以外,还有其他针对循环分支的分支预测技术,包括循环退出预测器(Loop Exit Predictor,LEP)和冻结历史预测器(Frozen History Predictor,FHP)。循环退出预测器(LEP)是一种基于计数的循环分支预

测技术。LEP为要预测的每个程序循环获取最终循环计数(即,循环将执行的迭代的总数)。最终循环计数被用来预测循环分支指令的分支行为。获取最终循环计数的典型方法包括:(1)先前循环发生:从同一循环的先前执行中捕捉最终循环计数(基于历史的循环计数预测);(2)运行时循环检测:通过识别可揭示存储循环总迭代数的位置的某些指令序列,使用指令流的运行时硬件分析;以及(3)具有架构支持的循环计数位置:如果在循环执行之前知道循环计数的位置,则通过具体识别循环计数值的位置来确定最终循环计数。

[0067] FHP基于冻结历史向量和先前观察到的捕捉的最终循环计数,为循环分支提供分支预测。冻结历史向量是进入或检测到循环时的分支历史向量的快照,它保留了先前的循环分支历史。一旦进入循环,就可以跟踪循环迭代计数。当循环退出被解决时,FHP捕捉快照的分支历史向量与循环退出时的最终循环迭代计数之间的相关性。冻结的历史和对于循环分支指令进行预测时的当前迭代计数被FHP用来与捕捉的相关性进行比较以进行预测。FHP的一个典型实现方式可以具有冻结历史和迭代跟踪表,当确定当前循环迭代计数等于捕捉的循环退出迭代计数时,FHP预测循环将被退出。

[0068] 一些实施例提供了一种基于上下文的循环分支预测技术,该技术不仅可有利地解决常规分支预测器在预测循环分支时存在的上述问题,而且当与现有的循环分支预测技术相比时,还具有根本的差异和优势。在一些实施例中,基于上下文的循环分支预测包括两个主要特征:(1)为循环分支使用专用的条件分支指令,这些指令与汇编代码/机器代码级别的其他条件分支不同;(2)基于上下文的硬件循环分支预测器,其可以识别循环分支并且提供基于推测性循环迭代外推的分支预测。

[0069] 一般而言,专用循环分支指令应包括分支目标和循环退出条件两者。在一些实施例中,这种指令可以具有如下的一般形式:

[0070] LOOPBRANCHcc<branch_target>,<loop_control_variable>,<loop_exit_value>

[0071] 其中“LOOPBRANCHcc”是采取其一般形式的指令操作码,<branch_target>是指定分支目标的操作对象,<loop_control_variable>是指定循环控制变量的操作对象,通常在每次循环迭代中作为循环条件的一部分被更新和检查,<loop_exit_value>是指定值的操作对象,该值通常是常数,<loop_control_variable>被与该值进行比较以确定循环是否将退出。“LOOPBRANCHcc”中的“cc”代表在<loop_control_variable>和<loop_exit_value>之间需要满足的条件,以便让循环继续(循环分支被采取并且跳转到<banch_target>),类似于英特尔x86指令集中的条件跳转(例如,“LOOPBRANCHNE”表示“如果<loop_control_variable>不等于<loop_exit_value>则继续循环”,“LOOPBRANCHG”表示“如果<loop_control_variable>大于<loop_exit_value>则继续循环”)。当<loop_control_variable>和<loop_exit_value>之间的指定条件不被满足时,循环将退出(循环分支不被采取并且继续到下一指令)。<loop_control_variable>可以是显式操作对象(例如,寄存器或存储器位置),或者可以是隐式操作对象(例如,指令中未指定的预定义寄存器)。取决于实现方式,<loop_control_variable>可以在执行时被循环分支指令本身递增或递减,或者留待其他指令来更新。<loop_exit_value>也可以是显式操作对象(例如,寄存器、存储器位置或者作为常数值的立即数操作对象)、隐式操作对象(例如,指令中未指定的预定义寄存器)、或者不存在,在这种情况下,<loop_control_variable>将被与0比较。

[0072] 一个现有的示例是英特尔x86指令集中的“LOOP”指令。“LOOP”指令具有指定其分

支目标(作为相对偏移量)的显式操作对象,并且还具有指定循环控制变量的隐式操作对象,在此情况下是ECX寄存器中的“循环计数器”(32位模式)。每次执行“LOOP”指令时,ECX寄存器中的值首先被递减,然后被与0比较:如果不相等,则循环继续(“LOOP”分支被采取并且跳转到其分支目标),否则,循环退出(“LOOP”分支不被采取并且继续到下一指令)。

[0073] “LOOP”指令可以被看作是前面提到的“LOOPBRANCHcc”指令的具体实现,它相当于:

[0074] LOOPBRANCHNE<branch_target>,ECX,0

[0075] 这里“LOOPBRANCHNE”是指“如果ECX不等于0,则继续循环”,在这种情况下,“LOOPBRANCHNE”指令在循环条件检查之前的每次执行中也会递减<loop_control_variable>(在ECX中)。

[0076] 对计算机程序中的循环使用专用的循环分支指令可以在编译器的支持下实现(例如,编译器插件)。专用的循环分支指令可以是现有的指令,如英特尔x86指令集中的“LOOP”指令,或者是新添加的指令,作为指令集扩展,其一般形式为上述的“LOOPBRANCHcc”指令。编译器支持确保当程序被编译时,可以使用专用循环分支指令来编译的任何循环都应被用这种指令来编译。在相对罕见的情况下,如果循环具有不能用循环分支指令来编译的复杂循环退出条件,则允许使用一般条件分支来编译循环。在这种情况下,实现循环的一般条件分支将被预测为任何其他的一般条件分支,这不会影响基于上下文的循环分支预测的有效性和性能优势。

[0077] 作为示例,C语言代码中的具有100次迭代的“for循环”可以具有如下的格式:

[0078] …

[0079] for(int i＝0;i<100;i++){

[0080] //循环内容

[0081] …

[0082] }

[0083] …

[0084] 在大多数情况下,当前的编译器使用一般条件分支来实现汇编语言/机器代码级别的循环。以英特尔x86指令集为例,上面的“for循环”可以被编译为:

[0085] …

[0086] MOV DWORD PTR[i_addr],0

[0087] JMP loop_control

[0088] loop_start:

[0089] ;loop content

[0090] …

[0091] INC DWORD PTR[i_addr]

[0092] loop_control:

[0093] CMP DWORD PTR[i_addr],0x64

[0094] JL loop_start

[0095] …

[0096] 这里,i_addr是循环控制变量i的存储器地址。在基于上下文的循环分支预测的情

况下,这个“for循环”可以在编译器的支持下用专用的循环分支指令来进行编译。例如,当使用英特尔x86指令集中现有的“LOOP”指令时,上面的“for循环”可以被编译为:

[0097] …

[0098] MOV ECX,0x65

[0099] JMP loop_control

[0100] loop_start:

[0101] ;loop content(which does not clobber ECX)

[0102] …

[0103] loop_control:

[0104] LOOP loop_start

[0105] …

[0106] 在这种情况下,“LOOP”指令按照其定义,在ECX寄存器中具有循环控制变量(循环计数器),每次执行“LOOP”指令时,它将被递减1,然后被与0比较:如果等于0,则“LOOP”指令将退出循环并且继续到下一指令。这种“for循环”也可用循环分支指令来编译,其更一般形式为早前描述的“LOOPBRANCHcc”指令:

[0107] …

[0108] MOV DWORD PTR[i_addr],0

[0109] JMP loop_control

[0110] loop_start:

[0111] ;loop content

[0112] …

[0113] INC DWORD PTR[i_addr]

[0114] loop_control:

[0115] LOOPBRANCHL loop_start,DWORD PTR[i_addr],0x64

[0116] …

[0117] 这里,“LOOPBRANCHL”指令如前所述是专用于实现循环的条件分支指令,它具有三个操作对象:“loop_start”作为分支目标,“DWORD PTR[i_addr]”作为循环控制变量,“0x64”(十进制的100)作为循环退出值。“LOOPBRANCHL”意思是“如果循环控制变量(DWORD PTR[i_addr])小于循环退出值(0x64),则继续循环”,在这种情况下,它跳转到“loop_start”(下一个循环迭代),否则循环退出。在这个示例中,“LOOPBRANCHL”指令本身并不像现有的“LOOP”指令那样递增或递减循环控制变量,而是将其留给其他指令来更新。

[0118] 在一些实施例中,基于上下文的循环分支预测可以包括硬件循环分支预测器,该预测器可以识别循环分支并且通过推测性循环迭代外推来生成预测。基本上,它利用了这样一个事实:在大多数情况下,程序循环是由循环控制变量来控制的,该变量在每次循环迭代中被以固定的跨度更新(递增或递减),并且被与某个值进行比较,以确定循环是否退出。基于上下文的循环分支预测器可以推测性计算下一次循环迭代的循环控制变量,检查循环是否会在下一次迭代时退出,并且为循环分支生成相应的分支预测。下一次迭代的循环控制变量的推测性计算可以基于当前和先前迭代的循环控制变量的值通过简单的外推来进行,假设它将被以固定的跨度更新:

[0119] (循环控制变量)$|_{下一迭代}$=(循环控制变量)$|_{当前迭代}$+跨度

[0120] 考虑到

[0121] 跨度=(循环控制变量)$|_{当前迭代}$-(l循环控制变量)$|_{先前迭代}$

[0122] 于是

[0123] (循环控制变量)$|_{下一迭代}$=(循环控制变量)$|_{当前迭代}$

[0124] +[(循环控制变量)$|_{当前迭代}$-(循环控制变量)$|_{先前迭代}$]

[0125] 这与如下相同：

[0126] (循环控制变量)$|_{下一迭代}$

[0127] =2×(循环控制变量)$|_{当前迭代}$-(循环控制变量)$|_{先前迭代}$

[0128] 基于上下文的循环分支预测器只需要保存来自先前循环迭代的循环控制变量值。然后,它可以计算出下一次迭代的推测性循环控制变量值,并且通过使用为下一次迭代计算的推测性循环控制变量值检查循环退出条件,来预测循环分支是否会在下一次迭代时退出循环。这可以简单地通过上述的专用循环分支指令的实现来实现：

[0129] LOOPBRANCHcc<branch_target>,<loop_control_variable>,<loop_exit_value>

[0130] 这允许了基于上下文的循环分支预测器通过其指令操作码轻松识别循环分支,并且还提供关于循环控制变量和循环退出条件(包括循环退出值)的明确信息。

[0131] 典型的基于上下文的循环分支预测器可以是硬件预测器,具有多个条目的阵列,每个条目对应一循环分支指令。条目可以包含但不限于以下字段：(1)用于识别循环分支的标签字段,可以基于分支IP来计算；(2)“先前循环控制变量”字段,用于存储来自先前循环迭代(最后执行)的循环控制变量值；(3)预测字段,具有按照数字形式的“采取”或“不采取”的预测分支方向；(4)有效性字段,指示出该条目是否在跟踪活跃循环分支(活跃意味着循环已经开始并且尚未退出)；(5)使用字段,存储用于条目替换选择的使用信息。

[0132] 基于上下文的循环分支预测器的典型工作流程具有以下步骤：

[0133] 1.识别在其预测范围内的循环分支,这可以在指令解码级通过专用循环分支指令的实现来轻松完成。

[0134] 2.为待预测的循环分支计算标签,并且在预测阵列中寻找与待预测的循环分支具有相同标签的匹配条目,如果没有现有的匹配条目则填充新的条目,如果有可用的则使用空的或无效的条目,或者基于存储在使用字段中的使用信息,用某些替换算法覆写现有的条目。

[0135] 3.从相应条目的预测字段提供预测结果,按照“采取”或“不采取”的形式,该预测字段是在循环分支的先前执行(先前循环迭代)中生成的。当循环分支在活跃循环中被第一次执行时(第一次循环迭代),预测默认为“采取”(继续循环)。

[0136] 4.基于当前循环控制变量和保存的先前循环控制变量的值,通过外推法为下一次循环迭代计算循环控制变量的预测值。

[0137] 5.通过使用下一次迭代的循环控制变量的推测值(在步骤4中计算)检查循环退出条件来生成循环分支的下一次执行(下一次循环迭代)的预测结果,并且相应地更新相应条目的预测字段。可简单地从专用循环分支指令的操作码和操作对象获得循环退出条件和循环退出值。基于上下文的循环分支预测器可作为内部硬件逻辑或者通过使用通用执行单元来实现这一检查。

[0138] 6.当相应的循环分支被解决为“不采取”(循环已退出)时,将条目的有效性字段更新为“无效”,表明该循环不再活跃,并且条目无效并且可以被覆写掉。

[0139] 与常规的条件分支预测器相比,基于上下文的循环分支预测器可有利地处理循环分支的分支预测。如上所述,基于其分支IP来识别和标记分支的类型I分支预测器,在循环退出时总是错误预测循环分支,而基于其分支历史向量来识别和标记分支的类型II分支预测器,由于各种限制,在许多情况下会错误预测循环退出时的循环分支。作为比较,基于上下文的循环分支预测器可以通过推测性循环迭代外推来正确预测循环退出(以及其他循环迭代)。此外,由于循环分支的预测可以由基于上下文的循环分支预测器来处理,所以常规的分支预测器不再需要将循环分支预测为一般的条件分支(这用上述的专用循环分支指令是很容易实现的),从而避免了循环分支可能通过在每次循环迭代时在其阵列中添加新的分支条目而过度填充类型II预测器的问题。因此,基于上下文的循环分支预测不仅可以提高循环分支的预测准确度,还可以提高常规分支预测器预测一般条件分支的效率。

[0140] 基于上下文的循环分支预测的实施例与其他循环预测技术(例如上述LEP和FHP)相比,也具有根本的差异和各种优势。LEP和FHP都是基于计数的循环分支预测:循环分支的预测是基于预测的最终循环计数的,该计数是由预测器在预测循环分支之前确定或捕捉的。预测的最终循环计数要么来自同一循环的执行历史(先前实例)(例如,对于FHP以及LEP的某些实现方式),要么是从对指令流或架构支持的复杂运行时硬件分析获得的(例如,LEP的其他实现方式)。与FHP和LEP不同,基于上下文的循环分支预测既不涉及最终的循环计数,也不涉及来自先前实例的循环的执行历史,而是基于推测性的循环迭代外推:在每次预测时,预测器会消耗来自先前循环迭代的预测,并且分支指令为下一个循环迭代生成预测。

[0141] 与现有的循环预测技术(例如,FHP和LEP)相比,基于上下文的循环分支预测的实施例有几个优点。首先,由于基于上下文的循环分支预测不依赖于来自先前实例的循环的执行历史(如FHP和LEP的某些实现方式),因此它可以更好地处理循环的首次执行的预测。其次,由于基于上下文的循环分支预测不是基于预测的最终循环数的,因此它可以更好地处理最终循环计数在同一循环的不同执行实例之间变化或者在循环的执行期间被更新的情况。

[0142] 与常规的分支预测器和其他现有的循环预测器(例如,FHP和LEP)相比,基于上下文的循环分支预测的实施例还具有另一个优势,即其关于预测阵列的所需足迹大小。为了做出正确的预测,上述所有其他预测器都需要在待预测的循环开始执行之前在其预测阵列中有相应的条目存在。因此,一般而言,这些预测器需要有较大的阵列大小,以保持尽可能多的条目(通常约为数百个),以提高预测效率和覆盖范围。另一方面,基于上下文的循环分支预测器使用循环迭代外推,这只需要在相应的循环处于活跃时保持条目有效(一旦循环退出,条目就被标记为无效),在这种情况下,条目的数目只需要足以覆盖(大多数情况下)同时处于活跃的循环的数目(例如嵌套循环或来自不同执行上下文的循环)。取决于实现要求,在基于上下文的循环分支预测器中,8条目或16条目的阵列对于实践中的大多数情况应是足够的。

[0143] 参考图7,集成电路110的一个实施例可以包括分支预测器112,以为一个或多个指令预测条件分支是被采取还是不被采取,该分支预测器112包括电路113,以识别一个或多个指令中的循环分支指令,并且基于循环分支指令的上下文为该循环分支指令提供分支预

测。例如,循环分支指令的上下文可以对应于来自循环分支指令的当前迭代和循环分支指令的先前迭代的循环分支指令的操作对象值。

[0144] 在一些实施例中,电路113可以被配置为推测性地计算循环分支指令的下一迭代的循环控制变量,并且基于计算出的循环控制变量是否指示出循环分支指令将在循环分支指令的下一迭代时退出循环来提供分支预测。例如,电路113可以被配置为基于来自循环分支指令的当前迭代和循环分支指令的先前迭代的各个循环控制变量,为循环分支指令的下一迭代外推循环控制变量。

[0145] 在一些实施例中,电路113可以被配置为保存来自循环分支指令的先前迭代的循环控制变量值。例如,电路113可以被配置为存储各自与各个循环分支指令相对应的条目的阵列。在一些实施例中,阵列的条目可以包括由电路113计算的用于识别循环分支的标签字段、用于存储循环控制变量的先前值的字段、具有按照“采取”或“不采取”而言的预测分支方向的预测字段、用于指示条目的有效性的有效性字段、以及用于存储用于条目替换选择的使用信息的使用字段。

[0146] 分支目标预测器112和/或电路13的实施例可以被包含到处理器中,该处理器包括例如核心990(图11B)、核心1102A-N(图13、图17)、处理器1210(图14)、协处理器1245(图14)、处理器1370(图15-图16)、处理器/协处理器1380(图15-图16)、协处理器1338(图15-图16)、协处理器1520(图17)、和/或处理器1614、1616(图18)。具体地,分支目标预测器112和/或电路113的实施例可以被包含到分支预测单元932(图11B)中。

[0147] 参考图8A至8C,方法120的一个实施例可以包括在方框121处由分支预测单元处理一个或多个指令,在方框122处识别一个或多个指令中的循环分支指令,并且在方框123处基于循环分支指令的上下文为该循环分支指令提供分支预测。例如,在方框124处,循环分支指令的上下文可以对应于来自循环分支指令的当前迭代和循环分支指令的先前迭代的循环分支指令的操作对象值。方法120的一些实施例还可以包括在方框125处推测性地计算循环分支指令的下一迭代的循环控制变量,并且在方框126处基于计算出的循环控制变量是否指示出循环分支指令将在循环分支指令的下一迭代时退出循环而提供分支预测。例如,方法120可以包括在方框127处基于来自循环分支指令的当前迭代和循环分支指令的先前迭代的各个循环控制变量,为循环分支指令的下一迭代外推循环控制变量。

[0148] 方法120的一些实施例还可以包括在方框128处保存来自循环分支指令的先前迭代的循环控制变量值。例如,方法120可以包括在方框129处存储各自与各循环分支指令相对应的条目的阵列,其中在方框130处,阵列的条目可以包括从分支IP计算出的用于识别循环分支的标签字段、用于存储循环控制变量的先前值的字段、具有按照“采取”或“不采取”而言的预测分支方向的预测字段、用于指示条目的有效性的有效性字段、以及用于存储用于条目替换选择的使用信息的使用字段。

[0149] 参考图9,电子装置140的一个实施例可以包括前端单元141,以提取和解码一个或多个指令,以及与前端单元141通信耦合的执行单元142,以执行该一个或多个指令并且向前端单元141提供信息。前端单元141可以包括分支预测单元143,它可以为一个或多个指令预测条件分支是被采取还是不被采取。分支预测单元143可以包括电路144,以识别一个或多个指令中的循环分支指令,并且基于循环分支指令的上下文为该循环分支指令提供分支预测。例如,循环分支指令的上下文可以对应于来自循环分支指令的当前迭代和循环分支

指令的先前迭代的循环分支指令的操作对象值。

[0150] 在一些实施例中，电路144可以被配置为推测性地计算循环分支指令的下一迭代的循环控制变量，并且基于计算出的循环控制变量是否指示出循环分支指令将在循环分支指令的下一迭代时退出循环来提供分支预测。例如，电路144可以被配置为基于来自循环分支指令的当前迭代和循环分支指令的先前迭代的各个循环控制变量，为循环分支指令的下一迭代外推循环控制变量。

[0151] 在一些实施例中，电路144可以被配置为将来自循环分支指令的先前迭代的循环控制变量值存储在条目的阵列中，每个条目对应于各循环分支指令。例如，阵列的条目包括由电路144从分支IP计算的用于识别循环分支的标签字段、用于存储循环控制变量的先前值的字段、具有按照“采取”或“不采取”而言的预测分支方向的预测字段、用于指示条目的有效性的有效性字段、以及用于存储用于条目替换选择的使用信息的使用字段。

[0152] 前端单元141、执行单元142、分支预测单元143和/或电路144的实施例可以被包含到处理器中，该处理器包括例如核心990(图11B)、核心1102A-N(图13、图17)、处理器1210(图14)、协处理器1245(图14)、处理器1370(图15-图16)、处理器/协处理器1380(图15-图16)、协处理器1338(图15-图16)、协处理器1520(图17)、和/或处理器1614、1616(图18)。具体地，电路144的实施例可以被包含到分支预测单元932(图11B)中。

[0153] 参考图10，基于上下文的循环分支预测器的预测阵列的一个实施例可以包括八(8)个条目。在这个示例性实施例中，x86指令集中的“LOOP”指令被用作专用的循环分支指令。循环分支预测阵列的条目的示例可以包括以下字段：

[0154] 标签字段：从分支IP计算出的字段，来识别用于分支预测的循环分支，在此示例中，它是10位的散列，是通过对位范围Bit[29:20]、Bit[19:10]和Bit[9:0]进行XOR(“异或”的二进制操作)从分支IP的低30位计算出来的；

[0155] 预测字段：按照数字表示形式的“采取”或“不采取”而言的预测分支方向；

[0156] 有效位：对条目的有效性的指示；

[0157] 使用字段：用于存储用于条目替换选择的使用信息，在此示例中，它是实现“最近最少使用”(LRU)算法的3位字段。

[0158] 在这个示例中，存储先前循环控制变量的字段是不存在的，因为循环控制变量(在32位模式中存储在ECX中)被LOOP指令本身以固定且已知的跨度1来递减。因此，在这种情况下不需要保存先前的循环控制变量值来确定跨度。

[0159] 每次“LOOP”指令被提取时，它的分支IP就被用来计算标签，并且通过将计算出的标签与阵列条目的标签相比较来寻找预测阵列中的匹配条目。如果存在匹配，则来自相应条目的预测结果被用作待预测的“LOOP”分支的分支预测。否则，将使用空的或无效的条目填充新的条目，或者如果没有空的或无效的条目可用，则替换最少使用的条目。下一迭代的循环控制变量的推测值可简单地从当前的循环控制变量再外推一个跨度来计算，基本上是在“LOOP”指令的内置递减之后从ECX(在32位模式中)中的值再递减1。当下一迭代的预测循环控制变量(递减后)为0时，“LOOP”分支将被预测为“不采取”(退出循环)，否则将被预测为“采取”(继续循环)。新的预测结果将被更新到相应条目的预测字段中，并且在下一次迭代中被消耗来用于循环分支预测。每当“LOOP”分支被解决为“不采取”(退出循环)时，相应条目的有效位就将被清除，以表明该循环不再活跃，并且该条目可用来被覆写。

[0160] 本领域的技术人员将会明白，各种各样的设备都可以从上述实施例中受益。以下示例性核心架构、处理器和计算机架构是可以有益地纳入本文描述的技术的实施例的设备的非限制性示例。

[0161] 示例性核心架构、处理器和计算机架构

[0162] 可以按不同的方式、为了不同的目的、在不同的处理器中实现处理器核心。例如，这种核心的实现方式可以包括：1)旨在用于通用计算的通用有序核心；2)旨在用于通用计算的高性能通用无序核心；3)主要旨在用于图形和/或科学(吞吐量)计算的专用核心。不同处理器的实现方式可以包括：1)包括旨在用于通用计算的一个或多个通用有序核心和/或旨在用于通用计算的一个或多个通用无序核心的CPU；以及2)包括主要旨在用于图形和/或科学(吞吐量)的一个或多个专用核心的协处理器。这样的不同处理器导致不同的计算机系统架构，这些架构可以包括：1)协处理器在与CPU分开的芯片上；2)协处理器在与CPU相同的封装中、分开的管芯上；3)协处理器与CPU在同一管芯上(在此情况下，这种协处理器有时被称为专用逻辑，例如集成图形和/或科学(吞吐量)逻辑，或者被称为专用核心)；以及4)片上系统，其可以在同一管芯上包括所描述的CPU(有时称为(一个或多个)应用核心或者(一个或多个)应用处理器)、上述的协处理器以及附加的功能。接下来描述示例性核心架构，然后是对示例性处理器和计算机架构的描述。

[0163] 示例性核心架构

[0164] 有序和无序核心框图

[0165] 图11A是根据本发明的实施例图示出示例性有序管线和示例性寄存器重命名、无序发出/执行管线两者的框图。图11B是根据本发明的实施例图示出要被包括在处理器中的有序架构核心的示例性实施例和示例性寄存器重命名、无序发出/执行架构核心两者的框图。图11A-11B中的实线框图示了有序管线和有序核心，而虚线框的可选添加图示了寄存器重命名、无序发出/执行管线和核心。考虑到有序方面是无序方面的子集，将描述无序方面。

[0166] 在图11A中，处理器管线900包括提取级902、长度解码级904、解码级906、分配级908、重命名级910、调度(也称为调遣或发出)级912、寄存器读取/存储器读取级914、执行级916、写回/存储器写入级918、异常处理级922、以及提交级924。

[0167] 图11B示出了处理器核心990包括耦合到执行引擎单元950的前端单元930，并且执行引擎单元950和前端单元930两者都耦合到存储器单元970。核心990可以是精简指令集计算(reduced instruction set computing，RISC)核心、复杂指令集计算(complex instruction set computing，CISC)核心、超长指令字(very long instruction word，VLIW)核心、或者混合或替代核心类型。作为另外一个选项，核心990可以是专用核心，例如，网络或通信核心、压缩引擎、协处理器核心、通用计算图形处理单元(general purpose computing graphics processing unit，GPGPU)核心、图形核心，等等。

[0168] 前端单元930包括分支预测单元932，分支预测单元932耦合到指令缓存单元934，指令缓存单元934耦合到指令转化后备缓冲器(translation lookaside buffer，TLB)936，指令TLB 936耦合到指令提取单元938，指令提取单元938耦合到解码单元940。解码单元940(或解码器)可以对指令解码，并且生成一个或多个微操作、微代码入口点、微指令、其他指令或其他控制信号作为输出，这些微操作、微代码入口点、微指令、其他指令或其他控制信号是从原始指令解码来的，或者以其他方式反映原始指令，或者是从原始指令得出的。可以

利用各种不同的机制来实现解码单元940。合适机制的示例包括但不限于查找表、硬件实现、可编程逻辑阵列(programmable logic array,PLA)、微代码只读存储器(read only memory,ROM),等等。在一个实施例中,核心990包括微代码ROM或其他介质,其为某些宏指令存储微代码(例如,在解码单元940中或者以其他方式在前端单元930内)。解码单元940耦合到执行引擎单元950中的重命名/分配器单元952。

[0169] 执行引擎单元950包括重命名/分配器单元952,重命名/分配器单元952耦合到引退单元954和一组一个或多个调度器单元956。(一个或多个)调度器单元956表示任何数目的不同调度器,包括预留站、中央指令窗口,等等。(一个或多个)调度器单元956耦合到(一个或多个)物理寄存器文件单元958。(一个或多个)物理寄存器文件单元958中的每一者表示一个或多个物理寄存器文件,这些物理寄存器文件中的不同物理寄存器文件存储一个或多个不同的数据类型,例如标量整数、标量浮点、压缩整数、压缩浮点、向量整数、向量浮点、状态(例如,作为要执行的下一指令的地址的指令指针),等等。在一个实施例中,(一个或多个)物理寄存器文件单元958包括向量寄存器单元、写入掩码寄存器单元、以及标量寄存器单元。这些寄存器单元可以提供架构式向量寄存器、向量掩码寄存器、以及通用寄存器。(一个或多个)物理寄存器文件单元958与引退单元954交叠以说明可用来实现寄存器重命名和无序执行的各种方式(例如,利用(一个或多个)重排序缓冲器和(一个或多个)引退寄存器文件;利用(一个或多个)未来文件、(一个或多个)历史缓冲器、以及(一个或多个)引退寄存器文件;利用寄存器图谱和寄存器的池;等等)。引退单元954和(一个或多个)物理寄存器文件单元958耦合到(一个或多个)执行集群960。(一个或多个)执行集群960包括一组一个或多个执行单元962和一组一个或多个存储器访问单元964。执行单元962可以在各种类型的数据(例如,标量浮点、压缩整数、压缩浮点、向量整数、向量浮点)上执行各种操作(例如,移位、加法、减法、乘法)。虽然一些实施例可以包括专用于特定功能或功能集合的若干个执行单元,但其他实施例可只包括一个执行单元或者全部执行所有功能的多个执行单元。(一个或多个)调度器单元956、(一个或多个)物理寄存器文件单元958和(一个或多个)执行集群960被示为可能是多个,因为某些实施例为某些类型的数据/操作创建单独的管线(例如,标量整数管线、标量浮点/压缩整数/压缩浮点/向量整数/向量浮点管线、和/或存储器访问管线,它们各自具有其自身的调度器单元、物理寄存器文件单元和/或执行集群——并且在单独的存储器访问管线的情况下,实现了某些实施例,其中只有此管线的执行集群具有(一个或多个)存储器访问单元964)。还应当理解,在使用分开管线的情况下,这些管线中的一个或多个可以是无序发出/执行,并且其余的是有序的。

[0170] 存储器访问单元964的集合耦合到存储器单元970,存储器单元970包括数据TLB单元972,数据TLB单元972耦合到数据缓存单元974,数据缓存单元974耦合到第2级(L2)缓存单元976。在一个示例性实施例中,存储器访问单元964可以包括加载单元、存储地址单元、以及存储数据单元,它们中的每一者耦合到存储器单元970中的数据TLB单元972。指令缓存单元934进一步耦合到存储器单元970中的第2级(L2)缓存单元976。L2缓存单元976耦合到一个或多个其他级别的缓存并且最终耦合到主存储器。

[0171] 作为示例,示例性寄存器重命名、无序发出/执行核心架构可以实现管线900如下:1)指令提取938执行提取和长度解码级902和904;2)解码单元940执行解码级906;3)重命名/分配器单元952执行分配级908和重命名级910;4)(一个或多个)调度器单元956执行调

度级912;5)(一个或多个)物理寄存器文件单元958和存储器单元970执行寄存器读取/存储器读取级914;执行集群960执行执行级916;6)存储器单元970和(一个或多个)物理寄存器文件单元958执行写回/存储器写入级918;7)在异常处理级922中可涉及各种单元;并且8)引退单元954和(一个或多个)物理寄存器文件单元958执行提交级924。

[0172] 核心990可以支持一个或多个指令集(例如,x86指令集(带有已随着较新版本添加的一些扩展);加州森尼维尔市的MIPS技术公司的MIPS指令集;加州森尼维尔市的ARM控股公司的ARM指令集(带有可选的附加扩展,例如NEON)),包括本文描述的(一个或多个)指令。在一个实施例中,核心990包括逻辑来支持压缩数据指令集扩展(例如,AVX1、AVX2),从而允许了被许多多媒体应用使用的操作被利用压缩数据来执行。

[0173] 应当理解,核心可以支持多线程处理(执行操作或线程的两个或更多个并行集合),并且可以按各种方式来支持多线程处理,包括时间切片式多线程处理、同时多线程处理(其中单个物理核心针对该物理核心在同时进行多线程处理的每个线程提供逻辑核心),或者这些的组合(例如,时间切片式提取和解码,然后是同时多线程处理,例如像Intel® Hyperthreading技术中那样)。

[0174] 虽然是在无序执行的上下文中描述寄存器重命名的,但应当理解,寄存器重命名可被用在有序架构中。虽然处理器的图示实施例还包括分开的指令和数据缓存单元934/974和共享的L2缓存单元976,但替代实施例可以针对指令和数据两者具有单个内部缓存,例如,第1级(L1)内部缓存或者多级别的内部缓存。在一些实施例中,系统可以包括内部缓存与在核心和/或处理器外部的外部缓存的组合。替代地,所有缓存可在核心和/或处理器外部。

[0175] 具体示例性有序核心架构

[0176] 图12A-12B图示出更具体的示例性有序核心架构的框图,该核心将是芯片中的若干个逻辑块(包括相同类型和/或不同类型的其他核心)之一。逻辑块通过高带宽互连网络(例如,环状网络)与某些固定功能逻辑、存储器I/O接口和其他必要I/O逻辑进行通信,这取决于应用。

[0177] 图12A是根据本发明的实施例的单个处理器核心及其与片上互连网络1002的连接以及其第2级(L2)缓存本地子集1004的框图。在一个实施例中,指令解码器1000支持具有压缩数据指令集扩展的x86指令集。L1缓存1006允许低时延访问以将存储器缓存到标量和向量单元中。虽然在一个实施例中(为了简化设计),标量单元1008和向量单元1010使用分开的寄存器集合(分别是标量寄存器1012和向量寄存器1014)并且在它们之间传送的数据被写入到存储器,然后被从第1级(L1)缓存1006读回,但本发明的替代实施例可以使用不同的方案(例如,使用单个寄存器集合或者包括允许数据在两个寄存器文件之间传送而不被写入和读回的通信路径)。

[0178] L2缓存的本地子集1004是全局L2缓存的一部分,全局L2缓存被划分成单独的本地子集,每个处理器核心有一个本地子集。每个处理器核心具有到其自身的L2缓存本地子集1004的直接访问路径。处理器核心所读取的数据被存储在其L2缓存子集1004中并且可被快速访问,这与其他处理器核心访问其自身的本地L2缓存子集并行进行。处理器核心所写入的数据被存储在其自身的L2缓存子集1004中并且在必要时被从其他子集冲刷出。环状网络确保了共享数据的一致性。环状网络是双向的,以允许诸如处理器核心、L2缓存和其他逻辑

块之类的代理在芯片内与彼此通信。每个环状数据路径在每个方向上是1012位宽的。

[0179] 图12B是根据本发明的实施例的图12A中的处理器核心的一部分的扩展视图。图12B包括L1缓存1006的L1数据缓存1006A部分，以及关于向量单元1010和向量寄存器1014的更多细节。具体而言，向量单元1010是16宽的向量处理单元(vector processing unit，VPU)(参见16宽ALU1028)，其执行整数、单精度浮点和双精度浮点指令中的一个或多个。VPU支持利用调配单元1020调配寄存器输入，利用数值转换单元1022A-B进行的数值转换，以及利用复制单元1024对存储器输入进行的复制。写入掩码寄存器1026允许断言结果向量(predicating resulting vector)写入。

[0180] 图13是根据本发明的实施例的处理器1100的框图，处理器1100可以具有多于一个核心，可以具有集成的存储器控制器，并且可以具有集成的图形。图13中的实线框图示了具有单个核心1102A、系统代理1110和一组一个或多个总线控制器单元1116的处理器1100，而虚线框的可选添加图示了具有多个核心1102A-N、系统代理单元1110中的一组一个或多个集成存储器控制单元1114以及专用逻辑1108的替代处理器1100。

[0181] 因此，处理器1100的不同实现方式可以包括：1)其中专用逻辑1108是集成图形和/或科学(吞吐量)逻辑(其可以包括一个或多个核心)并且核心1102A-N是一个或多个通用核心(例如，通用有序核心、通用无序核心或者两者的组合)的CPU；2)其中核心1102A-N是大量的主要旨在用于图形和/或科学(吞吐量)的专用核心的协处理器；以及3)其中核心1102A-N是大量的通用有序核心的协处理器。因此，处理器1100可以是通用处理器、协处理器或专用处理器，例如，网络或通信处理器、压缩引擎、图形处理器、GPGPU(通用图形处理单元)、高吞吐量集成众核(many integrated core，MIC)协处理器(包括30个或更多个核心)、嵌入式处理器，等等。处理器可以被实现在一个或多个芯片上。处理器1100可以是一个或多个衬底的一部分和/或可以利用若干个工艺技术中的任何一者被实现在一个或多个衬底上，这些技术例如是BiCMOS、CMOS或NMOS。

[0182] 存储器层次体系可以包括核心1102A-N内的各自的一个或多个级别的缓存1104A-N、一组或者一个或多个共享缓存单元1106以及耦合到该组集成存储器控制器单元1114的外部存储器(未示出)。该组共享缓存单元1106可以包括一个或多个中间级别缓存(例如第2级(L2)、第3级(L3)、第4级(4)或者其他级别的缓存)，最后一级缓存(last level cache，LLC)，和/或这些的组合。虽然在一个实施例中基于环的互连单元1112互连集成图形逻辑1108、该组共享缓存单元1106以及系统代理单元1110/(一个或多个)集成存储器控制器单元1114，但替代实施例也可以使用任何数目的公知技术来互连这种单元。在一个实施例中，在一个或多个缓存单元1106和核心1102A-N之间维持一致性。

[0183] 在一些实施例中，核心1102A-N中的一个或多个能够进行多线程处理。系统代理1110包括协调和操作核心1102A-N的那些组件。系统代理单元1110可包括例如功率控制单元(power control unit，PCU)和显示单元。PCU可以是或者可以包括对核心1102A-N和集成图形逻辑1108的功率状态进行调节所需要的逻辑和组件。显示单元用于驱动一个或多个在外部连接的显示器。

[0184] 核心1102A-N就架构指令集而言可以是同构的或者异构的；也就是说，核心1102A-N中的两个或更多个可能够执行同一指令集，而其他的核心可能够只执行该指令集的子集或者不同的指令集。

[0185] 示例性计算机架构

[0186] 图14-17是示例性计算机架构的框图。本领域中已知的用于膝上型电脑、桌面型电脑、手持PC、个人数字助理、工程工作站、服务器、网络设备、网络集线器、交换机、嵌入式处理器、数字信号处理器(digital signal processor,DSP)、图形设备、视频游戏设备、机顶盒、微控制器、蜂窝电话、便携式媒体播放器、手持设备、以及各种其他电子设备的其他系统设计和配置,也是合适的。一般而言,能够包含本文公开的处理器和/或其他执行逻辑的各种各样的系统或电子设备一般是合适的。

[0187] 现在参考图14,示出了根据本发明的一个实施例的系统1200的框图。系统1200可以包括一个或多个处理器1210、1215,它们耦合到控制器中枢1220。在一个实施例中,控制器中枢1220包括图形存储器控制器中枢(graphics memory controller hub,GMCH)1290和输入/输出中枢(Input/Output Hub,IOH)1250(它们可在分开的芯片上);GMCH 1290包括与存储器1240和协处理器1245耦合的存储器和图形控制器;IOH 1250将输入/输出(I/O)设备1260耦合到GMCH 1290。替代地,存储器和图形控制器中的一者或两者被集成在处理器内(如本文所述),存储器1240和协处理器1245直接耦合到处理器1210,并且控制器中枢1220与IOH 1250在单个芯片中。

[0188] 附加的处理器1215的可选性在图14中用虚线表示。每个处理器1210、1215可包括本文描述的处理核心中的一个或多个并且可以是处理器1100的某个版本。

[0189] 存储器1240可例如是动态随机访问存储器(dynamic random access memory,DRAM)、相变存储器(phase change memory,PCM)、或者两者的组合。对于至少一个实施例,控制器中枢1220经由多点分支总线(例如前端总线(frontside bus,FSB))、点到点接口(例如QuickPath Interconnect(QPI))或者类似的连接1295与(一个或多个)处理器1210、1215进行通信。

[0190] 在一个实施例中,协处理器1245是专用处理器,例如,高吞吐量MIC处理器、网络或通信处理器、压缩引擎、图形处理器、GPGPU、嵌入式处理器,等等。在一个实施例中,控制器中枢1220可包括集成的图形加速器。

[0191] 在物理资源1210、1215之间,就包括架构特性、微架构特性、热特性、功率消耗特性等等在内的价值度量的范围而言,可以有各种差异。

[0192] 在一个实施例中,处理器1210执行控制一般类型的数据处理操作的指令。嵌入在这些指令内的可以是协处理器指令。处理器1210将这些协处理器指令识别为应当由附接的协处理器1245执行的类型。因此,处理器1210在协处理器总线或其他互连上向协处理器1245发出这些协处理器指令(或者表示协处理器指令的控制信号)。(一个或多个)协处理器1245接受并且执行接收到的协处理器指令。

[0193] 现在参考图15,其中示出了根据本发明的实施例的第一更具体示例性系统1300的框图。如图15中所示,多处理器系统1300是点到点互连系统,并且包括经由点到点互连1350耦合的第一处理器1370和第二处理器1380。处理器1370和1380中的每一者可以是处理器1100的某个版本。在本发明的一个实施例中,处理器1370和1380分别是处理器1210和1215,而协处理器1338是协处理器1245。在另一个实施例中,处理器1370和1380分别是处理器1210和协处理器1245。

[0194] 处理器1370和1380被示为分别包括集成存储器控制器(integrated memory

controller,IMC)单元1372和1382。处理器1370还包括点到点(P-P)接口1376和1378作为其总线控制器单元的一部分;类似地,第二处理器1380包括P-P接口1386和1388。处理器1370、1380可以利用P-P接口电路1378、1388经由点到点(P-P)接口1350交换信息。如图15中所示,IMC 1372和1382将处理器耦合到各自的存储器,即存储器1332和存储器1334,存储器1332和存储器1334可以是在本地附接到各自处理器的主存储器的一部分。

[0195] 处理器1370、1380可各自利用点到点接口电路1376、1394、1386、1398经由个体P-P接口1352、1354与芯片组1390交换信息。芯片组1390可以可选地经由高性能接口1339和接口1392与协处理器1338交换信息。在一个实施例中,协处理器1338是专用处理器,例如,高吞吐量MIC处理器、网络或通信处理器、压缩引擎、图形处理器、GPGPU、嵌入式处理器,等等。

[0196] 共享缓存(未示出)可以被包括在任一处理器中,或者在两个处理器之外,但经由P-P互连与处理器连接,从而使得任一个或两个处理器的本地缓存信息在处理器被置于低功率模式中的情况下可以被存储在该共享缓存中。

[0197] 芯片组1390可以经由接口1396耦合到第一总线1316。在一个实施例中,第一总线1316可以是外围组件互连(Peripheral Component Interconnect,PCI)总线,或者诸如快速PCI总线或另一种第三代I/O互连总线之类的总线,虽然本发明的范围不限于此。

[0198] 如图15中所示,各种I/O设备1314可以耦合到第一总线1316以及总线桥1318,总线桥1318将第一总线1316耦合到第二总线1320。在一个实施例中,一个或多个附加的处理器1315(例如,协处理器、高吞吐量MIC处理器、GPGPU、加速器(例如,图形加速器或数字信号处理(DSP)单元)、现场可编程门阵列、或者任何其他处理器,耦合到第一总线1316。在一个实施例中,第二总线1320可以是低引脚数(low pin count,LPC)总线。各种设备可以耦合到第二总线1320,包括例如键盘和/或鼠标1322、通信设备1327和存储单元1328,例如硬盘驱动器或者其他大容量存储设备,其中该存储单元1328在一个实施例中可以包括指令/代码和数据1330。另外,音频I/O 1324可以耦合到第二总线1320。注意,其他架构是可能的。例如,取代图15的点到点架构,系统可以实现多点分支总线或者其他这种架构。

[0199] 现在参考图16,其中示出了根据本发明的实施例的第二更具体示例性系统1400的框图。图15和图16中的相似元素带有相似的标号,并且图15的某些方面被从图16中省略以避免模糊图16的其他方面。

[0200] 图16图示出处理器1370、1380可以分别包括集成存储器和I/O控制逻辑(“CL”)1472和1482。因此,CL 1472、1482包括集成存储器控制器单元并且包括I/O控制逻辑。图16图示出不仅存储器1332、1334耦合到CL 1472、1482,而且I/O设备1414也耦合到控制逻辑1472、1482。常规I/O设备1415耦合到芯片组1390。

[0201] 现在参考图17,其中示出了根据本发明的实施例的SoC 1500的框图。图13中的相似元素带有相似的标号。另外,虚线框是更先进SoC上的可选特征。在图17中,(一个或多个)互连单元1502耦合到:应用处理器1510,其包括一组一个或多个核心1102A-N和(一个或多个)共享缓存单元1106;系统代理单元1110;(一个或多个)总线控制器单元1116;(一个或多个)集成存储器控制器单元1114;一组或一个或多个协处理器1520,其可以包括集成图形逻辑、图像处理器、音频处理器、以及视频处理器;静态随机访问存储器(static random access memory,SRAM)单元1530;直接存储器访问(direct memory access,DMA)单元1532;以及显示单元1540,用于耦合到一个或多个外部显示器。在一个实施例中,(一个或多个)协

处理器1520包括专用处理器，例如网络或通信处理器、压缩引擎、GPGPU、高吞吐量MIC处理器、嵌入式处理器，等等。

[0202] 可以用硬件、软件、固件或者这种实现方案的组合来实现本文公开的机制的实施例。本发明的实施例可以被实现为在包括至少一个处理器、存储系统(包括易失性和非易失性存储器和/或存储元件)、至少一个输入设备、以及至少一个输出设备的可编程系统上执行的计算机程序或程序代码。

[0203] 程序代码，例如图15中所示的代码1330，可以被应用到输入指令以执行本文描述的功能并且生成输出信息。输出信息可以按已知的方式被应用到一个或多个输出设备。对于本申请的目的，处理系统包括任何具有处理器的系统，例如；数字信号处理器(digital signal processor，DSP)、微控制器、专用集成电路(application specific integrated circuit，ASIC)、或者微处理器。

[0204] 可以用高级别程式或面向对象的编程语言来实现程序代码以与处理系统进行通信。如果希望，也可以用汇编或机器语言来实现程序代码。实际上，本文描述的机制在范围上不限于任何特定的编程语言。在任何情况下，该语言可以是经编译或者经解译的语言。

[0205] 至少一个实施例的一个或多个方面可由被存储在机器可读介质上的表示处理器内的各种逻辑的代表性指令来实现，这些代表性指令当被机器读取时使得该机器制作逻辑来执行本文描述的技术。这种被称为“IP核心”的表现形式可以被存储在有形机器可读介质上并且被提供到各种客户或制造设施以加载到实际制作该逻辑或处理器的制作机器中。

[0206] 这种机器可读存储介质可以包括但不限于由机器或设备制造或形成的物品的非暂时性有形布置，包括诸如以下项之类的存储介质：硬盘，任何其他类型的盘(包括软盘、光盘、致密盘只读存储器(compact disk read-only memory，CD-ROM)、可改写致密盘(compact disk rewritable，CD-RW)、以及磁光盘)，半导体设备(诸如，只读存储器(read-only memory，ROM)，诸如动态随机访问存储器(dynamic random access memory，DRAM)、静态随机访问存储器(static random access memory，SRAM)之类的随机访问存储器(random access memory，RAM)，可擦除可编程只读存储器(erasable programmable read-only memory，EPROM)，闪速存储器，电可擦除可编程只读存储器(electrically erasable programmable read-only memory，EEPROM)，相变存储器(phase change memory，PCM))，磁卡或光卡，或者适合用于存储电子指令的任何其他类型的介质。

[0207] 因此，本发明的实施例还包括非暂时性有形机器可读介质，其包含指令或者包含定义本文描述的结构、电路、装置、处理器和/或系统特征的设计数据，例如硬件描述语言(Hardware Description Language，HDL)。这种实施例也可被称为程序产品。

[0208] 仿真(包括二进制转化、代码变形等等)

[0209] 在一些情况下，指令转换器可被用于将指令从源指令集转换到目标指令集。例如，指令转换器可将指令转化(例如，利用静态二进制转化、包括动态编译的动态二进制转化)、变形、仿真或者以其他方式转换到要被核心处理的一个或多个其他指令。可以用软件、硬件、固件或者其组合来实现指令转换器。指令转换器可以在处理器上、在处理器外、或者一部分在处理器上一部分在处理器外。

[0210] 图18是根据本发明的实施例的与使用软件指令转换器来将源指令集中的二进制指令转换成目标指令集中的二进制指令相对比的框图。在图示的实施例中，指令转换器是

软件指令转换器，虽然可替代地，可以用软件、固件、硬件或者其各种组合来实现指令转换器。图18示出了高级别语言1602的程序可被利用x86编译器1604来编译以生成x86二进制代码1606，x86二进制代码1606可以由具有至少一个x86指令集核心的处理器1616原生执行。具有至少一个x86指令集核心的处理器1616表示任何这样的处理器：这种处理器可以通过兼容地执行或以其他方式处理(1)英特尔x86指令集核心的指令集的实质部分或者(2)目标为在具有至少一个x86指令集核心的英特尔处理器上运行的应用或其他软件的目标代码版本，来执行与具有至少一个x86指令集核心的英特尔处理器基本上相同的功能，以便实现与具有至少一个x86指令集核心的英特尔处理器基本上相同的结果。x86编译器1604表示可操作来生成x86二进制代码1606(例如，目标代码)的编译器，x86二进制代码1606在带有或不带有附加的链接处理的情况下可以在具有至少一个x86指令集核心的处理器1616上被执行。类似地，图18示出了高级别语言1602的程序可被利用替代指令集编译器1608来编译以生成替代指令集二进制代码1610，替代指令集二进制代码1610可以由没有至少一个x86指令集核心的处理器1614(例如，具有执行加州森尼维尔市的MIPS技术公司的MIPS指令集和/或执行加州森尼维尔市的ARM控股公司的ARM指令集的核心的处理器)原生执行。指令转换器1612用于将x86二进制代码1606转换成可由没有x86指令集核心的处理器1614原生执行的代码。这个转换后的代码不太可能与替代指令集二进制代码1610相同，因为能够做到这一点的指令转换器是难以制作的；然而，转换后的代码将实现一般操作并且由来自替代指令集的指令构成。因此，指令转换器1612表示通过仿真、模拟或任何其他过程允许不具有x86指令集处理器或核心的处理器或其他电子设备执行x86二进制代码1606的软件、固件、硬件或者其组合。

[0211] 本文描述了用于取回要被保存到崩溃日志的状态信息的技术和架构。在以上描述中，出于说明目的，记载了许多具体细节以便提供对某些实施例的透彻理解。然而，本领域技术人员将会清楚，可以在没有这些具体细节的情况下实践某些实施例。在其他实例中，以框图形式示出了结构和设备以避免模糊描述。

[0212] 本说明书中提及“一个实施例”或“一个实施例”意指结合该实施例描述的特定特征、结构或特性被包括在本发明的至少一个实施例中。在本说明书中各种地方出现短语“在一个实施例中”不一定都指的是同一个实施例。

[0213] 这里的详细描述的一些部分是按计算机存储器内的数据位上的操作的算法和符号表示来呈现的。这些算法描述和表示是被计算领域的技术人员用来最有效地将其工作的实质传达给本领域的其他技术人员的手段。算法在这里并且一般而言被设想为是通向期望结果的步骤的自洽序列。这些步骤是要求对物理量的物理操纵的那些步骤。通常(但并非一定)，这些量采取能够被存储、传送、组合、比较和以其他方式操纵的电信号或磁信号的形式。已证明有时，主要是出于习惯用法的原因，将这些信号称为位、值、元素、符号、字符、项、数字等等，是方便的。

[0214] 然而，应当记住，所有这些和类似的术语都将与适当的物理量相关联，并且只是应用到这些量的方便标签。除非从这里的论述清楚看出另有具体声明，否则要明白在整个说明书各处，利用诸如“处理”或“计算”或“运算”或“确定”或“显示”之类的术语的论述指的是计算机系统或类似的电子计算设备的动作和过程，这些动作和过程将计算机系统的寄存器和存储器内的被表示为物理(电子)量的数据操纵和变换成计算机系统存储器或寄存器或

其他这种信息存储、传输或显示设备内的被类似地表示为物理量的其他数据。

[0215] 某些实施例还涉及用于执行这里的操作的装置。此装置可以是为要求的目的而专门构造的，或者其可以包括由存储在计算机中的计算机程序选择性地激活或重配置的通用计算机。这种计算机程序可被存储在计算机可读存储介质中，例如但不限于任何类型的盘(包括软盘、光盘、CD-ROM和磁光盘)、只读存储器(read-only memory，ROM)、随机访问存储器(random access memory，RAM)(例如动态RAM(dynamic RAM，DRAM))、EPROM、EEPROM、磁卡或光卡、或者适合用于存储电子指令并且耦合到计算机系统总线的任何类型的介质。

[0216] 本文给出的算法和显示并不内在地与任何特定的计算机或其他装置相关。各种通用系统可与根据本文的教导的程序一起使用，或者可证明，构造更专门的装置来执行所要求的方法步骤，是方便的。各种这些系统的必需结构将从本文的描述中显现。此外，某些实施例不是参考任何特定的编程语言来描述的。将会明白，可以使用各种编程语言来实现如本文所述的这种实施例的教导。

[0217] 除了本文描述的以外，还可对所公开的实施例及其实现方式做出各种修改，而不脱离其范围。因此，应当从说明意义而不是限制意义上来解释本文的图示和示例。应当仅通过参考所附权利要求来衡量本发明的范围。

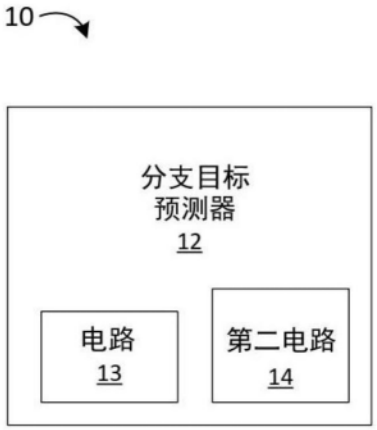

10
分支目标
预测器
12
电路
13
第二电路
14

图1

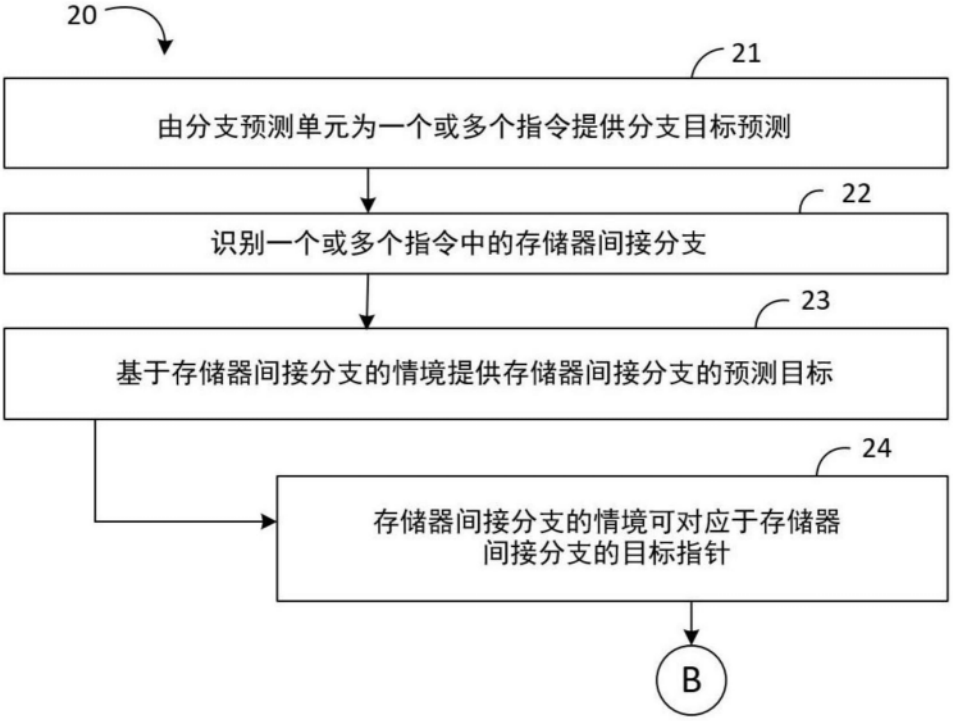

20
21
由分支预测单元为一个或多个指令提供分支目标预测
22
识别一个或多个指令中的存储器间接分支
23
基于存储器间接分支的情境提供存储器间接分支的预测目标
24
存储器间接分支的情境可对应于存储器
间接分支的目标指针
B

图2A

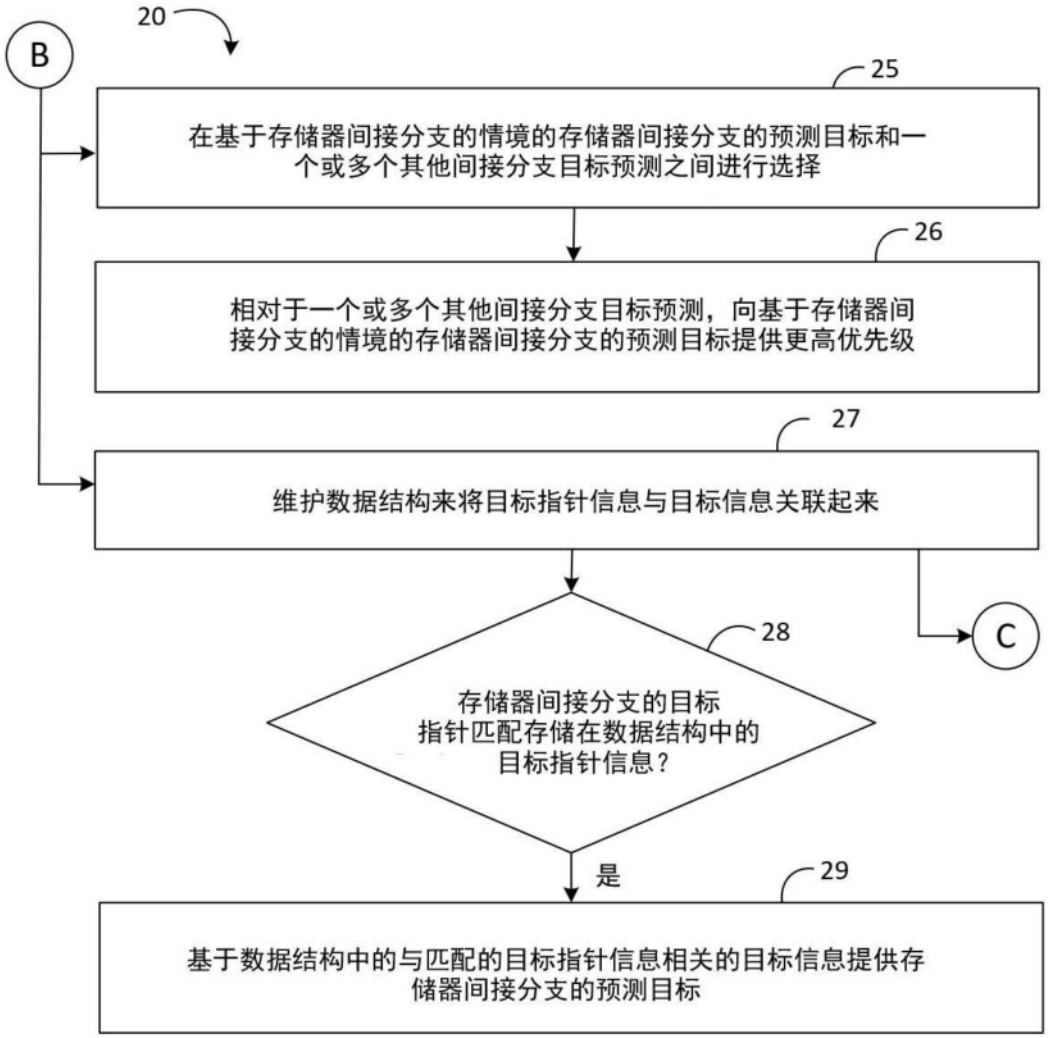

20
B
25
在基于存储器间接分支的情境的存储器间接分支的预测目标和一个或多个其他间接分支目标预测之间进行选择
26
相对于一个或多个其他间接分支目标预测，向基于存储器间接分支的情境的存储器间接分支的预测目标提供更高优先级
27
维护数据结构来将目标指针信息与目标信息关联起来
C
28
存储器间接分支的目标指针匹配存储在数据结构中的目标指针信息？
是
29
基于数据结构中的与匹配的目标指针信息相关的目标信息提供存储器间接分支的预测目标

图2B

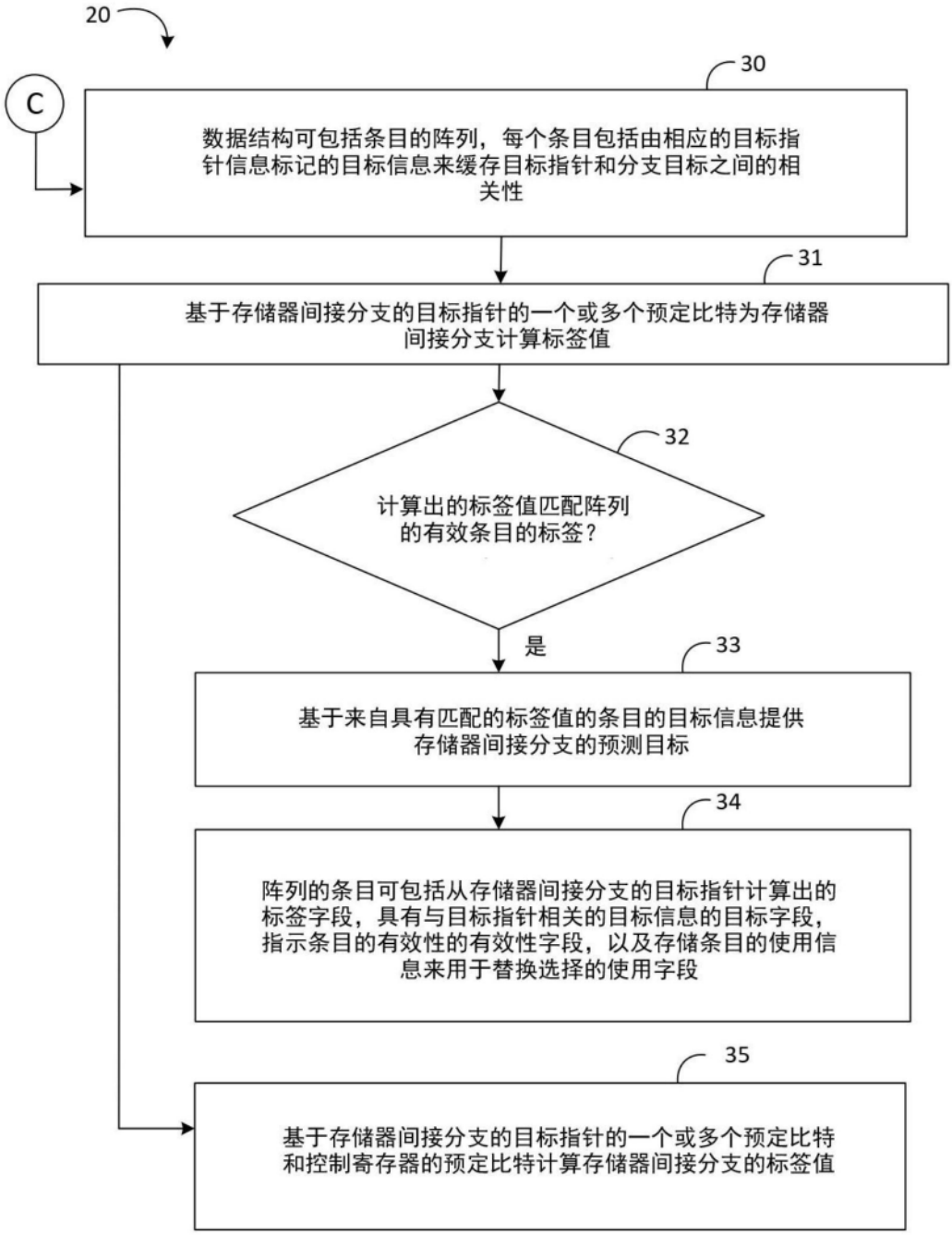

20
C
30
数据结构可包括条目的阵列，每个条目包括由相应的目标指针信息标记的目标信息来缓存目标指针和分支目标之间的相关性
31
基于存储器间接分支的目标指针的一个或多个预定比特为存储器间接分支计算标签值
32
计算出的标签值匹配阵列的有效条目的标签？
是
33
基于来自具有匹配的标签值的条目的目标信息提供存储器间接分支的预测目标
34
阵列的条目可包括从存储器间接分支的目标指针计算出的标签字段，具有与目标指针相关的目标信息的目标字段，指示条目的有效性的有效性字段，以及存储条目的使用信息来用于替换选择的使用字段
35
基于存储器间接分支的目标指针的一个或多个预定比特和控制寄存器的预定比特计算存储器间接分支的标签值

图2C

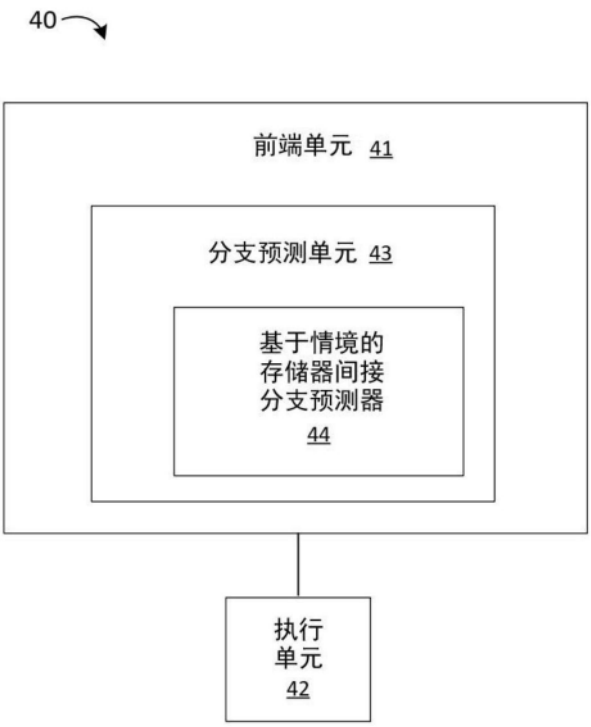

40
前端单元 41
分支预测单元 43
基于情境的
存储器间接
分支预测器
44
执行
单元
42

图3

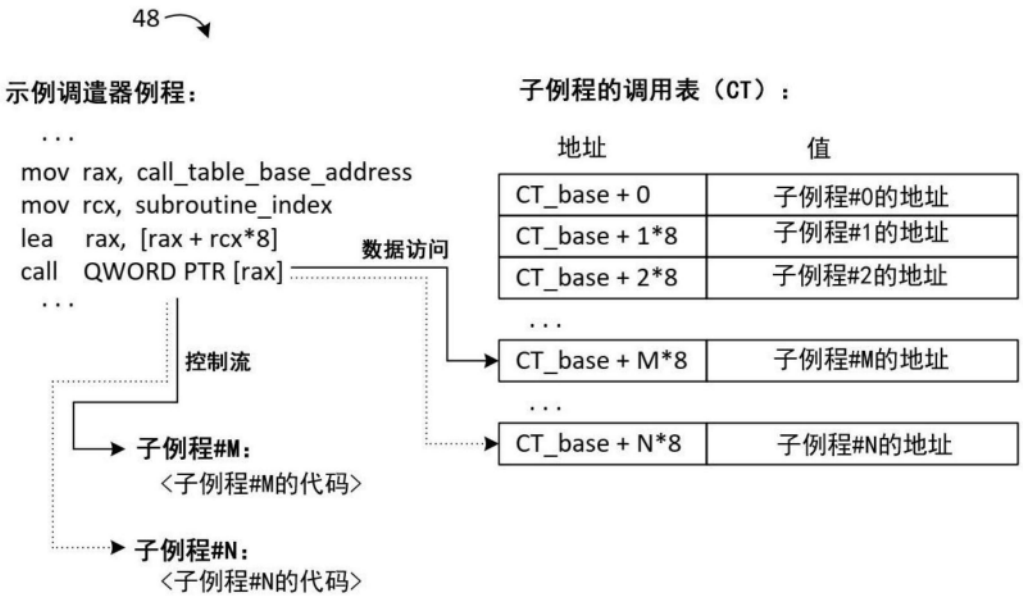

48
示例调遣器例程：
...
mov rax, call_table_base_address
mov rcx, subroutine_index
lea rax, [rax + rcx*8]
call QWORD PTR [rax]
...
数据访问
控制流
子例程#M：
<子例程#M的代码>
子例程#N：
<子例程#N的代码>
子例程的调用表（CT）：
地址
值
CT_base + 0
子例程#0的地址
CT_base + 1*8
子例程#1的地址
CT_base + 2*8
子例程#2的地址
...
CT_base + M*8
子例程#M的地址
...
CT_base + N*8
子例程#N的地址

图4

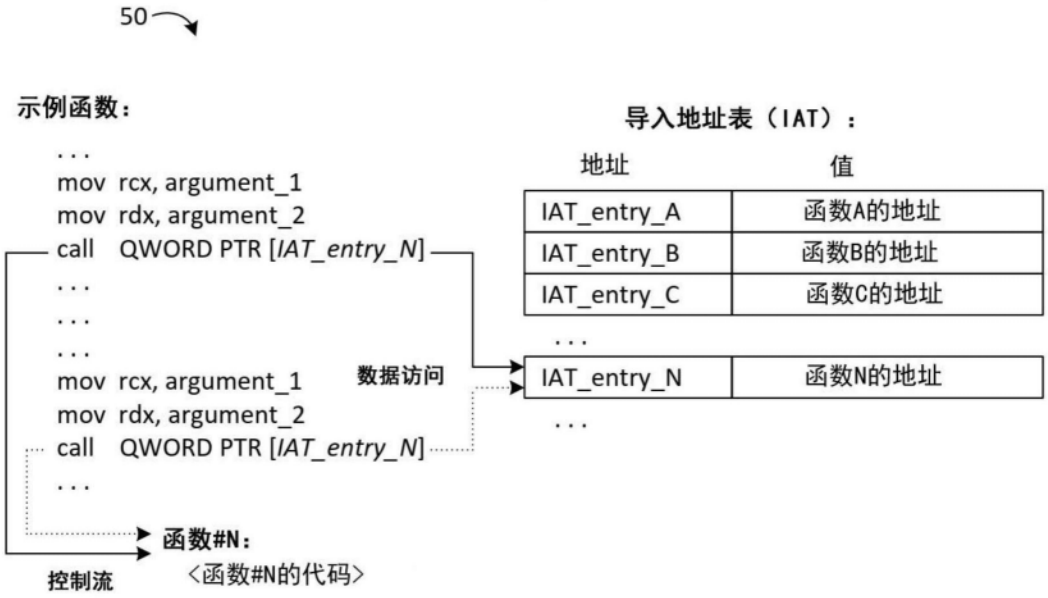

50
示例函数：
. . .
mov rcx, argument_1
mov rdx, argument_2
call QWORD PTR [IAT_entry_N]
. . .
. . .
. . .
mov rcx, argument_1
mov rdx, argument_2
call QWORD PTR [IAT_entry_N]
. . .
数据访问
函数#N：
<函数#N的代码>
控制流
导入地址表（IAT）：
地址
值
IAT_entry_A
函数A的地址
IAT_entry_B
函数B的地址
IAT_entry_C
函数C的地址
. . .
IAT_entry_N
函数N的地址
. . .

图5

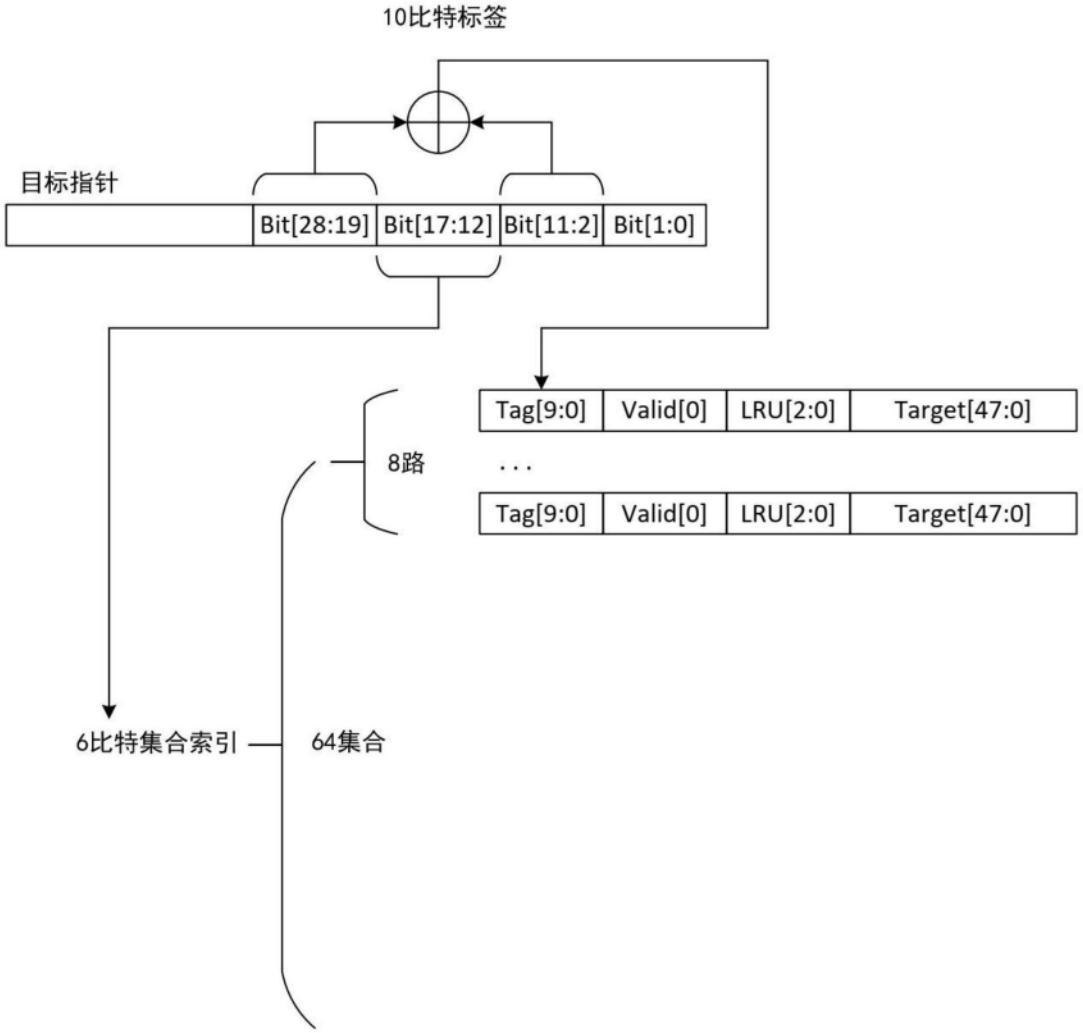

10比特标签
目标指针
Bit[28:19]
Bit[17:12]
Bit[11:2]
Bit[1:0]
Tag[9:0]
Valid[0]
LRU[2:0]
Target[47:0]
8路
...
Tag[9:0]
Valid[0]
LRU[2:0]
Target[47:0]
6比特集合索引
64集合

图6

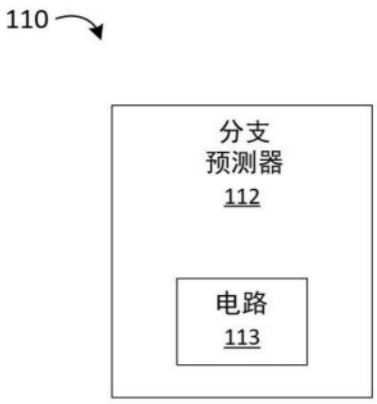

110
分支
预测器
112
电路
113

图7

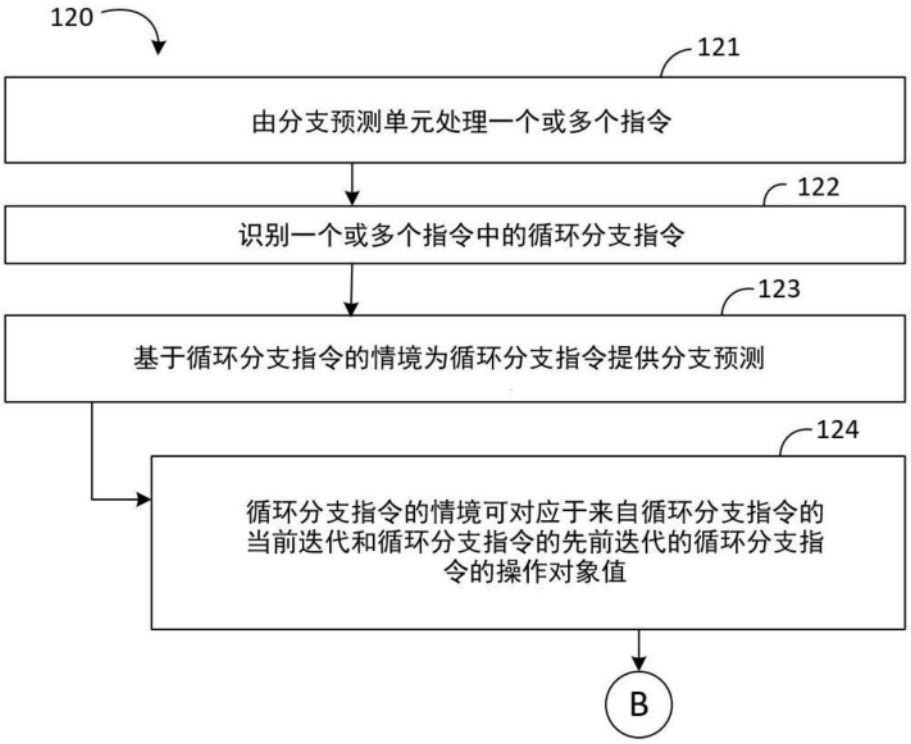

120
121
由分支预测单元处理一个或多个指令
122
识别一个或多个指令中的循环分支指令
123
基于循环分支指令的情境为循环分支指令提供分支预测
124
循环分支指令的情境可对应于来自循环分支指令的当前迭代和循环分支指令的先前迭代的循环分支指令的操作对象值
B

图8A

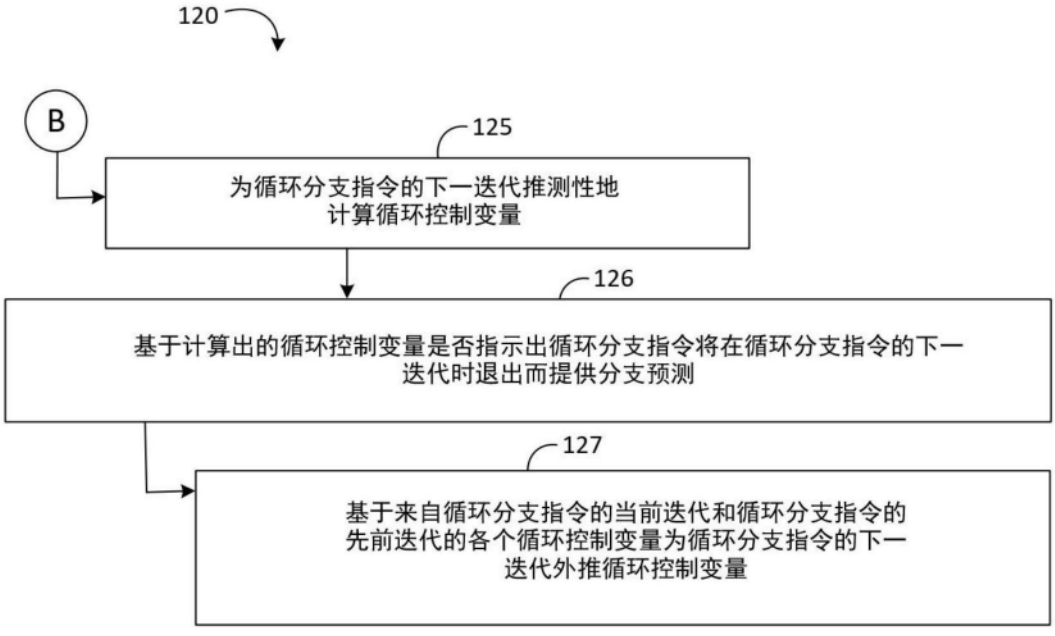

120
B
125
为循环分支指令的下一迭代推测性地计算循环控制变量
126
基于计算出的循环控制变量是否指示出循环分支指令将在循环分支指令的下一迭代时退出而提供分支预测
127
基于来自循环分支指令的当前迭代和循环分支指令的先前迭代的各个循环控制变量为循环分支指令的下一迭代外推循环控制变量

图8B

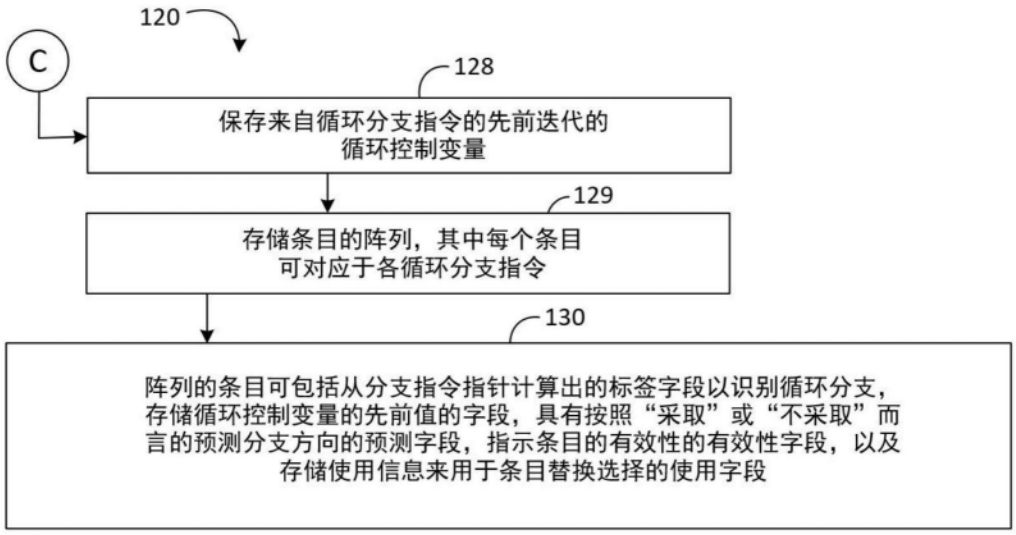

120
C
128
保存来自循环分支指令的先前迭代的
循环控制变量
129
存储条目的阵列，其中每个条目
可对应于各循环分支指令
130
阵列的条目可包括从分支指令指针计算出的标签字段以识别循环分支，
存储循环控制变量的先前值的字段，具有按照“采取”或“不采取”而
言的预测分支方向的预测字段，指示条目的有效性的有效性字段，以及
存储使用信息来用于条目替换选择的使用字段

图8C

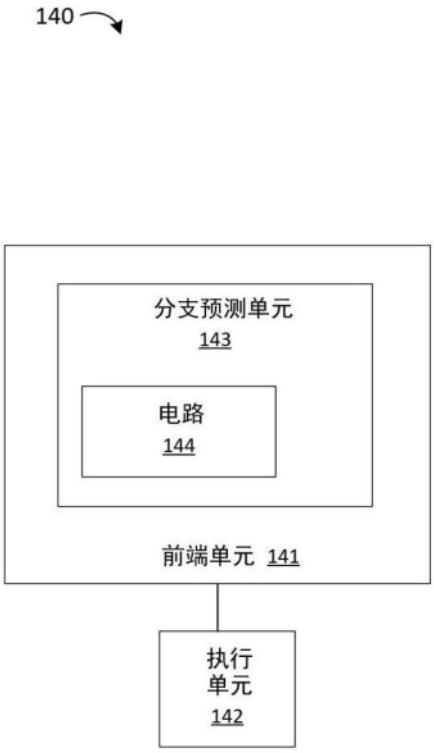

140
分支预测单元
143
电路
144
前端单元 141
执行
单元
142

图9

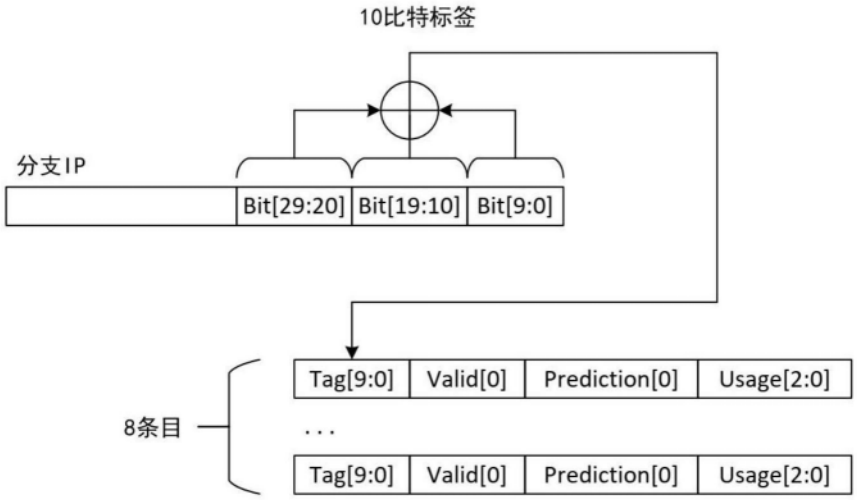

10比特标签
分支IP
Bit[29:20]
Bit[19:10]
Bit[9:0]
8条目
Tag[9:0]
Valid[0]
Prediction[0]
Usage[2:0]
...
Tag[9:0]
Valid[0]
Prediction[0]
Usage[2:0]

图10

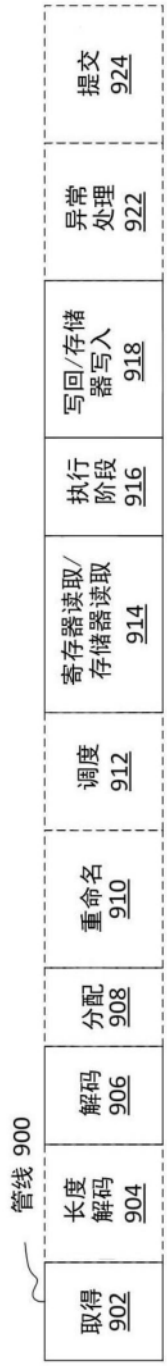

管线 900
取得 902
长度解码 904
解码 906
分配 908
重命名 910
调度 912
寄存器读取/存储器读取 914
执行阶段 916
写回/存储器写入 918
异常处理 922
提交 924

图11A

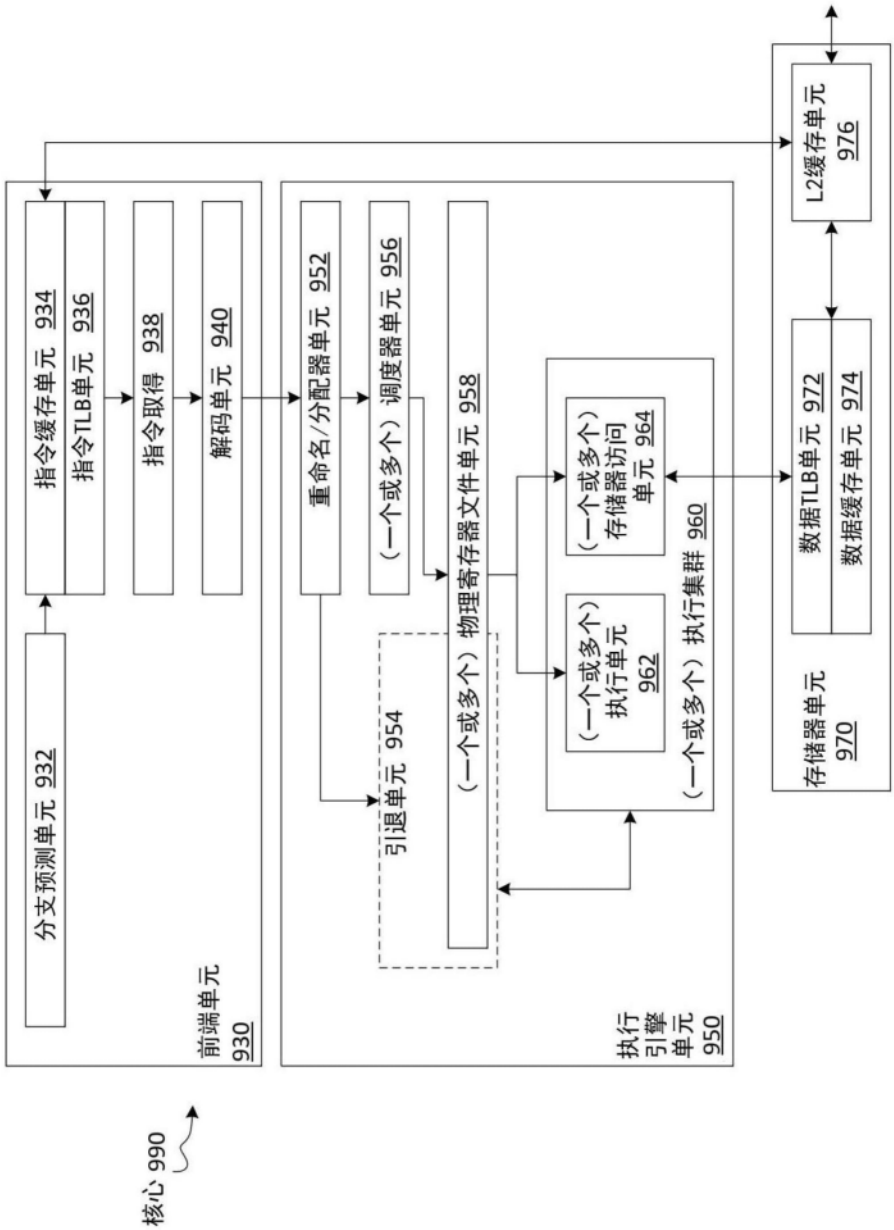

核心 990
前端单元 930
分支预测单元 932
指令缓存单元 934
指令TLB单元 936
指令取得 938
解码单元 940
执行引擎单元 950
重命名/分配器单元 952
引退单元 954
(一个或多个) 调度器单元 956
(一个或多个) 物理寄存器文件单元 958
(一个或多个) 执行集群 960
(一个或多个) 执行单元 962
(一个或多个) 存储器访问单元 964
存储器单元 970
数据TLB单元 972
数据缓存单元 974
L2缓存单元 976

图11B

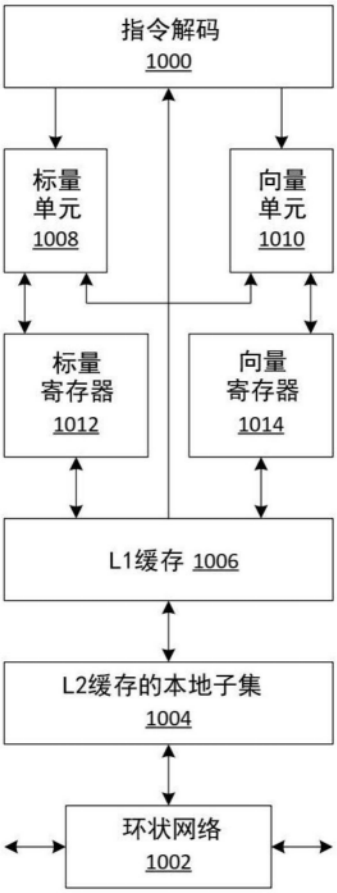

指令解码
1000
标量
单元
1008
向量
单元
1010
标量
寄存器
1012
向量
寄存器
1014
L1缓存 1006
L2缓存的本地子集
1004
环状网络
1002

图12A

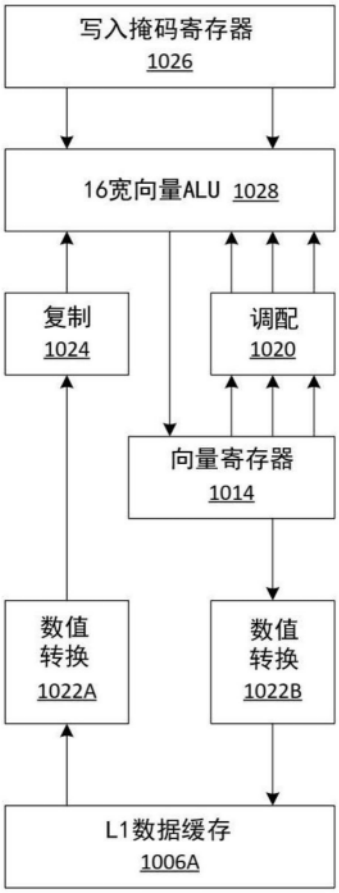

写入掩码寄存器
1026
16宽向量ALU 1028
复制
1024
调配
1020
向量寄存器
1014
数值
转换
1022A
数值
转换
1022B
L1数据缓存
1006A

图12B

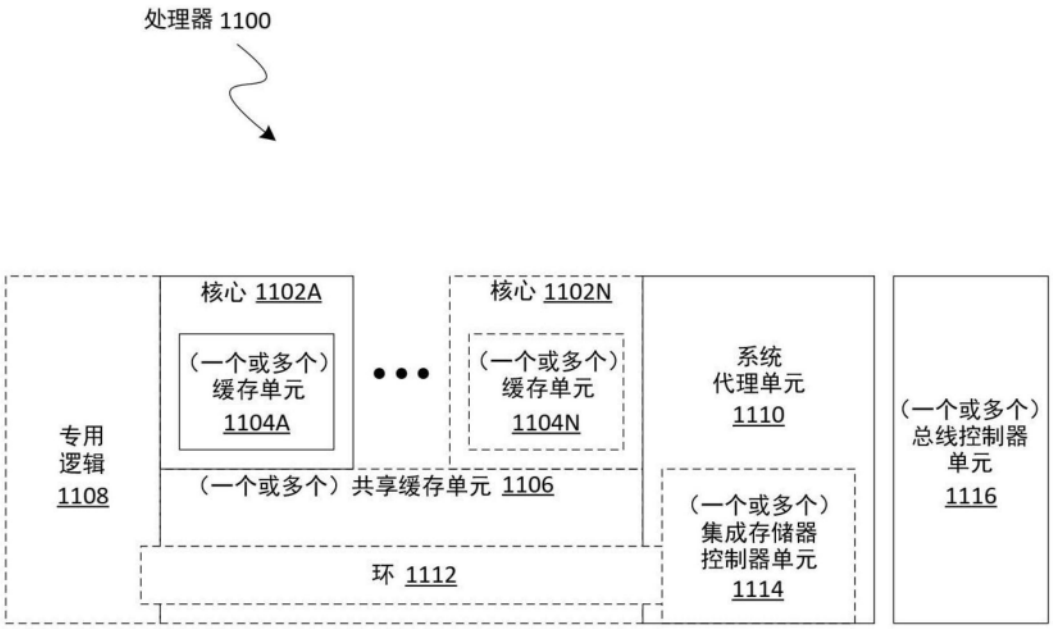

处理器 1100
核心 1102A
(一个或多个)
缓存单元
1104A
核心 1102N
(一个或多个)
缓存单元
1104N
专用
逻辑
1108
(一个或多个) 共享缓存单元 1106
环 1112
系统
代理单元
1110
(一个或多个)
集成存储器
控制器单元
1114
(一个或多个)
总线控制器
单元
1116

图13

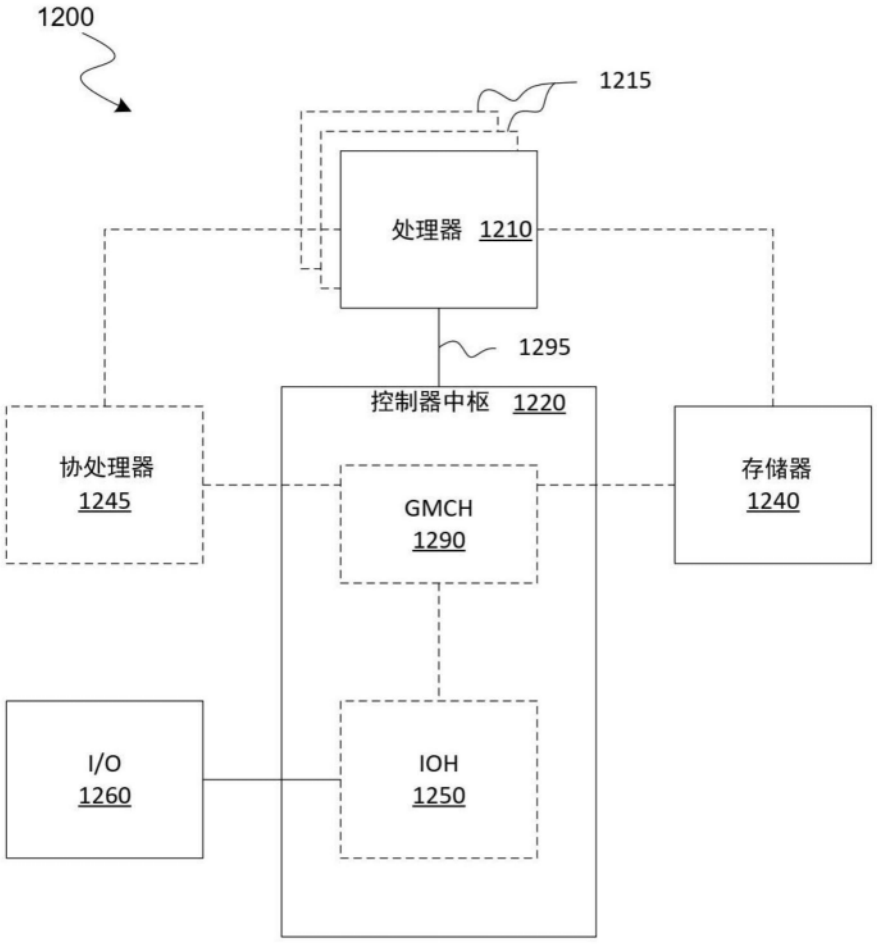

1200
1215
处理器 1210
1295
控制器中枢 1220
协处理器
1245
GMCH
1290
存储器
1240
I/O
1260
IOH
1250

图14

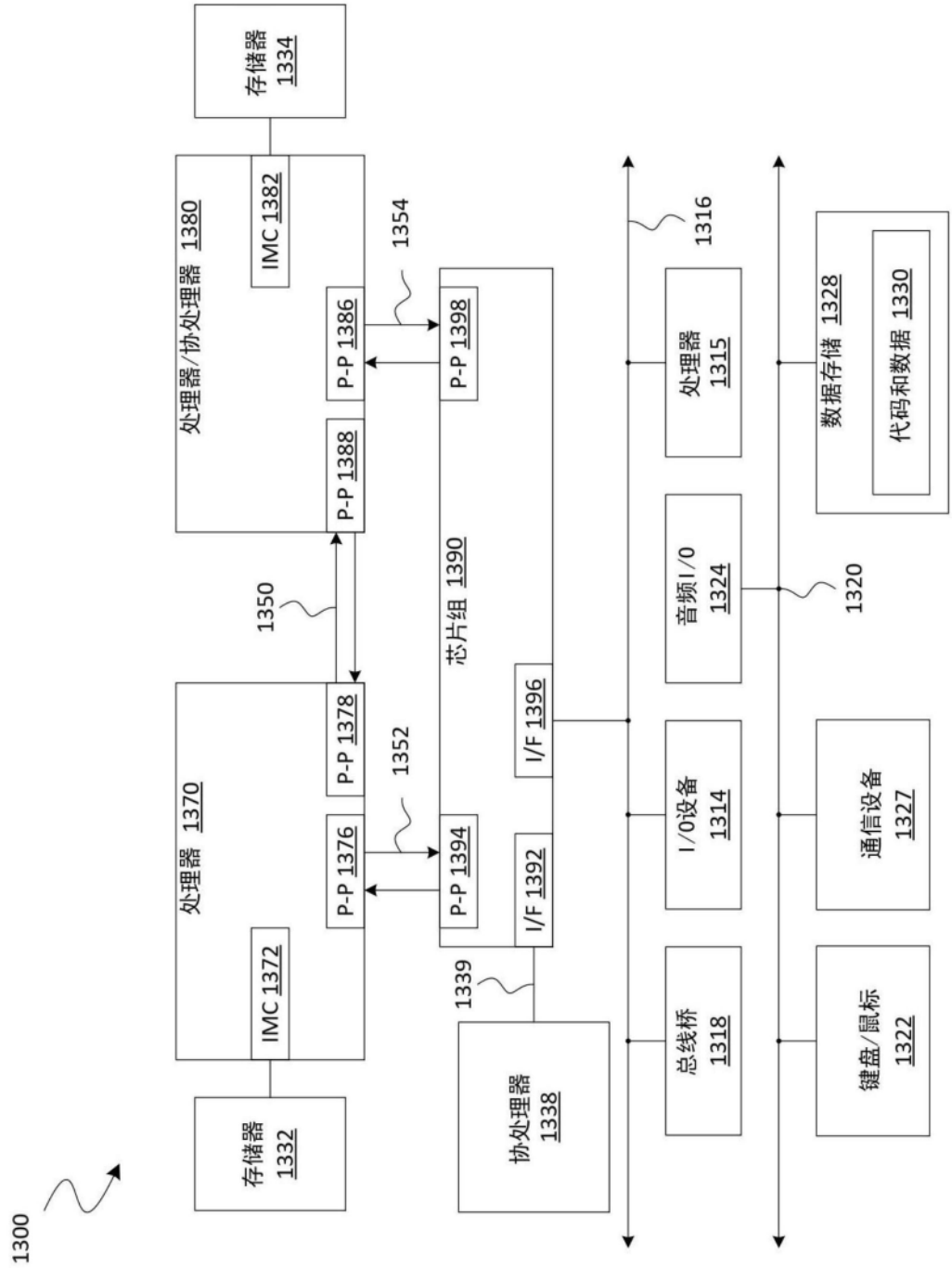

1300
存储器 1332
处理器 1370
IMC 1372
P-P 1376
P-P 1378
1350
1352
处理器/协处理器 1380
IMC 1382
P-P 1386
P-P 1388
1354
存储器 1334
芯片组 1390
P-P 1394
P-P 1398
I/F 1392
I/F 1396
1339
协处理器 1338
总线桥 1318
I/O设备 1314
音频I/O 1324
处理器 1315
1316
1320
键盘/鼠标 1322
通信设备 1327
数据存储 1328
代码和数据 1330

图15

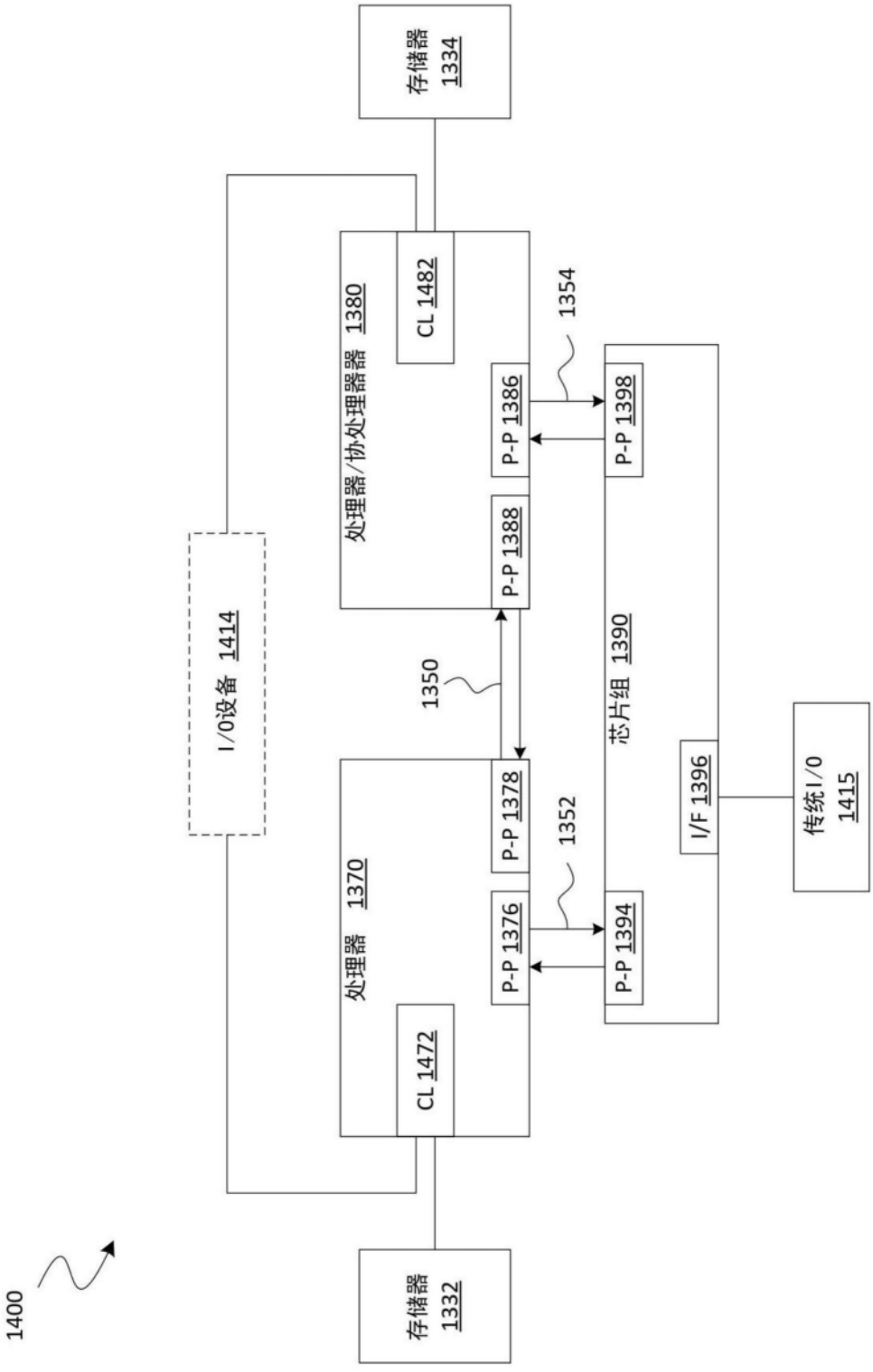

存储器 1334
处理器/协处理器器 1380
CL 1482
P-P 1386
P-P 1388
1354
P-P 1398
芯片组 1390
I/O设备 1414
1350
处理器 1370
P-P 1378
P-P 1376
1352
P-P 1394
I/F 1396
传统I/O 1415
CL 1472
1400
存储器 1332

图16

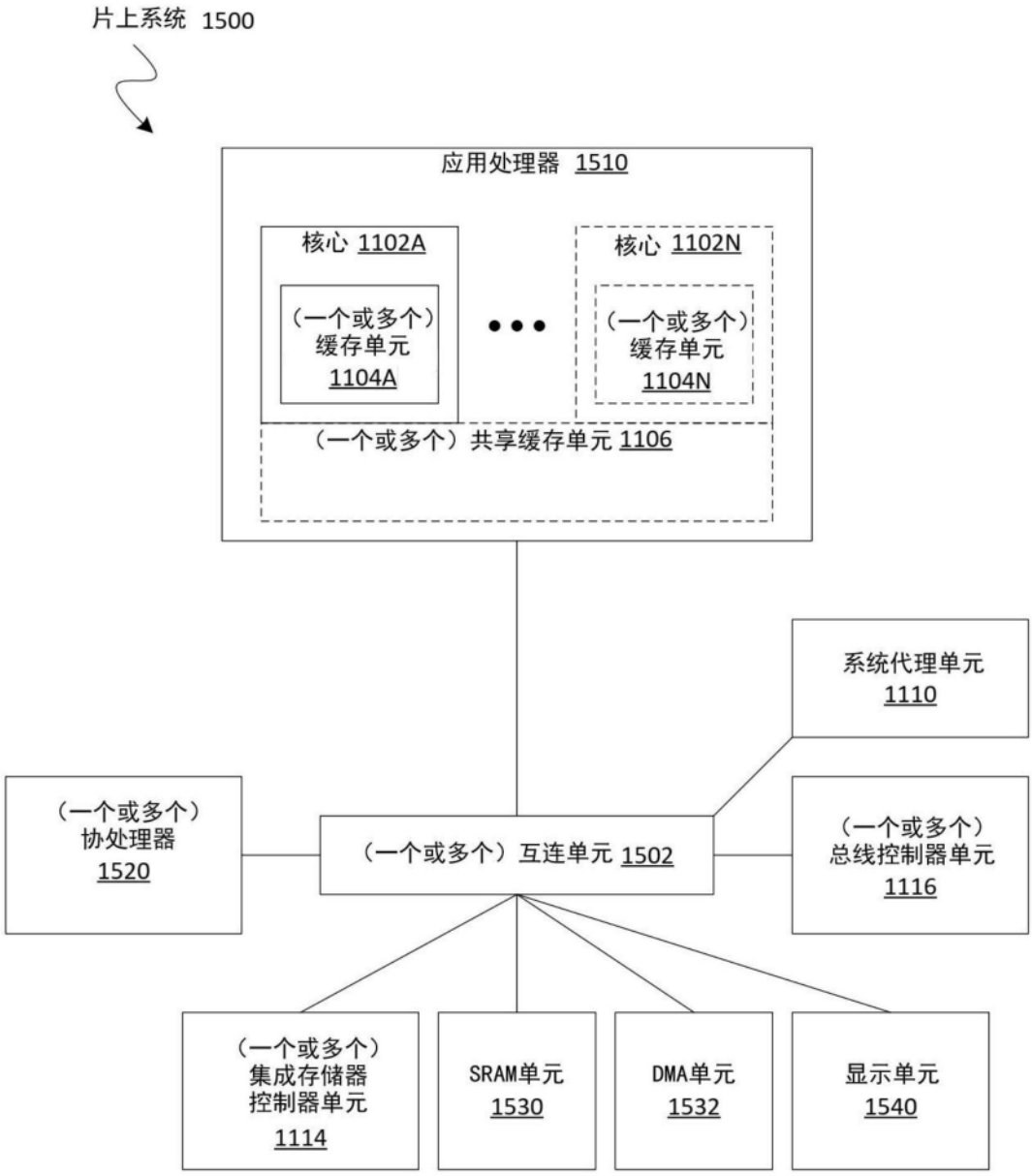

片上系统 1500
应用处理器 1510
核心 1102A
(一个或多个)
缓存单元
1104A
核心 1102N
(一个或多个)
缓存单元
1104N
(一个或多个)共享缓存单元 1106
系统代理单元
1110
(一个或多个)
协处理器
1520
(一个或多个)互连单元 1502
(一个或多个)
总线控制器单元
1116
(一个或多个)
集成存储器
控制器单元
1114
SRAM单元
1530
DMA单元
1532
显示单元
1540

图17

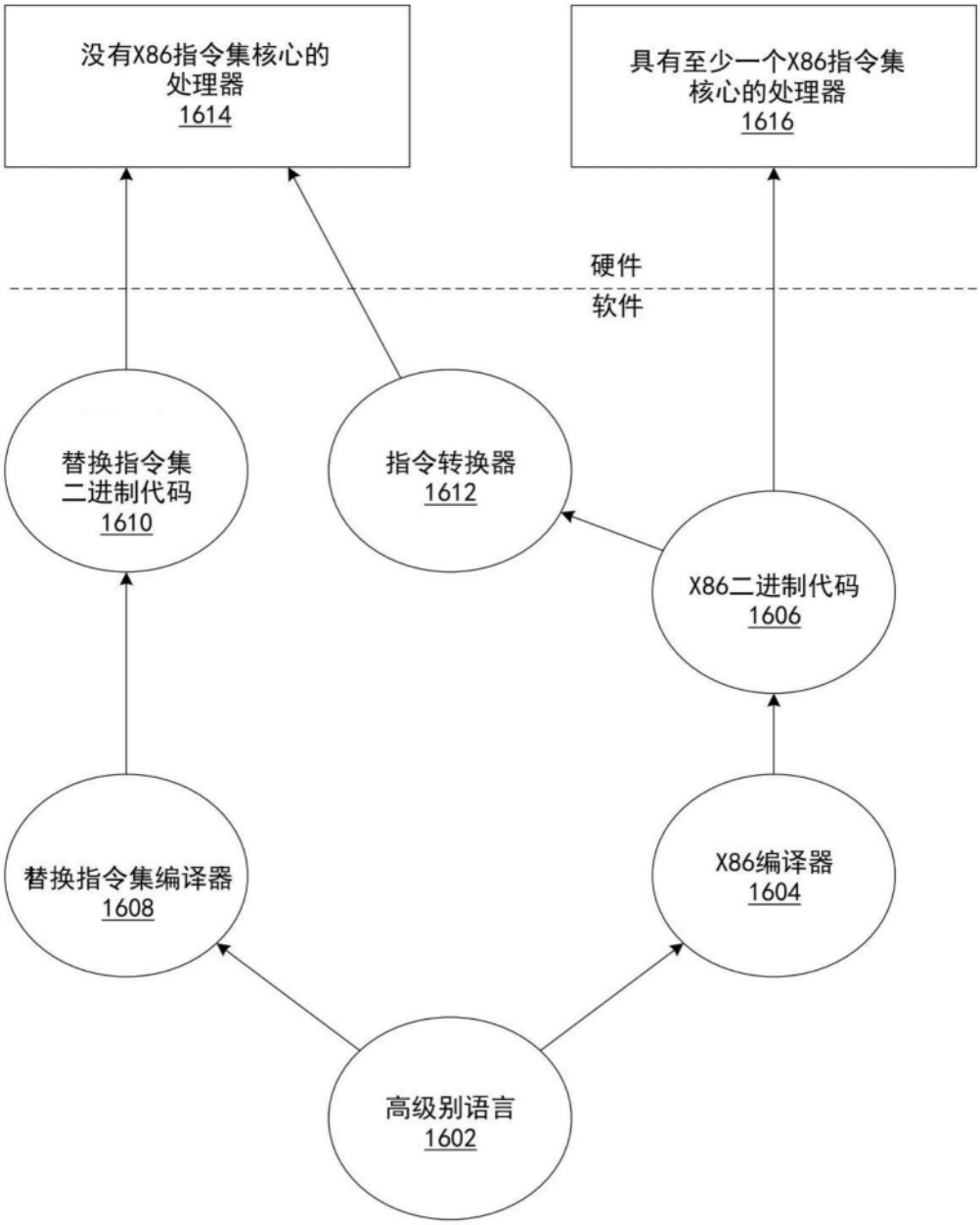

没有X86指令集核心的
处理器
1614
具有至少一个X86指令集
核心的处理器
1616
硬件
软件
替换指令集
二进制代码
1610
指令转换器
1612
X86二进制代码
1606
替换指令集编译器
1608
X86编译器
1604
高级别语言
1602

图18