(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 110249302 A
(43)申请公布日 2019.09.17

(21)申请号 201880010320.1

(22)申请日 2018.01.30

(30)优先权数据
15/425,632 2017.02.06 US

(85)PCT国际申请进入国家阶段日
2019.08.05

(86)PCT国际申请的申请数据
PCT/US2018/015816 2018.01.30

(87)PCT国际申请的公布数据
WO2018/144408 EN 2018.08.09

(71)申请人 微软技术许可有限责任公司
地址 美国华盛顿州

(72)发明人 G·古普塔 D·C·伯格

(74)专利代理机构 北京市金杜律师事务所 11256
代理人 王茂华 彭梦晔

(51)Int.Cl.
G06F 9/30(2006.01)
G06F 9/38(2006.01)

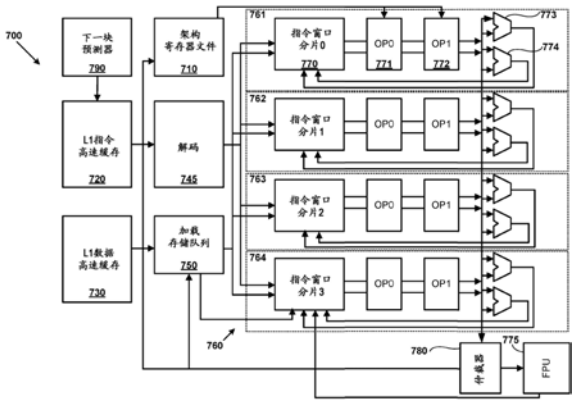
权利要求书2页 说明书28页 附图15页

(54)发明名称

在处理器核上同时执行多个程序

(57)摘要

公开了用于将资源分配给基于块的处理器架构中的上下文的系统和方法。在所公开的技术的一个示例中,处理器被配置为在由处理器执行的多个上下文之间在空间上分配资源,包括高速缓存、功能单元和寄存器文件。在所公开的技术的第二示例中,处理器被配置为例如在时钟周期的基础上在多个上下文之间在时间上分配资源,包括高速缓存、寄存器文件和分支预测器。保证每个上下文都能访问其分配的资源,以避免竞争处理器资源的上下文出现饥饿。结果缓冲器可以用于将较大的指令块折叠成可以映射到较小大小的指令窗口的部分。结果缓冲器存储可以传递到指令块的后续部分的操作数结果。



CN 110249302 A

1.一种处理器，可配置为根据两种或更多种模式在空间上分配一组执行资源，所述模式包括：

第一模式，在所述第一模式下，所述处理器被配置为将所述一组执行资源的第一部分在空间上分配给第一上下文并且将所述一组执行资源的不同的第二部分在空间上分配给第二上下文，相应的所述第一部分和所述第二部分在所述上下文处于活动状态的同时不在所述第一上下文与所述第二上下文之间被共享。

2.根据权利要求1所述的处理器，其中所述模式还包括：

第二模式，在所述第二模式下，所述处理器被配置为将所述一组执行资源全部在空间上重新分配给单个上下文。

3.根据权利要求1或2所述的处理器，其中所述模式还包括：

第三模式，在所述第三模式下，所述处理器被配置为将所述一组执行资源的一部分但不是全部在空间上重新分配给单个上下文，并且不分配所述一组执行资源的剩余部分。

4.根据权利要求1至3中任一项所述的处理器，其中所分配的所述一组执行资源至少包括以下中的一项或多项：指令窗口、功能单元、数据高速缓存、指令高速缓存、结果缓冲器或物理寄存器文件。

5.根据权利要求1至4中任一项所述的处理器，其中所述一组执行资源被首先设置，并且其中所述处理器的第二组执行资源在时间上被分配给所述第一上下文和所述第二上下文。

6.根据权利要求5所述的处理器，其中在时间上被分配的所述第二组执行资源至少包括以下中的一项或多项：分支预测器、指令高速缓存或数据高速缓存。

7.根据权利要求1至6中任一项所述的处理器，其中所述一组执行资源的所述第一部分包括结果缓冲器，所述结果缓冲器被配置为从执行资源的所述第一部分的第一分片向执行资源的所述第一部分的第二分片传送用于原子指令块的操作数。

8.一种处理器，包括：

一组执行资源，被配置为执行针对两个或更多个执行上下文的处理器指令；

控制单元，被配置为：

在所述两个或更多个执行上下文之间在空间上分配所述一组资源的第一部分，以及

在所述两个或更多个执行上下文之间在时间上分配所述一组执行资源的不同的第二部分；以及

其中所述第一部分和所述第二部分在执行所述两个或更多个执行上下文中的至少一个执行上下文的执行之前被确定。

9.根据权利要求8所述的处理器，其中所述一组执行资源在时钟周期的基础上被分配给所述执行上下文中的相应执行上下文。

10.根据权利要求8或9所述的处理器，其中：

所述处理器是基于块的处理器；以及

所述一组执行资源包括分支预测器，所述分支预测器基于以下输入中的一个或多个输入生成下一指令块地址的预测：

目标块地址，

是否采取了分支，

源块地址,或者

针对执行上下文的标识符。

11.根据权利要求8至10中任一项所述的处理器,其中:

所述处理器是基于块的处理器;以及

所述执行上下文中的每个执行上下文包括架构状态数据,所述架构状态数据针对包括两个或更多个处理器指令的原子块的线程或进程。

12.一种操作基于块的处理器的方法,所述方法包括:

根据预定分配将所述基于块的处理器的执行资源分配给所述处理器的一个或多个执行上下文;以及

利用所分配的所述执行资源执行针对所述执行上下文的指令块。

13.根据权利要求12所述的方法,还包括:

重新分配所述执行资源,其中所述执行资源直到所述上下文完成执行才被重新分配。

14.根据权利要求12或13所述的方法,还包括:

所述预定分配将所述执行资源的第一部分在空间上指派给所述执行上下文中的每个执行上下文;以及

所述预定分配将所述执行资源的不相交的第二部分在时间上分指派所述执行上下文中的每个执行上下文。

15.根据权利要求12至14中任一项所述的方法,还包括对于所述执行上下文中的第一执行上下文:

将由所述第一执行上下文的执行资源的第一部分生成的数据临时存储在结果缓冲器中;以及

将临时存储的所述数据从所述结果缓冲器发送到结果缓冲器中的所述第一执行上下文的执行资源的第二部分。

在处理器核上同时执行多个程序

背景技术

[0001] 由于通过摩尔定律预测的连续晶体管缩放，微处理器受益于晶体管数目、集成电路成本、制造资本、时钟频率和能量效率的持续提高，其中相关处理器指令集架构(ISA)几乎没有变化。然而，在过去40年中推动半导体行业发展的光刻缩放所带来的好处正在减缓或甚至逆转。多年来，精简指令集计算(RISC)架构一直是处理器设计的主要范例。无序的超标量实现没有表现出面积或性能的持续改进。因此，有足够的机会改进处理器ISA以扩展性能改进。

发明内容

[0002] 公开了用于处理器的方法、装置和计算机可读存储设备，包括具有基于块的处理器指令集架构(BB-ISA)的那些处理器，以使用可变数目的分配资源来执行指令。例如，显式数据图执行(EDGE)ISA处理器可以将其所有可指派资源分配给单个执行上下文，或者在两个或更多个上下文之间划分所指派的资源，从而实现更高的吞吐量和/或资源利用率。

[0003] 所描述的技术和工具可以潜在地改进处理器性能，并且可以单独实现，或者以彼此的各种组合实现。如下面将更全面地描述的，所描述的技术和工具可以在数字信号处理器、微处理器、专用集成电路(ASIC)、软处理器(例如，使用可重新配置逻辑在现场可编程门阵列(FPGA)中实现的微处理器核)、可编程逻辑或其他合适的逻辑电路中实现。对于本领域普通技术人员来说很清楚的是，所公开的技术可以在各种计算平台中实现，包括但不限于服务器、大型机、蜂窝电话、智能电话、PDA、手持设备、手持计算机、触摸屏平板设备、平板计算机、可穿戴计算机和笔记本计算机。

[0004] 在所公开的技术的一些示例中，处理器包括基于块的处理器核，该处理器核可以在上下文(例如，进程或线程)之间在空间上和/或在时间上分配其资源。可以折叠指令块部分以使用结果缓冲器映射到比指令块小的指令窗口。

[0005] 提供本“发明内容”是为了以简化的形式介绍一些概念，这些概念将在下面的“具体实施方式”中进一步描述。本“发明内容”不旨在标识所要求保护的主题的关键特征或必要特征，也不旨在用于限制所要求保护的主题的范围。通过参考附图进行的以下详细描述，所公开的主题的前述和其他目的、特征和优点将变得更加明显。

附图说明

[0006] 图1示出了可以在所公开的技术的一些示例中使用的包括多个处理器核的基于块的处理器。

[0007] 图2示出了可以在所公开的技术的一些示例中使用的基于块的处理器核。

[0008] 图3示出了根据所公开的技术的某些示例的多个指令块。

[0009] 图4示出了源代码和相应指令块的部分。

[0010] 图5示出了可以在所公开的技术的一些示例中使用的基于块的处理器头和指令。

[0011] 图6是示出基于块的处理器的处理器核的状态的进展的示例的流程图。

[0012] 图7是示出可以在所公开的技术的某些示例中使用的可配置为以多种操作模式操作的示例处理器的框图。

[0013] 图8A-8C是示出可以在所公开的技术的某些示例中使用的被配置为以多种操作模式操作的示例处理器的框图。

[0014] 图9是示出可以在所公开的技术的某些示例中使用的可配置为使用结果缓冲器的示例处理器的框图。

[0015] 图10是示出可以在所公开的技术的某些示例中执行的处理器资源的空间和时间分配的示例的图。

[0016] 图11是概述可以在所公开的技术的某些示例中执行的分配资源和执行指令块的示例的流程图。

[0017] 图12是概述使用空间和/或时间分配的资源来执行指令块的示例的流程图，包括使用结果缓冲器。

[0018] 图13是示出用于实现所公开的技术的一些实施例的合适的计算环境的框图。

具体实施方式

[0019] I.一般考虑因素

[0020] 在代表性实施例的上下文中阐述本发明，这些代表性实施例无意以任何方式进行限制。

[0021] 除非上下文另有明确规定，否则如本申请中使用的，单数形式“一个(a)”、“一个(an)”和“该(the)”包括复数形式。另外，术语“包括(includes)”表示“包括(comprises)”。此外，术语“耦合(coupled)”包括将项目耦合或链接在一起的机械、电气、磁性、光学以及其他实际方式，并且不排除在耦合的项目之间的中间元件的存在。此外，如本文中使用的，术语“和/或”表示短语中的任何一个项目或项目组合。

[0022] 本文中描述的系统、方法和装置不应当被解释为以任何方式进行限制。相反，本公开内容单独地并且以彼此的各种组合和子组合涉及各种公开的实施例的所有新颖和非明显的特征和方面。所公开的系统、方法和装置不限于任何特定方面或特征或其组合，所公开的事物和方法也不要求存在任何一个或多个特定优点或解决问题。此外，所公开的实施例的任何特征或方面可以以彼此的各种组合和子组合使用。

[0023] 尽管为了方便呈现，以特定的顺序次序描述了一些所公开的方法的操作，但是应当理解，这种描述方式包括重新布置，除非下面阐述的特定语言需要特定顺序。例如，在某些情况下，顺序描述的操作可以重新布置或同时执行。此外，为了简单起见，附图可能未示出所公开的事物和方法可以与其他事物和方法结合使用的各种方式。另外，本说明书有时使用诸如“产生(produce)”、“生成(generate)”、“显示(display)”、“接收(receive)”、“发射(emit)”、“验证(verify)”、“执行(execute)”和“启动(initiate)”等术语来描述所公开的方法。这些术语是对所执行的实际操作的高级描述。与这些术语相对应的实际操作将根据具体顺序而变化，并且本领域普通技术人员可容易地辨别。

[0024] 本文中参考本公开的装置或方法而呈现的操作理论、科学原理或其他理论描述已经被提供以便方便理解，而非不旨在限制范围。所附权利要求中的装置和方法不限于以这种操作理论描述的方式起作用的那些装置和方法。

[0025] 所公开的任何方法可以实现为存储在一个或多个计算机可读介质(例如,计算机可读介质,诸如一个或多个光学介质盘、易失性存储器组件(诸如DRAM或SRAM)或非易失性存储器组件(诸如硬盘驱动器))上并且在计算机(例如,任何商用计算机,包括智能电话或包括计算硬件的其他移动设备)上执行的计算机可执行指令。用于实现所公开的技术的任何计算机可执行指令、以及在所公开的实施例的实现期间创建和使用的任何数据可以存储在一个或多个计算机可读介质(例如,计算机可读存储介质)上。计算机可执行指令可以是例如专用软件应用或经由web浏览器访问或下载的软件应用或其他软件应用(诸如远程计算应用)的一部分。这样的软件可以例如在单个本地计算机上执行(例如,使用在任何合适的商用计算机上执行的通用和/或基于块的处理器),或者使用一个或多个网络计算机在网络环境中执行(例如,经由因特网、广域网、局域网、客户端服务器网络(诸如云计算网络)或其他这样的网络)。

[0026] 为清楚起见,仅描述了基于软件的实现的某些所选择的方面。省略了本领域公知的其他细节。例如,应当理解,所公开的技术不限于任何特定的计算机语言或程序。例如,所公开的技术可以由用C、C++、Java或任何其他合适的编程语言编写的软件来实现。同样地,所公开的技术不限于任何特定计算机或硬件类型。合适的计算机和硬件的某些细节是公知的,并且不需要在本公开中详细阐述。

[0027] 此外,可以通过合适的通信装置上载、下载或远程访问任何基于软件的实施例(包括例如用于引起计算机执行任何所公开的方法的计算机可执行指令)。这种合适的通信装置包括例如因特网、万维网、内联网、软件应用、电缆(包括光纤电缆)、磁通信、电磁通信(包括RF、微波和红外通信)、电子通信、或其他这样的通信装置。

[0028] II.所公开的技术的简介

[0029] 超标量无序微架构使用大量电路资源来重命名寄存器,以数据流顺序调度指令,在错误推测之后清理,并且为了精确异常而按顺序退出结果。这包括昂贵的耗能电路,诸如深度、多端口寄存器文件、用于数据流指令调度唤醒的多端口内容可访问存储器(CAM)、以及多宽度总线多路复用器和旁路网络,所有这些都是资源密集型的。例如多读多写RAM的基于FPGA的实现通常需要复制、多周期操作、时钟倍频、存储体交织、实时值表和其他昂贵技术的混合。

[0030] 所公开的技术可以通过应用包括高指令级并行(ILP)、无序(OoO)、超标量执行在内的技术来实现能量效率和/或性能增强,同时避免处理器硬件和相关软件方面的大的复杂性和开销。在所公开的技术的一些示例中,包括多个处理器核的基于块的处理器使用被设计用于区域和能量有效的高ILP执行的显式数据图执行(EDGE)ISA。在一些示例中,EDGE架构和相关编译器的使用巧妙地消除了很多寄存器重命名、CAM和复杂性。在一些示例中,基于块的处理器的相应核可以存储或高速缓存可以重复执行的所获取和解码的指令,并且所获取和解码的指令可以被重用以潜在地实现功率降低和/或性能增加。

[0031] 在所公开的技术的某些示例中,EDGE ISA可以消除对一个或多个复杂架构特征的需要,包括寄存器重命名、数据流分析、错误推测恢复和有序退出,同时支持诸如C和C++等主流编程语言。在所公开的技术的某些示例中,基于块的处理器执行多个两个或更多个指令作为原子块。基于块的指令可以用于以更明确的方式表达程序数据流和/或指令流的语义,从而提高编译器和处理器性能。在所公开的技术的某些示例中,显式数据图执行指令集

架构(EDGE ISA)包括关于程序控制流的信息,该信息可以用于改进对不适当的控制流指令的检测,从而提高性能,节省存储器资源和/或节约能源。

[0032] 在所公开的技术的一些示例中,在指令块内组织的指令被原子地获取、执行和提交。在指令块内,指令通过操作数缓冲器直接将结果传送给消费指令。跨块,结果通过命名寄存器传送。由原子指令块内的指令产生的影响架构状态的中间结果在本地缓冲,直到指令块被提交。当提交指令块时,对由执行指令块的指令而产生的可见架构状态的更新对其他指令块是可见的。块内的指令以数据流顺序执行,这减少或消除了使用寄存器重命名并且提供了功率有效的OoO执行。在当前的工艺技术中,典型的BB-ISA设计可以支持多达128指令块。块大小可以影响处理器性能,但是通常,使用当前处理技术优选直到特定大小的更大块(例如,128或256字块)。除了其他因素之外,程序的特性可能会影响最佳指令块大小。

[0033] 有时,程序可能没有指令级并行性(ILP)以保持处理器的执行资源得到充分利用。执行资源包括可以在空间或时间基础上分配给针对进程或线程的特定上下文的处理器硬件。空间分配的执行资源在预定的基础上在一段时间内专门分配给其分配的上下文。因此,与线程可以使用资源直到线程被阻塞或线程等待,并且然后被指派给另一线程的其他方法不同,空间分配的执行资源在分配的持续时间内提供对相关联的上下文的独占访问。时间分配的执行资源在预定的基础上在两个或更多个上下文之间分配,但是在时间上共享。例如,第一上下文可以在所有奇数时钟周期访问时间分配的执行资源,并且第二上下文可以在所有偶数时钟周期访问时间分配的执行资源。可以使用更复杂的时间分配方案,但是先前选择该方案来分配资源。

[0034] 可以在空间上分配给上下文的执行资源的示例包括指令窗口、功能单元、数据高速缓存、指令高速缓存、结果缓冲器或物理寄存器文件。可以在在时间上分配给上下文的执行资源的示例包括:分支预测器、指令高速缓存或数据高速缓存。可以在空间上或在时间上分配这些示例性执行资源的任何合适组合。

[0035] 可以在多个“上下文”之间逻辑地划分执行资源。每个上下文包括由处理器托管的进程或线程的架构状态数据。例如,与进程或线程的单个上下文相关联的架构状态数据可以包括关联进程或线程的优先级信息、关联进程或线程的调度信息、子/父进程信息、进程间通信数据、权限信息、状态信息、进程或线程的标识符、架构寄存器文件值、指令块地址指针、指令块内的各个指令指针、存储器管理信息、或与进程或线程相关的其他合适信息。

[0036] 通过为每个上下文分配资源的子集,现在可以同时执行多个上下文。该方案有效地将处理器分成多个较小的处理器,通俗地称为“核裂变”。核裂变可以通过在运行时动态地分配资源(在空间上和/或在时间上)来执行。处理器可以从默认的“大”核(其中所有资源被分配给单个上下文)切换到“小”核(其中资源的各部分分别被不相交地分配给两个或更多个上下文)。资源的分配可以由操作系统控制,由程序本身控制,或者由处理器的控制电路自主控制。

[0037] 如下面将进一步讨论的,所公开的处理器核的某些示例可以被划分并且指派给一个、两个、三个、四个或更多个上下文。可以在空间上分配资源子集,其中资源的各部分以固定方式分配给上下文,该固定方式在分配资源的时段期间不变化。可以在时间上分配资源子集(基于时间的分配),其中资源的各部分在固定时间段内在两个或更多个上下文之间被分配。例如,第一上下文可以每个偶数时钟周期访问资源的一部分,并且第二上下文可以每

个奇数时钟周期访问资源的一部分。在一些示例中，处理器的一些资源在空间上被分配，而其他资源以时间方式被分配。例如，可以将一组空间分配的资源的一半分配给第一上下文，包括指令高速缓存、指令窗口、功能单元和寄存器文件，剩余部分被分配给第二上下文。可以将一组时间分配的资源的一半分配给第一上下文，包括分支预测器、数据高速缓存和指令解码器，其中第一上下文在指定的时钟周期期间访问时间分配的资源，第二上下文在未指定用于第一上下文的时钟周期期间被分配资源。如受益于本公开的本领域普通技术人员将容易确定的，根据所公开的示例分配资源可以向每个分配的上下文提供预定量的资源，并且避免了在将资源分配给其他线程之前等待执行线程变得饥饿或停滞的其他方法的问题。此外，空间和时间分配方案可以通过在预先确定的基础上而不是在运行时间的基础上(这可能更不可预测并且更容易受到由输入到线程的数据引起的变化和其他变化的影响)分配资源来减少用于实现分配的控制资源。

[0038] 编译器可以用于通过ISA显式地编码数据依赖性，从而减少或消除使处理器核控制逻辑在运行时重新发现依赖性的负担。使用预测执行，可以将块内分支转换为数据流指令，并且除了存储器依赖性之外的依赖性可以限于直接数据依赖性。所公开的目标形式编码技术允许块内的指令直接经由操作数缓冲器传送它们的操作数，从而减少对耗电的多端口物理寄存器文件的访问。

[0039] 在指令块之间，指令可以使用诸如存储器和寄存器等可见架构状态进行通信。因此，通过利用混合数据流执行模型，EDGE架构仍然可以支持命令式编程语言和顺序存储器语义，但是理想地还享有无序执行的好处，同时具有接近有序的功率效率和复杂性。

[0040] 在所公开的技术的一些示例中，下一块预测器可以用于预测要执行的下一指令块的地址。预测的地址可以用于推测性地获取或解码后续块以最小化在执行后续块时的延迟。下一块预测器可以接受以下一个或多个作为输入：执行指令块的块地址、目标指令块的块地址、是否采用指令块的一个或多个分支指令、在指令块头中编码的提示、和/或上下文标识符(例如，指派给进程或线程的标识符)。下一块预测器产生一个或多个预测目标地址作为输出。在一些示例中，为一个或多个目标地址提供置信水平。

[0041] 在一些示例中，可以推测性地加载或存储存储器。例如，存储器加载/存储队列可以用于路由由指令执行的存储器访问，从而最小化在存储器中存储和获取数据时的延迟。在一些示例中，可以利用指令级并行性来并行地执行块内的多个指令。在所公开的技术的一些示例中，BB-ISA向程序员公开指令块大小和加载存储队列大小的细节，以改进优化(由手动程序员或编译器)。

[0042] 如相关领域的普通技术人员将容易理解的，具有各种面积、性能和功率权衡的所公开的技术的一系列实现是可能的。

[0043] III.示例性的基于块的处理器

[0044] 图1是可以在所公开的技术的一些示例中实现的基于块的处理器100的框图10。处理器100被配置为根据指令集架构(ISA)执行指令的原子块，ISA描述处理器操作的多个方面，包括寄存器模型、由基于块的指令执行的多个定义的操作、存储器模型、中断和其他架构特征。基于块的处理器包括多个处理核110，包括处理器核111。

[0045] 如图1所示，处理器核经由核互连120彼此连接。核互连120在核110中的各个核、存储器接口140和输入/输出(I/O)接口145之间传送数据和控制信号。核互连120可以使用电、

光、磁或其他合适的通信技术发射和接收信号，并且可以根据特定的期望配置提供根据多种不同拓扑布置的通信连接。例如，核互连120可以具有交叉开关、总线、点对点总线或其他合适的拓扑。在一些示例中，核110中的任何一个可以连接到任何其他核，而在其他示例中，一些核仅连接到其他核的子集。例如，每个核可以仅连接到最近的4、8或20个相邻核。核互连120可以用于传输去往和来自核的输入/输出数据，以及传输去往和来自核的控制信号和其他信息信号。例如，每个核110可以接收和发射信号量，该信号量指示当前由每个相应核执行的指令的执行状态。在一些示例中，核互连120被实现为连接核110和存储器系统的导线，而在其他示例中，核互连可以包括用于在(多个)互连线路、开关和/或路由组件(包括有源信号驱动器和中继器)、或其他合适的电路上复用数据信号的电路。在所公开的技术的一些示例中，在处理器100内部和向/从处理器100传输的信号不限于全摆幅电数字信号，但是处理器可以被配置为包括差分信号、脉冲信号、或用于传输数据和控制信号的其他合适信号。

[0046] 在图1的示例中，处理器的存储器接口140包括用于连接到附加存储器的接口逻辑，例如，位于除了处理器100之外的另一集成电路上的存储器。如图1所示，外部存储器系统150包括L2高速缓存152和主存储器155。在一些示例中，L2高速缓存可以使用静态RAM(SRAM)来实现，并且主存储器155可以使用动态RAM(DRAM)来实现。在一些示例中，存储器系统150被包括在与处理器100的其他组件相同的集成电路上。在一些示例中，存储器接口140包括允许在不使用(多个)寄存器文件和/或处理器100的情况下在存储器中传输数据块的直接存储器访问(DMA)控制器。在一些示例中，存储器接口140可以包括用于管理和分配虚拟存储器以扩展可用的主存储器155的存储器管理单元(MMU)。

[0047] I/O接口145包括用于向其他组件接收和发送输入和输出信号的电路，诸如硬件中断、系统控制信号、外围接口、协处理器控制和/或数据信号(例如，用于图形处理单元、浮点协处理器、物理处理单元、数字信号处理器或其他协处理组件的信号)、时钟信号、信号量或其他合适的I/O信号。I/O信号可以是同步的或异步的。在一些示例中，I/O接口的全部或一部分使用存储器映射的I/O技术结合存储器接口140来实现。

[0048] 基于块的处理器100还可以包括控制单元160。控制单元可以经由核互连120或边带互连(未示出)与处理核110、I/O接口145和存储器接口140通信。控制单元160监督处理器100的操作。控制单元160可以执行的操作可以包括用于执行指令处理的核的分配和解除分配、任何核、寄存器文件、存储器接口140和/或I/O接口145之间的输入数据和输出数据的控制、执行流程的修改、以及验证分支指令的(多个)目标位置、指令头和控制流程中的其他变化。控制单元160还可以处理硬件中断，并且控制特殊系统寄存器的读取和写入，例如存储在一个或多个寄存器文件中的程序计数器(指令块地址寄存器)。在所公开的技术的一些示例中，控制单元160至少部分使用一个或多个处理核110来实现，而在其他示例中，控制单元160使用非基于块的处理核(例如，耦合到存储器的通用RISC处理核)来实现。在一些示例中，控制单元160至少部分使用以下中的一个或多个来实现：硬连线有限状态机、可编程微码、可编程门阵列或其他合适的控制电路。在替代示例中，控制单元功能可以由一个或多个核110执行。

[0049] 控制单元160包括用于将指令块分配给处理器核110的调度器。如本文中使用的，调度器分配是指用于指导指令块的操作的硬件，包括启动指令块映射、获取、解码、执行、提

交、中止、空闲和刷新指令块。在一些示例中，硬件接收使用计算机可执行指令生成的信号以指导指令调度器的操作。处理器核110在指令块映射期间被指派给指令块。所述的指令操作阶段是出于说明的目的，并且在所公开的技术的一些示例中，某些操作可以被组合、省略、分成多个操作，或者添加附加操作。控制单元160还可以用于将处理器核内的功能资源分配给上下文。例如，指令高速缓存、指令窗口、功能单元、寄存器文件、分支预测器(下一块预测器)和/或数据高速缓存可以由控制单元160分配。

[0050] 基于块的处理器100还包括时钟生成器170，时钟生成器170将一个或多个时钟信号分配给处理器内的各种组件(例如，核110、互连120、存储器接口140和I/O接口145)。在所公开的技术的一些示例中，所有这些组件共享公共时钟，而在其他示例中，不同组件使用不同时钟，例如，具有不同时钟频率的时钟信号。在一些示例中，时钟的一部分被门控以允许在一些处理器组件未被使用时节省功率。在一些示例中，使用锁相环(PLL)生成时钟信号以生成固定、恒定频率和占空比的信号。接收时钟信号的电路可以在单个边沿(例如，上升沿)被触发，而在其他示例中，至少一些接收电路由时钟上升沿和下降沿触发。在一些示例中，时钟信号可以光学地或无线地传输。IV.示例性的基于块的处理器核

[0051] 图2是进一步详述可以在所公开的技术的某些示例中使用的用于基于块的处理器100并且具体地是基于块的处理器核(处理器核111)之一的实例的示例微架构的框图200。为了便于说明，示例性的基于块的处理器核111分五个阶段示出：指令获取(IF)、解码(DC)、操作数获取、执行(EX)和存储器/数据访问(LS)。然而，如受益于本公开的相关领域的普通技术人员将容易理解的，对所示微架构的修改(诸如添加/移除阶段、添加/移除执行操作的单元以及其他实现细节)可以适用于基于块的处理器的特定应用。

[0052] 在所公开的技术的一些示例中，处理器核111可以用于执行和提交程序的指令块。指令块是包括指令块头和多个指令的基于块的处理器指令的原子集合。如下面将进一步讨论的，指令块头可以包括描述指令块的执行模式的信息和可以用于进一步定义指令块内的多个指令中的一个或多个指令的语义的信息。取决于所使用的特定ISA和处理器硬件，还可以在指令的执行期间使用指令块头，以通过例如允许提前获取指令和/或数据来改进执行指令块的性能，改进分支预测、推测执行，提高能源效率，以及改进代码紧凑性。

[0053] 指令块的指令可以是明确地编码指令块的生产者消费者指令之间的关系的数据流指令。特别地，指令可以通过仅为目标指令保留的操作数缓冲器将结果直接传送给目标指令。存储在操作数缓冲器中的中间结果通常对执行核之外的核不可见，因为块原子执行模型仅在指令块之间传递最终结果。当指令块被提交时，由于执行原子指令块的指令而产生的最终结果在执行核之外变得可见。因此，由每个指令块生成的可见架构状态可以表现为执行核之外的单个事务，并且中间结果通常在执行核之外是不可观察的。但是，程序员可能会发现在调试在基于块的处理器上运行的程序时，中间结果很有用。如本文所述，处理器核111可以包括调试模式，其中可以通过使用专门被保留用于调试的数据路径来在执行核之外观察中间结果。

[0054] 如图2所示，处理器核111包括控制单元205，控制单元205可以从其他核接收控制信号并且生成控制信号以调节核操作，并且处理器核111使用指令调度器206调度核内的指令流。控制单元205可以包括用于检查核状态和/或配置处理器核111的操作模式的状态访问逻辑207。控制单元205可以包括用于在处理器核111的一个或多个操作模式期间生成控

制信号的执行控制逻辑208。可以由控制单元205和/或指令调度器206执行的操作可以包括用于在任何核、寄存器文件、存储器接口140和/或I/O接口145之间执行指令处理、输入数据和输出数据的控制的核的分配和解除分配。控制单元205还可以处理硬件中断，并且控制特殊系统寄存器的读取和写入，例如存储在一个或多个寄存器文件中的程序计数器。在所公开的技术的其他示例中，控制单元205和/或指令调度器206使用非基于块的处理核(例如，耦合到存储器的通用RISC处理核)来实现。在一些示例中，控制单元205、指令调度器206、状态访问逻辑207和/或执行控制逻辑208至少部分使用以下中的一个或多个来实现：硬连线有限状态机、可编程微码、可编程门阵列或其他合适的控制电路。

[0055] 控制单元205可以解码指令块头以获取关于指令块的信息。例如，可以通过各种执行标记在指令块头中指定指令块的执行模式。解码的执行模式可以存储在执行控制逻辑208的寄存器中。基于执行模式，执行控制逻辑208可以生成控制信号以调节核操作并且调度核111内的指令流，诸如通过使用指令调度器206。例如，在默认执行模式期间，执行控制逻辑208可以对在处理器核111的一个或多个指令窗口(例如，210、211)上执行的一个或多个指令块的指令进行排序。具体地，每个指令可以通过指令获取、解码、操作数获取、执行和存储器/数据访问级来排序，以便指令块的指令可以流水线化并且并行执行。当其操作数可用时，指令准备好执行，并且指令调度器206可以选择执行指令的顺序。指令可以从寄存器文件或其他先前指令接收其操作数。指令的结果可以发送到指令块内的其他目标指令或发送到寄存器文件。可以同时检查指令窗口内的一个或多个指令以确定指令块的哪些指令准备好执行。

[0056] 作为另一示例，在调试执行模式期间，执行控制逻辑208可以单步执行在处理器核111的一个或多个指令窗口上执行的指令块的指令。具体地，可以通过指令获取和解码阶段对每个指令进行排序，并且每个指令可以一次执行一个指令。例如，可以在接收到执行单步操作的指示时执行单个指令。例如，可以经由状态访问逻辑207接收执行单步操作的指示。

[0057] 状态访问逻辑207可以包括用于其他核和/或处理器级控制单元(诸如图1的控制单元160)的接口以与核111通信以及访问核111的状态。例如，状态访问逻辑207可以连接到核互连(诸如图1的核互连120)，而其他核可以经由控制信号、消息、读取和写入寄存器等进行通信。

[0058] 状态访问逻辑207可以包括控制状态寄存器或用于修改和/或检查指令块和/或核状态的模式和/或状态的其他逻辑。作为示例，核状态可以指示特定资源是否静态地分配给上下文，诸如指令高速缓存、数据高速缓存、指令窗口、加载存储队列、寄存器文件和/或功能单元。作为另一示例，核状态可以指示特定资源是否在时间上被分配给上下文，诸如数据高速缓存、寄存器文件、加载存储队列和/或下一分支预测器。状态访问逻辑207还可以存储指示指令块是否准备好提交、指令块是否正在执行提交、以及指令块是否空闲的数据。状态访问逻辑207还可以存储指示块是否已经跨越多个执行资源分片折叠的数据，如下面将进一步详述的。作为另一示例，指令块的状态可以包括指示指令块是正在执行的最老的指令块的标记或标志、以及推测性地指示指令块正在执行的标记。

[0059] 在一些示例中，状态访问逻辑207中的一些或所有寄存器可以被映射到被保留用于由基于块的处理器使用的唯一存储器位置。在一些示例中，可以使用基于块的处理器的通用存储器读取和写入指令来访问状态访问逻辑207寄存器。附加地或替代地，可以使用特

定的读取和写入指令来访问状态访问逻辑207寄存器。因此，一个核可以通过读取所指派的存储器位置和/或使用特定指令来检查配置状态。

[0060] 状态访问逻辑207可以包括用于配置和/或重新配置核以在不同操作模式下操作的寄存器或其他逻辑，如本文中进一步描述的。例如，状态访问逻辑207可以包括控制特定资源是在空间上还是在在时间上分配给特定上下文的控制寄存器位。

[0061] 指令窗口210和211中的每个可以从输入端口220、221和222中的一个或多个接收指令和数据，输入端口220、221和222连接到互连总线和指令高速缓存227，指令高速缓存227又连接到指令解码器228和229。另外的控制信号也可以在附加输入端口225上接收。指令解码器228和229中的每个解码指令块的指令并且将解码的指令存储在位于每个相应指令窗口210和211中的存储器存储215和216中。

[0062] 处理器核111还包括耦合到L1(第一级)高速缓存235的寄存器文件230。寄存器文件230存储在基于块的处理器架构中定义的寄存器的数据，并且可以具有一个或多个读取端口和一个或多个写入端口。例如，寄存器文件可以包括用于在寄存器文件中存储数据的两个或更多个写入端口以及用于从寄存器文件内的各个寄存器读取数据的多个读取端口。在一些示例中，单个指令窗口(例如，指令窗口210)一次只能访问寄存器文件的一个端口，而在其他示例中，指令窗口210可以访问一个读取端口和一个写入端口，或者可以同时访问两个或更多个读取端口和/或写入端口。在一些示例中，寄存器文件230可以包括64个寄存器，每个寄存器保持32位数据的字。(除非另有说明，否则本申请将32位数据称为字。)在一些示例中，寄存器文件230内的一些寄存器可以被分配用于特殊目的。例如，一些寄存器可以专用作系统寄存器，系统寄存器的示例包括存储常量值(例如，全零字)、(多个)程序计数器(PC)(其指示正在执行的程序线程的当前地址)、物理核号、逻辑核号、核指派拓扑、核控制标记、处理器拓扑或其他合适的专用目的的寄存器。在一些示例中，存在多个程序计数器寄存器(每个程序计数器一个)，以允许跨一个或多个处理器核和/或处理器并发执行多个执行线程。在一些示例中，程序计数器被实现为指定的存储器位置而不是寄存器文件中的寄存器。在一些示例中，系统寄存器的使用可以受操作系统或其他监督计算机指令的限制。在一些示例中，寄存器文件230被实现为触发器阵列，而在其他示例中，寄存器文件可以使用锁存器、SRAM或其他形式的存储器存储来实现。给定处理器(例如，处理器100)的ISA规范指定如何定义和使用寄存器文件230内的寄存器。

[0063] 在一些示例中，处理器100包括由多个处理器核共享的全局寄存器文件。在一些示例中，取决于处理器ISA和配置，可以组合与处理器核相关联的各个寄存器文件以静态地或动态地形成更大的文件。

[0064] 如图2所示，指令窗口210的存储器存储215包括多个解码指令241、左操作数(LOP)缓冲器242、右操作数(ROP)缓冲器243、谓词缓冲器244和指令记分板245。在所公开的技术的示例中，指令块的每个指令被分解为一行解码指令、左右操作数和记分板数据，如图2所示。解码的指令241可以包括存储为位级控制信号的指令的部分或完全解码版本。操作数缓冲器242和243存储操作数(例如，从寄存器文件230接收的寄存器值、从存储器接收的数据、在指令内编码的立即操作数、由较早发出的指令计算的操作数、或其他操作数值)，直到它们的解码指令已经准备好执行。指令操作数从操作数缓冲器242和243读取，而不是从寄存器文件读取。

[0065] 第二指令窗口211的存储器存储216存储与存储器存储215类似的指令信息(解码的指令、操作数和记分板),但是为简单起见而未在图2中示出。根据ISA约束并且控制单元205的指示,指令块可以由第二指令窗口211相对于第一指令窗口同时或顺序执行。

[0066] 在所公开的技术的一些示例中,前端流水线级IF(指令获取)和DC(指令解码)可以与后端流水线级(IS(发出)、EX(执行)、LS(加载/存储)解耦合。在一个实施例中,控制单元可以在每个时钟周期获取两个指令并且将其解码到指令窗口210和211中的每个中。在替代实施例中,控制单元可以在每个时钟周期获取一个、四个或另一数目的指令并且将其解码到对应数目的指令窗口中。控制单元205提供指令窗口数据流调度逻辑,用于使用记分板245监测每个解码指令的输入(例如,每个相应指令的(多个)谓词和(多个)操作数)的就绪状态。当特定解码指令的所有输入就绪时,指令准备好发出。控制逻辑205然后在每个周期启动一个或多个下一指令(例如,编号最小的就绪指令)的执行,并且其解码的指令和输入操作数被发送到一个或多个功能单元260用于执行。解码指令还可以编码多个就绪事件。控制逻辑205中的调度器接受来自其他源的这些和/或事件,并且更新窗口中的其他指令的就绪状态。因此,从处理器核111的就绪零输入指令、零输入指令所针对的指令等开始,执行继续。

[0067] 解码指令241不需要以与它们被布置在指令窗口210的存储器215中相同的顺序执行。而是,指令记分板245用于跟踪解码指令的依赖性,并且当依赖性满足时,调度相关的个体解码指令以供执行。例如,当已经满足相应指令的依赖性时,可以将对相应指令的引用推送到就绪队列,并且可以从就绪队列以先进先出(FIFO)顺序调度指令。存储在记分板245中的信息可以包括但不限于相关指令的执行谓词(诸如指令是否正在等待计算谓词位以及如果谓词位为真或假则指令是否执行)、操作数对指令的可用性、或在执行相关个体指令之前所需要的其他先决条件。

[0068] 在一个实施例中,记分板245可以包括由指令解码器228初始化的解码就绪状态、以及由控制单元205在执行指令期间初始化的活动就绪状态。例如,解码的就绪状态可以编码相应指令是否已经被解码,可能经由广播信道等待谓词和/或某个(些)操作数,或者立即准备发出。活动就绪状态可以编码相应指令是否等待谓词和/或某个(些)操作数,准备好发出,或者已经发出。解码的就绪状态可以在块复位或块刷新时清除。在分支到新指令块时,解码的就绪状态和活动就绪状态被清除(块或核复位)。然而,当在核上重新执行指令块时,诸如当它分支回自身时(块刷新),仅清除活动就绪状态。块刷新可以立即发生(当指令块分支到自身时)或者在执行很多其他中间指令块之后发生。因此可以保留指令块的解码就绪状态,从而不必重新获取和解码块的指令。因此,块刷新可以用于在循环和其他重复程序结构中节省时间和能量。

[0069] 存储在每个指令窗口中的指令的数目通常对应于指令块内的指令的数目。在一些示例中,指令块内的指令的数目可以是32、64、128、1024或另一数目的指令。在所公开的技术的一些示例中,跨处理器核内的多个指令窗口分配指令块。在一些示例中,指令窗口210、211可以被逻辑分区,使得可以在单个处理器核上执行多个指令块。例如,可以在一个核上执行一个、两个、四个或另一数目的指令块。相应指令块可以彼此同时或顺序执行。

[0070] 可以使用位于处理器核111内的控制单元205来分配和调度指令。控制单元205协调从存储器获取指令,解码指令,一旦将指令加载到相应的指令窗口中就执行指令,数据流

入/流出处理器核111，以及由处理器核输入和输出的控制信号。例如，控制单元205可以包括用于调度指令的就绪队列，如上所述。存储在位于每个相应指令窗口210和211中的存储器存储215和216中的指令可以原子方式执行。因此，可以在核内本地缓存对由所执行的指令影响的可见架构状态(诸如寄存器文件230和存储器)的更新，直到提交指令。控制单元205可以确定指令何时准备好提交，对提交逻辑进行排序，以及发出提交信号。例如，指令块的提交阶段可以在所有寄存器写入被缓冲，对存储器的所有写入被缓冲以及计算分支目标时开始。当对可见架构状态的更新完成时，可以提交指令块。例如，当寄存器写入被写入寄存器文件，将存储发送到加载/存储单元或存储器控制器，以及生成提交信号时，可以提交指令块。控制单元205还至少部分控制功能单元260到每个相应指令窗口的分配。

[0071] 如图2所示，具有多个执行流水线寄存器255的第一路由器250用于将数据从指令窗口210和211中的任一个发送到一个或多个功能单元260，功能单元260可以包括但不限于整数ALU(算术逻辑单元)(例如，整数ALU 264和265)、浮点单元(例如，浮点ALU 267)、移位/旋转逻辑(例如，桶形移位器268)或其他合适的执行单元，功能单元260可以包括图形函数、物理函数和其他数学运算。来自功能单元260的数据然后可以通过第二路由器270路由到输出290、291和292，路由回操作数缓冲器(例如，LOP缓冲器242和/或ROP缓冲器243)，或者反馈回另一功能单元，这取决于正在执行的特定指令的要求。第二路由器270可以包括可以用于发出存储器指令的加载/存储队列275、存储从核输出到存储器的数据的数据高速缓存277、以及加载/存储流水线寄存器278。

[0072] 核还包括控制输出295，控制输出295用于指示例如指令窗口210或211中的一个或多个的所有指令的执行何时完成。当指令块的执行完成时，指令块被指定为“已经提交”，并且来自控制输出295的信号又可以由基于块的处理器100内的其他核和/或由控制单元160使用以启动其他指令块的调度、获取和执行。第一路由器250和第二路由器270都可以将数据发送回指令(例如，作为指令块内的其他指令的操作数)。

[0073] 如相关领域的普通技术人员将容易理解的，个体核内的组件不限于图2所示的那些，而是可以根据特定应用的要求而变化。例如，核可以具有更少或更多的指令窗口，单个指令解码器可以由两个或更多个指令窗口共享，并且所使用的功能单元的数目和类型可以变化，这取决于基于块的处理器的特定目标应用。在利用指令核来选择和分配资源时应用的其他考虑因素包括性能要求、能量使用要求、集成电路裸片、工艺技术和/或成本。

[0074] 对于受益于本公开的相关领域的普通技术人员将很清楚的是，可以通过设计和在指令窗口内分配资源以及通过处理器核110的控制逻辑205来对处理器性能进行折衷(例如，指令窗口)。区域、时钟周期、能力和限制基本上确定各个核110的实现性能和基于块的处理器核110的吞吐量。

[0075] 指令调度器206可以具有不同的功能。在某些更高性能的示例中，指令调度器是高度并发的。例如，每个周期，(多个)解码器将指令的解码就绪状态和解码指令写入一个或多个指令窗口中，选择要发出的下一指令，并且作为响应，后端发送就绪事件——针对特定指令的输入槽(谓词、左操作数、右操作数等)的目标就绪事件或针对所有指令的广播就绪事件。每指令就绪状态位与解码的就绪状态一起可以用于确定指令准备好发出。

[0076] 在一些示例中，指令调度器206使用存储装置(例如，先进先出(FIFO)队列、内容可寻址存储器(CAM))来实现，该存储装置存储指示用于根据所公开的技术来调度指令块的执

行的信息的数据。例如，关于指令依赖性、控制转移、推测、分支预测和/或数据加载和存储的数据被布置在存储装置中以支持确定将指令块映射到处理器核。例如，指令块依赖性可以与存储在FIFO或CAM中的标签相关联，并且稍后由用于将指令块映射到一个或多个处理器核的选择逻辑访问。在一些示例中，指令调度器206使用耦合到存储器的通用处理器来实现，该存储器被配置为存储用于调度指令块的数据。在一些示例中，指令调度器206使用专用处理器或使用耦合到存储器的基于块的处理器核来实现。在一些示例中，指令调度器206被实现为耦合到存储器的有限状态机。在一些示例中，在处理器(例如，通用处理器或基于块的处理器核)上执行的操作系统生成可以至少部分用于利用指令调度器206来调度指令块的优先级、预测和其他数据。对于受益于本公开的相关领域的普通技术人员将容易明白的是，可以使用在集成电路、可编程逻辑或其他合适的逻辑中实现的其他电路结构来实现用于指令调度器206的硬件。

[0077] 在一些情况下，调度器206接受尚未解码的目标指令的事件，并且还必须禁止重发所发出的就绪指令。指令可以是非谓词的或谓词的(基于真或假条件)。谓词指令在被另一指令的谓词结果作为目标并且该结果与谓词条件相匹配之前不会就绪。如果关联的谓词不匹配，则指令永远不会发出。在一些示例中，可以推测性地发出和执行谓词指令。在一些示例中，处理器可以随后检查推测性地发出和执行的指令是否被正确推测。在一些示例中，可以重新执行错误推定的发出指令和消耗其输出的块中的指令的特定传递闭包，或者取消错误指定的副作用。在一些示例中，发现错误指定的指令导致整个指令块的完整回滚和重新执行。

[0078] V.指令块的示例流

[0079] 现在转向图3的图300，示出了包括多个可变长度指令块311-315(A-E)的基于块的指令流的一部分310。指令流可以用于实现用户应用、系统服务或任何其他合适的用途。在图3所示的示例中，每个指令块以指令头开始，后面是不同数目的指令。例如，“A”指令块311包括四字头320和七十个指令321。在一些示例中，指令块被划分或“折叠”以适合已经被分配用于执行指令块的资源。例如，根据所公开的技术的基于块的处理器可以包括被配置为跟踪状态并且执行32字组块中的指令的指令窗口。因此，A指令块311可以分成三个部分。这些部分可以分配给三个执行分片，或者第一指令块部分的结果可以在传递到指令块的后续第二部分之前临时存储在结果缓冲器中。类似地，“B”指令块312是64字指令块，其可以被分成两个32字部分330，并且“E”指令块315是128字指令块，其可以被分成四个32字部分340。如受益于本公开的本领域普通技术人员将容易理解的，取决于特定实现中的资源，指令块可以被划分为不同大小的部分。此外，适于生成用于所公开的处理器的机器指令的编译器可以通过例如分配指令以落入指令块的某些部分内来在指令块内布置指令以提高处理器性能。在其他配置中，可以组合处理器资源，使得能够获取指令块(例如，指令块311或315)并且将其作为单个指令块执行。在一些示例中，根据所公开的技术的处理器可以被重新配置为以两种或更多种不同的这种模式操作。

[0080] 所示的特定的示例性指令头320包括多个数据字段，这些数据字段部分地控制指令块内的指令的执行，并且还实现改进的性能增强技术，包括例如分支预测、推测执行、懒惰评估和/或其他技术。指令头320还包括ID位，该ID位指示头是指令头而不是指令。指令头320还包括指令块大小的指示。指令块大小可以是比一个更大的指令组块，例如，指令块中

包含的4指令组块的数目。换言之，块的大小被移位4位，以便压缩被分配用于指定指令块大小的头空间。因此，大小值为0指示最小大小的指令块，其是块头，其后面是四个指令。在一些示例中，指令块大小表示为字节数、字数、n字组块的数目、地址、地址偏移，或者使用用于描述指令块的大小的其他合适的表达来表达。在一些示例中，指令块大小由指令块头和/或尾部中的终止位模式指示。

[0081] 指令块头320还可以包括指示特殊指令执行要求的执行标记。例如，取决于特定应用，可以针对某些指令块禁止分支预测或存储器依赖性预测。作为另一示例，执行标记可以用于控制指令块是以默认执行模式还是以调试执行模式执行。

[0082] 在所公开的技术的一些示例中，指令头320包括指示编码数据是指令头的一个或多个标识位。例如，在一些基于块的处理器ISA中，最低有效位空间中的单个ID位总是被设置为二进制值1以指示有效指令块的开始。在其他示例中，不同的比特编码可以用于(多个)标识比特。在一些示例中，指令头320包括指示对其相关指令块进行编码的ISA的特定版本的信息。

[0083] 块指令头还可以包括用于例如分支预测、控制流确定和/或坏跳检测的多个块退出类型。退出类型可以指示分支指令的类型，例如：指向存储器中的下一连续指令块的顺序分支指令；作为相对于偏移而计算的存储器地址处的另一指令块的分支的偏移指令；子程序调用或子程序返回。通过对指令头中的分支出口类型进行编码，分支预测器可以在获取和/或解码相同指令块内的分支指令之前至少部分开始操作。

[0084] 指令块头320还包括标识被指派用于存储操作的加载存储队列标识符的存储掩码。指令块头还可以包括标识相关指令块将写入哪个(些)全局寄存器的写入掩码。在指令块可以完成之前，相关寄存器文件必须接收对每个条目的写入。在一些示例中，基于块的处理器架构不仅可以包括标量指令，还可以包括单指令多数据(SIMD)指令，这些指令实现在单个指令内具有更大数目的数据操作数的操作。

[0085] VI.示例块指令目标编码

[0086] 图4是描绘C语言源代码的两个部分410和415以及它们各自的指令块420和425(以汇编语言)的示例的图400，示出了基于块的指令如何可以明确地编码它们的目标。高级C语言源代码可以由目标是基于块的处理器的编译器转换为低级汇编语言和机器代码。高级语言可以抽象出底层计算机架构的很多细节，以便程序员可以专注于程序的功能。相反，机器代码根据目标计算机的ISA对程序进行编码，以便可以使用计算机的硬件资源在目标计算机上执行。汇编语言是一种人类可读形式的机器代码。

[0087] 在以下示例中，汇编语言指令使用以下命名法：“I[<number>]指定指令块内的指令编号，其中编号从指令头之后的指令的零开始，并且指令编号针对每个连续指令递增；指令的操作(诸如READ、ADDI、DIV等)跟随指令编号；可选值(诸如立即值1)或寄存器引用(诸如寄存器0的R0)跟随操作；以及要接收指令结果的可选目标跟随值和/或操作。每个目标可以到另一指令、到其他指令的广播信道、或者在提交指令块时可以对另一指令块可见的寄存器。指令目标的示例是T[1R]，T[1R]以指令1的右操作数为目标。寄存器目标的示例是W[R0]，其中目标被写入寄存器0。

[0088] 在图400中，指令块420的前两个READ指令430和431分别以ADD指令432的右(T[2R])和左(T[2L])操作数作为目标。在图示ISA中，读取指令是从全局寄存器文件读取的唯

一指令;但是任何指令都可以以全局寄存器文件作为目标。当ADD指令432接收到两个寄存器读取的结果时,它将变为就绪并且执行。

[0089] 当TLEI(小于等于立即测试)指令433从ADD接收其单个输入操作数时,它将变为就绪并且执行。然后,测试产生谓词操作数,该谓词操作数在通道1(B[1P])上广播给在广播通道上监听的所有指令,这些指令在该示例中是两个谓词分支指令(BRO_T 434和BRO_F435)。接收匹配谓词的分支将触发。

[0090] 还示出了指令块420的依赖图440,作为指令节点的阵列450及其对应的操作数目标455和456。这示出了指令块420、对应的指令窗口条目和由指令表示的基础数据流图之间的对应关系。这里,解码的指令READ 430和READ 431准备好发出,因为它们没有输入依赖性。当它们发出并且执行时,从寄存器R6和R7读取的值被写入ADD432的右和左操作数缓冲器,以将ADD 432的左和右操作数标记为“就绪”。结果,ADD 432指令变为就绪,向ALU发出,执行,并且将总和写入TLEI 433的左操作数。

[0091] 在一些示例中,可以以较小的部分处理较大的指令块以允许分配处理器资源以执行该块。例如,适于执行多达128字指令块的处理器可以被配置为以32字部分处理这些块。然而,在某些示例中,可能需要在指令块部分之间传递指令结果。例如,subi#1指令460既以指令编号4的右操作数(“T[4R]”)为目标,又以指令编号100的右操作数(“T[100R]”)为目标。因此,当处理器被配置为以较小的部分处理指令块时,结果缓冲器用于临时存储指令的目标操作数,直到目标指令465可以消耗其(多个)输入操作数。

[0092] 作为比较,传统的无序RISC或CISC处理器将使用附加的硬件复杂性、功率、面积以及降低的时钟频率和性能来在运行时动态地构建依赖图。但是,依赖图在编译时是静态已知的,EDGE编译器可以通过ISA直接编码指令之间的生产者消费者关系,从而使微架构不需要动态地重新发现它们。这可以实现更简单的微架构,以减少面积、功率并且提高频率和性能。

[0093] VII.示例性的基于块的指令格式

[0094] 图5是示出用于指令头510、通用指令520、分支指令530和存储器访问指令540(例如,存储器加载或存储指令)的指令格式的一般化示例的图。指令格式可以用于根据在指定操作模式的指令头中指定的多个执行标记而执行的指令块。根据位数来标记每个指令头或指令。例如,指令头510包括四个32位字,并且从其最低有效位(lsb)(位0)标记到其最高有效位(msb)(位127)。如图所示,指令头包括写入掩码字段、多个退出类型字段511、多个执行标记字段、指令块大小字段512和指令头ID位(指令头的最低有效位)。在一些示例中,指令头510包括可以用于控制指令块执行和性能的附加方面的附加元数据515。

[0095] 图5中描绘的执行标记字段占用指令块头510的位6至13,并且指示用于执行指令块的一个或多个操作模式。例如,操作模式可以包括核裂变操作(例如,通过指定是使用全部或部分还是使用一定量的执行资源来执行指令块)、矢量模式操作、分支预测器抑制、存储器依赖性预测器禁止、块同步、块后中断、块前中断、块下降和/或有序或确定性指令执行。块同步标记占用指令块的第9位,并且在设置为逻辑1时禁止指令块的推测执行。

[0096] 退出类型字段包括可以用于指示在指令块内编码的控制流指令的类型的数据。例如,退出类型字段可以指示指令块包括以下中的一个或多个:顺序分支指令、偏移分支指令、间接分支指令、调用指令和/或返回指令。在一些示例中,分支指令可以是用于在指令块

之间传送控制流的任何控制流指令，包括相对和/或绝对地址，以及使用条件或无条件谓词。除了确定隐式控制流指令之外，退出类型字段还可以用于分支预测和推测执行。可以计算下一指令块的地址，以便推测性地执行和存储在指令块地址寄存器中。在一些示例中，可以在退出类型字段中编码多达六个退出类型，并且可以通过例如检查指令块中的控制流指令来确定字段与对应的显式或隐式控制流指令之间的对应关系。

[0097] 所示的通用块指令520被存储为一个32位字，并且包括操作码字段、谓词字段、广播ID字段(BID)、第一目标字段(T1)和第二目标字段(T2)。对于消费者多于目标字段的指令，编译器可以使用移动指令构建扇出树，或者可以为广播指派高扇出指令。广播支持通过轻量级网络向核中的任何数目的消费者指令发送操作数。

[0098] 虽然通用指令520概述的通用指令格式可以表示由基于块的处理器处理的一些或所有指令，但是本领域技术人员将容易理解，即使对于ISA的特定示例，一个或多个指令字段可以偏离特定指令的通用格式。操作码字段指定由指令520执行的(多个)操作，诸如存储器读/写、寄存器加载/存储、加、减、乘、除、移位、旋转、系统操作或其他合适的指令。谓词字段指定用于执行指令的条件。例如，谓词字段可以指定值“真”，并且只有在对应的条件标记与指定的谓词值相匹配时才会执行指令。在一些示例中，谓词字段至少部分指定用于比较谓词的内容，而在其他示例中，执行基于由先前指令(例如，指令块中的先前指令)设置的标记作为谓词。在一些示例中，谓词字段可以指定将始终或从不执行指令。因此，通过减少分支指令的数目，谓词字段的使用可以实现更密集的目标代码、改进的能量效率和改进的处理器性能。

[0099] 目标字段T1和T2指定基于块的指令的结果被发送到的指令。例如，指令槽5处的ADD指令可以指定其计算结果将被发送到槽3和10处的指令，包括操作数槽的指定(例如，左操作、右操作数或谓词操作数)。取决于特定指令和ISA，所示目标字段中的一个或两个可以由其他信息替换，例如，第一目标字段T1可以由立即操作数、附加操作码、指定两个目标等替换。

[0100] 分支指令530包括操作码字段、谓词字段、广播ID字段(BID)和偏移字段。操作码和谓词字段在格式和功能上类似于关于通用指令所描述的。偏移可以以四个指令组为单位表示，从而扩展了可以执行分支的存储器地址范围。用通用指令520和分支指令530示出的谓词可以用于避免指令块内的附加分支。例如，特定指令的执行可以基于先前指令的结果(例如，两个操作数的比较)作为谓词。如果谓词为假，则指令不会提交由特定指令计算的值。如果谓词值与所需要的谓词不匹配，则指令不会发出。例如，如果发送了伪谓词值，则会发出BRO_F(谓词假)指令。

[0101] 应当容易理解，如本文中使用的，术语“分支指令”不限于将程序执行改变为相对存储器位置，还包括跳转到绝对或符号存储器位置、子程序调用和返回、以及可以修改执行流程的其他指令。通过改变指令块地址寄存器的值来修改执行流程(例如，使用分支指令来隐式地改变该值以指向要执行的下一指令块的存储器位置)，而在其他示例中，可以通过修改存储在虚拟存储器中的指定位置的值来改变执行流程(例如，通过被配置为检测对指定存储器位置的读取和写入并且将值存储/加载到指令块地址寄存器的存储器控制器)。在一些示例中，跳转寄存器分支指令用于跳转到存储在寄存器中的存储器位置。在一些示例中，子程序调用和返回分别使用跳转以及链接和跳转寄存器指令来实现。

[0102] 存储器访问指令540格式包括操作码字段、谓词字段、广播ID字段(BID)、立即字段(IMM)偏移字段和目标字段。操作码、广播、谓词字段在格式和功能上类似于关于通用指令所描述的。例如,特定指令的执行可以基于先前指令的结果(例如,两个操作数的比较)作为谓词。如果谓词为假,则指令不会提交由特定指令计算的值。如果谓词值与所需要的谓词不匹配,则指令不会发出。立即字段(例如,移位多个位)可以用作发送到加载或存储指令的操作数的偏移。操作数加上(移位)立即偏移用作加载/存储指令的存储器地址(例如,从存储器读取数据或将数据存储到存储器中的地址)。

[0103] 应当容易理解,如本文中使用的,术语“分支指令”不限于将程序执行改变为相对存储器位置,还包括跳转到绝对或符号存储器位置、子程序调用和返回、以及可以修改执行流程的其他指令。在一些示例中,通过改变系统寄存器(例如,程序计数器PC或指令指针)的值来修改执行流程,而在其他示例中,可以通过修改存储在存储器中的指定位置的值来改变执行流程。在一些示例中,跳转寄存器分支指令用于跳转到存储在寄存器中的存储器位置。在一些示例中,子程序调用和返回分别使用跳转以及链接和跳转寄存器指令来实现。

[0104] VIII.处理器核的示例状态

[0105] 图6是示出基于块的计算机的处理器核的状态600的进展的示例的流程图。基于块的计算机由共同用于运行或执行软件程序的多个处理器核组成。该程序可以用各种高级语言编写,然后使用针对基于块的处理器的编译器针对基于块的处理器被编译。编译器可以发出代码,这些代码当在基于块的处理器上运行或执行时将执行由高级程序指定的功能。编译的代码可以存储在可以由基于块的处理器访问的计算机可读存储器中。编译的代码可以包括分组成一系列指令块的指令流。在执行期间,基于块的处理器可以执行一个或多个指令块以执行程序的功能。通常,程序将包括比在任何时间在核上执行的指令块更多的指令块。因此,程序的块被映射到各个核,核执行由块指定的工作,然后各个核上的块被替换为不同的块,直到程序完成。一些指令块可以多次执行,诸如在程序的循环或子程序期间。可以在每次执行指令块时创建指令块的“实例”。因此,指令块的每次重复可以使用指令块的不同实例。当程序运行时,可以基于架构约束、可用硬件资源和程序的动态流程将相应指令块映射到处理器核并且在处理器核上执行。在程序的执行期间,各个处理器核可以转换通过状态600的进展,使得一个核可以处于一个状态而另一核可以处于不同的状态。

[0106] 在状态605,可以取消映射相应处理器核的状态。未映射的处理器核是当前未指派用于执行指令块的实例的核。例如,在程序开始在基于块的计算机上的执行之前,可以取消映射处理器核。作为另一示例,在程序开始执行之后可以取消映射处理器核但不是使用所有核。具体地,根据程序的动态流程,至少部分执行程序的指令块。程序的某些部分通常可以串行或顺序流动,诸如当后面的指令块依赖于来自较早指令块的结果时。程序的其他部分可以具有更平行的流程,诸如当多个指令块可以同时执行而不使用并行执行的其他块的结果时。在程序的更多顺序流期间可以使用更少的核来执行程序,并且在程序的更多并行流期间可以使用更多核来执行程序。

[0107] 在状态610,可以映射相应处理器核的状态。映射的处理器核是当前被指派以执行指令块的实例的核。当指令块映射到特定处理器核时,指令块在飞行中。飞行中的指令块是针对基于块的处理器的特定核的块,并且该块将在特定处理器核上推测性地或非推测性地执行或正在执行。特别地,飞行中的指令块对应于被映射到状态610-650的处理器核的指令

块。当在块的映射期间已知程序将使用由执行指令块提供的工作时，块以非推测方式执行。当在映射期间不知道程序是否将使用由执行指令块提供的工作时，块推测性地执行。推测性地执行块可以潜在地提高性能，诸如当与将要在已知块的工作将被使用之后或之时开始块的情况相比，推测块更早地开始时。然而，推测性地执行潜在地增加在执行程序时使用的能量，诸如当程序不使用推测性工作时。

[0108] 基于块的处理器包括有限数目的同构或异构处理器核。典型的程序可以包括比可以装入处理器核的指令块更多的指令块。因此，程序的各个指令块通常将与程序的其他指令块共享处理器核。换言之，给定核可以在程序的执行期间执行若干不同指令块的指令。具有有限数目的处理器核还表示，当所有处理器核忙于执行指令块并且没有新核可以用于分派时，程序的执行可以停滞或延迟。当处理器核变为可用时，指令块的实例可以映射到处理器核。

[0109] 指令块调度器可以指派哪个指令块将在哪个处理器核上执行以及何时将执行指令块。映射可以基于多种因素，诸如用于执行的目标能量、处理器核的数目和配置、处理器核的当前和/或先前使用情况、程序的动态流程、是否启用推测执行、将执行推测性块的置信度、以及其他因素。可以将指令块的实例映射到当前可用的处理器核(诸如当当前没有指令块在其上执行时)。在一个实施例中，指令块的实例可以映射到当前正忙的处理器核(诸如当核正在执行指令块的不同实例时)，并且后面映射的实例可以在先前映射的实例完成时开始。

[0110] 在状态620，可以获取相应处理器核的状态。例如，处理器核的IF流水线级可以在获取状态期间是活动的。获取指令块可以包括将块从存储器(诸如L1高速缓存、L2高速缓存或主存储器)传送到处理器核，以及从处理器核的本地缓冲器读取指令以便可以解码指令。例如，指令块的指令可以加载到处理器核的指令高速缓存、缓冲器或寄存器中。可以在同一时钟周期期间并行(例如，同时)获取指令块的多个指令。获取状态可以是多个周期长，并且可以在处理器核被流水线化时与解码(630)和执行(640)状态重叠。

[0111] 当指令块的指令被加载到处理器核上时，指令块驻留在处理器核上。当加载指令块的一些但非全部指令时，指令块部分驻留。当加载指令块的所有指令时，指令块完全驻留。指令块将驻留在处理器核上，直到处理器核复位或不同的指令块被获取到处理器核上。特别地，当核处于状态620-670时，指令块驻留在处理器核中。

[0112] 在状态630，可以解码相应处理器核的状态。例如，处理器核的DC流水线级可以在获取状态期间是活动的。在解码状态期间，解码指令块的指令，以便它们可以存储在处理器核的指令窗口的存储器存储中。特别地，指令可以从相对紧凑的机器代码转换为可以用于控制处理器核的硬件资源的不太紧凑的表示。解码状态可以是多个周期长，并且当处理器核被流水线化时可以与获取(620)和执行(640)状态重叠。在解码指令块的指令之后，其可以在满足指令的所有依赖性时执行。

[0113] 在状态640，可以执行相应处理器核的状态。执行状态可以包括各种操作模式，诸如默认执行模式和调试模式。在执行状态的默认模式期间，执行指令块的指令。特别地，处理器核的EX和/或LS流水线级可以在执行状态期间是活动的。指令块可以推测性地或非推测性地执行。推测块可以执行以完成，或者可以在完成之前终止，诸如当确定将不使用由推测块执行的工作时。当指令块终止时，处理器可以转换到中止状态。例如，当确定将使用块

的工作，缓冲所有寄存器写入，缓冲对存储器的所有写入，以及计算分支目标时，推测块可以完成。例如，当所有寄存器写入被缓冲，对存储器的所有写入被缓冲，以及计算分支目标时，非推测性块可以执行以完成。执行状态可以是多个周期长，并且当处理器核被流水线化时可以与获取(620)和解码(630)状态重叠。当指令块完成时，处理器可以转换到提交状态。

[0114] 在执行状态的调试模式期间，指令块的指令可以是单步的或者一次一个地执行。例如，处理器核可以在执行状态的调试模式下停止，诸如当控制信号被断言时或者当指令头指定调试模式将用于指令块时。在接收到执行单步操作的指示时，可以执行指令块的一个指令。可以从处理器核扫描或读出处理器核的中间状态。在接收到执行单步操作的另一指示时，可以对指令块的下一指令重复该过程。“下一”指令可以基于编译器生成的顺序、调度器生成的顺序或在处理器核外部生成的顺序(诸如通过在不同核上运行的调试软件)来确定。块的指令可以继续单步执行，直到满足提交条件，然后处理器可以转换到提交状态。

[0115] 在状态650，可以将相应处理器核的状态设置为提交或中止。在提交期间，指令块的指令的工作可以以原子方式提交，以便其他块可以使用指令的工作。特别地，提交状态可以包括提交阶段，其中本地缓冲的架构状态被写入对其他处理器核可见或由其他处理器核可访问的架构状态。当更新可见架构状态时，可以发出提交信号并且可以释放处理器核，以便可以在处理器核上执行另一指令块。在中止状态期间，可以停止核的流水线以减少动态电力调度。在某些应用中，可以对核进行电源门控以降低静态电力调度。在提交/中止状态结束时，处理器核可以接收要在处理器核上执行的新指令块，可以刷新核，可以使核空闲，或者可以重置核。

[0116] 在所公开的技术的某些示例中，将指令块分成多个部分，并且使用不同的空间和/或时间分配的资源来执行各个部分。例如，处理器的执行分片可以适于处理指令块的32字部分。当一部分的所有指令都已经执行时，可以部分地提交由该相应部分产生的结果。在一些示例中，部分提交的结果实际上被写入架构寄存器文件，临时存储在物理寄存器文件中，和/或实际写入存储器。在其他示例中，临时存储部分提交的结果，直到确定是否将提交或中止整个指令块。

[0117] 在状态660，可以确定指令块是否具有要获取、解码和执行的附加部分。例如，如果指令块大小大于当前上下文的分配资源可以处理的最大指令数，则来自当前部分的结果存储在结果缓冲器中，并且处理器进行到状态620以便获取、解码和执行指令块的后续部分的指令。在一些示例中，可以通过将多个执行逻辑分片分配给上下文来同时获取、解码和执行这些部分。如果指令块大小等于或小于当前上下文的分配资源可以处理的最大指令数，或者如果当前指令块没有剩余部分要执行，则处理器进行到状态670。

[0118] 在状态670，相应处理器核的状态可以是空闲的。基于块的处理器的性能和功耗可以潜在地基于在给定时间活动的处理器核的数目来调节或折衷。例如，对同时运行的核执行推测性工作可以提高计算速度，但是如果推测误预测率很高则增加功率。作为另一示例，在提交或中止先前执行的指令块之后立即向处理器分配新指令块可以增加并发执行的处理器的数目，但是可以减少重用驻留在处理器核上的指令块的机会。当保持空闲处理器核的高速缓存或池时，可以增加重用。例如，当处理器核提交常用指令块时，处理器核可以放置在空闲池中，以便下次要执行相同的指令块时可以刷新核。如上所述，刷新处理器核可以节省用于获取和解码驻留指令块的时间和能量。可以基于由编译器执行的静态分析或由指

令块调度器执行的动态分析来确定放置在空闲高速缓存中的指令块/处理器核。例如，指示可能重用指令块的编译器提示可以放在块的头中，并且指令块调度器可以使用该提示来确定在提交指令之后块是否使块空闲或将块重新分配给不同的指令块。例如，当空闲时，处理器核可以被置于低功率状态以减少动态功耗。

[0119] 在状态680，可以确定是否可以刷新驻留在空闲处理器核上的指令块。如果要刷新核，则可以断言块刷新信号，并且核可以转换到执行状态(640)。如果不打算刷新核，则可以断言块重置信号，并且核可以转换到未映射状态(605)。当核被重置时，核可以被放入具有其他未映射核的池中，以便指令块调度器可以向核分配新的指令块。

[0120] IX.示例处理器微架构

[0121] 图7是概述其中可以实现所公开的技术的某些方面的示例处理器微架构的框图700。例如，可以使用类似于框图700所示的架构来实现基于块的处理器ISA，包括EDGE ISA。所描绘的微架构可以以任何合适的技术实现，包括作为集成电路、SOC或具有可编程逻辑，诸如在FPGA中。

[0122] 所示出的示例微架构包括架构寄存器文件710，架构寄存器文件710包含存储架构状态的寄存器，该架构状态可以被传递去往和来自不同的指令块。架构状态寄存器由处理器的指令集架构定义。因为寄存器文件在架构上对程序员可见，所以每个同时执行的上下文(例如，线程或进程)被分配其自己的寄存器文件或其自己的架构寄存器文件710的部分。在一些示例中，这可以通过以下方式实现：为每个上下文提供单独的寄存器文件，或者使用足够大以容纳每个活动上下文但具有适当分区的单个寄存器文件。可以使用任何合适的存储技术来实现架构寄存器文件710，包括使用由触发器和锁存器、静态RAM(SRAM)或其他合适的存储器技术形成的存储元件。架构寄存器文件710可以由物理寄存器文件的一部分或全部形成，物理寄存器文件是由存储元件形成的寄存器阵列，其不必限于处理器ISA。例如，可以对物理寄存器文件进行分区以形成多个架构寄存器文件，每个架构寄存器文件与特定上下文相关联。在一些示例中，物理寄存器可以被配置为存储重命名的寄存器值，或者存储用于推测性地执行的线程或进程的寄存器值。物理寄存器文件可以由类似的存储元件形成，包括触发器和锁存器、静态RAM(SRAM)或其他合适的存储器技术。在某些实现中，这样的寄存器文件还可以具有多个输入端口和/或多个读取端口。

[0123] 示例微架构还包括一级(L1)指令高速缓存720和L1数据高速缓存730。在一些示例或配置中，指令高速缓存720和数据高速缓存730可以跨所有活动上下文共享。在其他示例中，指令高速缓存720和数据高速缓存730中的一个或两个被分区为多个单独的存储体。例如，可以对数据高速缓存进行分区，使得每个上下文存在一个存储体，并且高速缓存的每个分区对于每个相应的上下文是私有的。在上下文接收私有指令和/或数据高速缓存分区的示例中，使用附加控制逻辑来保持高速缓存一致性。

[0124] 图7中描绘的示例微架构还包括解码器逻辑745、加载存储队列750、多个功能单元分片761、762、763和764的集合760。每个功能分片761-764包含用于执行与指令(例如，EDGE指令块中的指令)相关联的操作的执行逻辑。可以配置示例微架构处理器，使得分片被分配给一个、两个、三个或四个不同的上下文。例如，所有功能分片761-764可以被分配用于由处理器执行单个上下文。稍后可以重新配置处理器，使得第一功能分片761用于执行第一上下文，第二功能分片762用于执行第二上下文，等等。功能分片761-764在空间上被分配，因为

功能分片的资源以预定方式被分配给特定上下文。因此，与当例如功能分片的资源停滞时上下文共享功能资源的使用的其他技术不同，在所示实施例中，空间分配的资源在预定时段专用于上下文。

[0125] 每个功能分片包括类似的执行资源，包括例如用于指令分片761：指令窗口分片770、第一操作数缓冲器771和第二操作数缓冲器772；以及一个或多个执行单元，包括例如执行单元773或774。所示的执行单元可以包括例如整数算术和逻辑单元(ALU)、加法器、减法器、乘法器、除法器、移位器、旋转器或其他合适的执行单元。在所示示例中，包括浮点单元(FPU)775，并且FPU 775可以由任何所示功能分片761-764执行的任何上下文共享。在其他示例中，FPU和其他专用逻辑(诸如DSP或图形功能)可以被共享，或者可以被包括在分片内并且因此不在当前指派给一个或多个分片的上下文之外共享。

[0126] 指令窗口分片(例如，指令窗口分片770)存储诸如解码的指令、用于指令块的当前执行状态的状态信息、以及用于控制上下文在一个或多个指令分片上的执行的控制逻辑等信息。在一些示例中，功能分片可以每个时钟周期执行多于一个指令。指令窗口分片770还可以包括用于提供控制推测的逻辑。操作数缓冲器临时存储为映射到功能分片的指令块内的指令生成和由其消耗的操作数。例如，在EDGE ISA中，由指令生成的数据操作数可以在被一个功能单元使用之前临时存储在操作数缓冲器之一中，并且结果被发送到指令目标。

[0127] 所示的微架构还包括控制数据从操作数缓冲器到功能单元的流动的仲裁器电路780。例如，仲裁器电路可以基于功能执行单元消耗数据的可用性引起在分片内或分片之间暂停执行指令。仲裁器电路780还可以控制对加载存储队列750和寄存器文件710的访问，因为输出是从相应的指令块生成的。例如，微架构可以在每个时钟周期仅支持对加载存储队列750的一次或两次写入，因此仲裁器逻辑将选择允许哪些分片在给定时钟周期上访问加载存储队列750。

[0128] 所示的微架构还包括下一块预测器电路790。下一块预测器可以用于为指令块内的分支指令提供分支预测。下一块预测器790可以接受以下中的一个或多个作为输入：目标块地址、是否采用分支、源块地址或执行上下文的标识符(例如，线程或进程ID)。在一些示例中，为每个功能分片分配单独的下一块预测器。在其他示例中，下一块预测器在功能分片之间在时间上共享，如将在以下示例中进一步详细讨论的。

[0129] X.处理器资源的示例分配

[0130] 图8A-8C是示出可以在所公开的技术的某些示例中执行的在基于块的处理器中的资源到上下文的示例分配的图。例如，可以使用所示的微架构来实现EDGE ISA处理器。

[0131] 图8A是示出第一模式下的处理器的配置的框图800。当处理器处于该第一模式时，所有功能分片都可用于单个上下文。所示处理器包括存储从耦合到处理器的存储器获取的指令块的指令的指令高速缓存720。处理器还包括物理寄存器文件810、L1数据高速缓存730、加载存储队列740和一组执行分片760。处理器还包括下一块预测器790。在该第一模式下，指令高速缓存720、物理寄存器文件810、执行资源760、加载存储队列740、数据高速缓存730和下一块预测器790内的所有资源可用于执行单个上下文。在所示的示例中，可以划分处理器的资源，使得存在四个执行分片。

[0132] 在所公开的技术的一个示例中，每个分片包括可以存储多达128个指令的私有指令窗口、以及足够的操作数缓冲器和完全大小的私有加载存储队列。假定最大块可以包括

128个指令，该示例实现128指令窗口(其可以保持单个足够大的块或几个较小的块)。在这样的配置中，将处理器划分为分片减少了指派给分片的执行单元，但是不减少与操作数缓冲器、加载存储队列和用于支持上下文的其他资源相关联的开销。因此，尽管私有指令窗口可以容纳整个128指令块，但是处理器不包括用于完全利用这样大的指令窗口的资源。因此，在很多情况下，处理器过度供应并且其资源可能未得到充分利用。

[0133] 在所公开的技术的其他示例中，处理器可以将执行资源760逻辑地划分为“分片”，并且选择并行地在分片内和跨分片执行指令。例如，它可以将128指令窗口分成四个分片，并且每个周期从每个分片执行最多两个指令(总共最多8个)。图8B是示出这样的替代资源优化配置的处理器的框图810。

[0134] 在图8B的配置中，可以为每个分片指派32指令的指令窗口(例如，128指令的指令窗口的四分之一)和为“完全资源模型”提供的加载存储队列资源的一部分。当遇到包含多于32个指令的指令块时，可以通过获取指令块的子块来将指令块“折叠”到该窗口中。例如，前32个指令被获取到32指令分片的指令窗口中。在这32个指令之外发送数据的指令必须保持其结果，直到下一指令到达窗口并且作为指令的输入操作数被提供。可以进一步修改处理器以包括结果缓冲器，该结果缓冲器保持这样的中间结果，直到它们可以被当前指令窗口之外的后续指令消耗。

[0135] 例如，如图8B所示的阴影所示，第一上下文已经在空间上分配了指令高速缓存720的一部分821(“L1指令高速缓存0”)、物理寄存器文件810的一部分811(“架构寄存器文件0”)、执行分片761(包括结果缓冲器861)、加载存储队列740的一部分841(“加载存储队列0”)、以及L1数据高速缓存730的一部分831(“L1数据高速缓存0”)。第二上下文在空间上被分配了不同的较大资源部分，包括指令高速缓存720的第二部分822(“L1指令高速缓存1”、“L1指令高速缓存2”)、物理寄存器文件810的第二部分812(“架构寄存器文件1”、“架构寄存器文件2”)、两个执行分片762和763(包括结果缓冲器862和863)、加载存储队列740的第二部分842(“加载存储队列1”、“加载存储队列2”)和L1数据高速缓存730的第二部分832(“L1数据高速缓存1”、“L1数据高速缓存1”)。空间分配的部分是私有分配的，因此上下文不能访问非分配的资源。因此，例如，第一上下文和第二上下文中的每个都可以访问专用但较小的指令高速缓存、数据高速缓存。切换逻辑(诸如多路复用器、交叉开关矩阵、可重新配置逻辑或其他合适的技术)用于实现将处理器重新配置到所分配的空间部分中。应当注意，空间分配的资源不像在通用处理器的多线程实现中那样在上下文之间动态地重新分配。因此，为每个上下文提供对其空间分配的资源的独立访问。在一些示例中，可以将资源的各方面分配给上下文。例如，多端口数据高速缓存或寄存器文件可以将多个读或写入端口分配给上下文。

[0136] 执行分片761-764中的每个与相应的结果缓冲器861-864相关联，如下面将进一步详细讨论的。结果缓冲器可以用于临时存储要在指令块的各部分之间发送的操作数。

[0137] 图8C是示出这种替代资源优化配置的处理器的框图820。在图8C的配置中，L1数据高速缓存730和下一块预测器790在时间上被分配给上面讨论的第一上下文和第二上下文。例如，多路复用和解多路复用逻辑可以被配置为允许在偶数时钟周期上访问第一上下文并且在奇数时钟周期上访问第二上下文。因此，第一上下文可以在偶数时钟周期期间将数据读取或写入L1数据高速缓存730并且从预测器790接收下一块预测，并且第二上下文在奇数

时钟周期期间被提供类似访问。应当注意，可以使用更细粒度和更复杂的方案来在时间上分配资源。例如，可以为第二上下文分配对时间分配的资源的访问，其时钟周期是第一上下文的两倍。应当进一步理解，处理器可以具有在彼此组合子组合中提供的空间和时间分配的资源。例如，如图8C所示，第一上下文在空间上被分配指令高速缓存720的四分之一、物理寄存器文件810、一个执行分片760和一个加载存储队列740，并且还在在时间上被分配对L1数据高速缓存730和下一块预测器790的访问。第二上下文在空间上被分配相应资源的一半，并且还在在时间上被分配对数据高速缓存和下一块预测器的访问。

[0138] XI.使用结果缓冲器的示例块折叠

[0139] 图9是概述上面在图7和图8A-8C中详述的微架构内的处理器分片的示例的框图900，包括可以在所公开的技术的某些示例中使用的结果缓冲器(例如，缓冲器861-864)的进一步细节。

[0140] 如图9所示，结果缓冲器861的大小已经调节为包括192个条目。该大小足以容纳96个指令的两个输入操作数。对于映射到32个指令窗口的128指令的指令块，这是需要保存的最大输入操作数，因为128指令的指令块的前四分之三的操作数可以存储在结果缓冲器中，并且然后由指令块的最后四分之一消耗。例如，如上面在图4的示例中讨论的，指令460(I[3])的结果存储在结果缓冲器861中，直到它被指令465消耗(I[100])。在折叠指令块的各部分之间共享谓词的示例中，可以使用结果缓冲器861中的附加条目。

[0141] 在图9的示例中，结果缓冲器可以被组织为表(例如，具有RAM)，其中一个结果块条目专用于指令编号，并且当目标指令到达并且准备好执行时提供结果。在一些示例中，存储在结果缓冲器中的数据包括接收存储在结果缓冲器中的数据的目标指令的一个或多个标识符。在一些示例中，结果缓冲器被组织为FIFO，并且将值与其目标指令标识符一起存储，其然后被发送到后续指令块中的相关指令。还应当注意，对于将其他输入操作数发送到指令的指令集架构，可以对应地扩展结果缓冲器。例如，在包括谓词指令的基于块的处理器ISA中，由指令生成的谓词可以临时存储在结果缓冲器中，直到由稍后的指令在稍后的时间点消耗。所示结果缓冲器861可以存储随后被反馈到相同功能分片761的值，而在其他示例中，结果缓冲器861存储由被映射到指令块的后续部分的其他执行分片消耗的值。

[0142] 结果缓冲器861可以使用任何合适的技术来实现，包括用触发器或锁存器、SRAM或其他合适的存储器技术实现的寄存器。例如，使用指令标识符作为输入地址的RAM可以存储用于存储器中该地址位置处的指令的目标操作数的数据。然后，输入操作数可以由目标指令在稍后的时间点访问。

[0143] 使用很多不同的技术可以减小结果缓冲器的大小。例如，一个32指令分片可以生成最多32个结果，这可以形成用于最多64个操作数的目标(在所描绘的示例中，每个指令可以将其结果发送到最多两个目标指令)。因此，前两个32指令子块可以为指令块内的随后两个子块生成操作数。在这样的示例中，结果缓冲器的大小可以足够大以便为剩余块存储多达128个结果。在这种设计中，为每个条目保持目标指令ID，以便解决目标指令中的任何歧义。在一些示例中，用于生成指令块的编译器可以通过最小化在指令块部分之间传递的操作数的数目来帮助减小结果缓冲器大小。例如，编译器可以强制限制例如可以在各部分之间传递32个操作数。在一些示例中，编译器可以在部分之间移动指令以便满足该限制。在一些示例中，编译器可以在两个不同部分中复制指令以避免达到跨越部分边界的操作数限

制。

[0144] 当指令子块完成并且另一指令子块到达时，存储在结果缓冲器中的结果可以被复制到指令的操作数缓冲器中。如受益于本公开的本领域普通技术人员将容易理解的，可以采用设计和优化结果缓冲器的设计的替代方法。

[0145] 加载存储队列740可以以多种不同方式布置。例如，加载存储队列不必是32条目队列，但可以限制为八个条目的大小。在这样的示例中，在指令块的每个子块完成之后，队列中的条目被提交到存储器并且可以被释放以供指令块内的后续子块使用。在这样的示例中，不是在整个指令块完成之后提交整个指令块，而是在其执行时部分地提交指令块。

[0146] 因此，如上所述，所公开的技术可以改进处理器(包括BB-ISA处理器)中的资源利用率。这种空间和时间分配技术可以为每个上下文提供有保证的每个上下文吞吐量和前进进度，即使其他上下文无法取得进展也是如此。因此，在某些实施例中，可以支持多个上下文，同时降低控制逻辑复杂性和资源利用不足。此外，在某些实施例中，可以增强处理器和系统安全性，因为执行资源被分配给特定上下文，但是在分配给特定上下文时，所分配的资源不与其他上下文共享。因此，可以防止恶意代码访问由处理器执行的其他上下文。

[0147] XII.处理器资源的空间和时间分配的示例

[0148] 图10是示出了可以在所公开的技术的某些示例中执行的处理器资源的空间和时间分配的示例的示图1000。时间轴1010用于指示在根据所公开的技术配置的基于块的处理器中的三个上下文(上下文0、上下文1和上下文2)之间的资源分配。

[0149] 如图10所示，在三个上下文之间在空间上分配1020一组六个执行分片。上下文0在空间上被分配三个执行分片，标记为EX.0、EX.1、和EX.2。上下文1被分配给两个执行分片，标记为EX.3和EX.4。上下文2被分配单个执行分片，标记为EX.5。如图10所示，在整个所示时间段内，每个上下文在空间上被分配给各个执行分片。因此，各个上下文不会争夺空间分配的资源。

[0150] 图10中还示出了三个上下文之间的资源的时间分配1030。标记为DC 0、DC 1的两个数据高速缓存被示出为在每个时钟周期的基础上在三个上下文之间在时间上被分配。因此，上下文0可以每第三个时钟周期访问DC 0和DC 1，而上下文1在随后的时钟周期访问DC 0和DC 1，等等。也可以为多个时钟周期分配资源。例如，标记为IF/DX的指令获取/指令解码器单元在前两个时钟周期被分配给上下文0，在第二两个时钟周期被分配给上下文1，并且在第三组两个时钟周期被分配给上下文2。还示出了标记为NBP的下一块预测器的分配。下一块预测器NBP在上下文之间不是在时间上被平均分配的。如图所示，上下文0在三个时钟周期内接收对下一块预测器NBP的访问，而另外两个上下文(1和2)在单个时钟周期内接收对NBP的访问。例如，如果上下文的工作流程使用更多分支，具有更小的指令块，或者需要比其他上下文更频繁地使用分支预测，则可以执行这样的分配。然而，应当注意，所示的时间分配对于每个上下文是固定的，因此，每个上下文保证至少一部分时间分配的资源。

[0151] XIII.处理器资源的空间和时间分配的示例方法

[0152] 图11是概述可以在所公开的技术的某些示例中执行的分配资源和执行指令块的示例的流程图1100。

[0153] 在处理框1110，根据预定分配将资源在空间和/或时间上分配给一个或多个执行上下文。例如，空间分配的资源可以包括指令高速缓存、指令窗口、功能单元、数据高速缓存

或寄存器文件的部分。不需要均匀地分配空间分配的资源，但是可以为一些上下文分配比其他上下文更多的资源，包括更多的执行分片。时间分配的资源可以包括分支预测器、数据高速缓存或其他合适的资源。应当注意，分配是预先确定的，因为资源在空间和/或时间上分配的方式不依赖于当前处理器负载或状态。因此，可以向每个上下文保证用于执行给定上下文的一定量的资源。

[0154] 在处理框1120，利用所分配的资源执行用于执行上下文的指令块。例如，分配给给定上下文的指令窗口可以在给定上下文操作时跟踪操作数和其他处理器状态。此外，用于上下文的指令块可以根据例如每时钟周期分配来访问时间分配的资源。

[0155] XIV.使用结果缓冲器来分配资源的示例方法

[0156] 图12是概述可以在所公开的技术的某些示例中执行的使用在空间和/或时间上分配的资源(包括使用结果缓冲器)来执行指令块的示例的流程图1200。

[0157] 在处理框1210，根据预定分配将资源在空间上分配给一个或多个执行上下文。

[0158] 在处理框1220，根据预定分配将资源在时间上分配给一个或多个执行上下文。因此，应当注意，处理器的某些资源可以在空间上分配，而其他资源在在时间上分配给上下文。在一些示例中，处理器可以支持相同资源的空间或时间分配。例如，数据高速缓存可以可配置为在空间上被分配，并且然后重新配置以便在时间上分配给在处理器上执行的上下文。

[0159] 在处理框1230，将从指令块的第一部分生成的结果临时存储在结果缓冲器中。例如，128指令的指令块可以分成4个32指令部分。可以临时存储来自指令块的第一部分的结果，直到它们被指令块的另一部分中的指令消耗。

[0160] 在处理框1240，利用来自结果缓冲器的指令块的第二部分从结果缓冲器中消耗数据。例如，指令块的后续部分可以访问结果缓冲器以接收该第二部分内的特定指令的指令操作数。取决于处理器配置，可以使用与当前指令块相同的功能分片或不同的功能分片来执行后续指令块。

[0161] 在处理框1250，完成给定上下文的执行，并且可以将其在空间和/或时间上分配的资源重新分配给其他上下文。

[0162] XV.计算环境示例

[0163] 图13示出了其中可以实现所描述的实施例、技术和工艺(包括分配用于执行基于块的处理器的指令块的资源)的合适的计算环境1300的一般化示例。

[0164] 计算环境1300不旨在对技术的使用范围或功能提出任何限制，因为该技术可以在不同的通用或专用计算环境中实现。例如，所公开的技术可以用其他计算机系统配置来实现，包括手持设备、多处理器系统、可编程消费电子器件、网络PC、小型计算机、大型计算机等。所公开的技术还可以在分布式计算环境中实现，其中任务由通过通信网络链接的远程处理设备执行。在分布式计算环境中，程序模块(包括用于基于块的指令块的可执行指令)可以位于本地和远程存储器存储设备中。

[0165] 参考图13，计算环境1300包括至少一个基于块的处理单元1310和存储器1320。如图13所示，该最基本配置1330被包括在虚线内。基于块的处理单元1310执行计算机可执行指令，并且可以是真实或虚拟处理器。在多处理系统中，多个处理单元执行计算机可执行指令以增加处理能力，因此，多个处理器可以同时运行。存储器1320可以是易失性存储器(例

如,寄存器、高速缓存、RAM)、非易失性存储器(例如,ROM、EEPROM、闪存等)、或两者的某种组合。存储器1320存储可以例如与本文中描述的技术的实现一起使用的软件1380、图像和视频。计算环境可以具有附加特征。例如,计算环境1300包括存储装置1340、一个或多个输入设备1350、一个或多个输出设备1360、以及一个或多个通信连接1370。互连机制(未示出)(诸如总线、控制器或网络)互连计算环境1300的组件。通常,操作系统软件(未示出)为在计算环境1300中执行的其他软件提供操作环境,并且协调计算环境1300的组件的活动。

[0166] 存储装置1340可以是可移动的或不可移动的,并且包括磁盘、磁带或盒式磁带、CD-ROM、CD-RW、DVD、或者可以用于存储信息并且可以在计算环境1300内被访问的任何其他介质。存储装置1340存储可以用于实现本文中描述的技术的软件1380、插件数据和消息的指令。

[0167] 输入设备1350可以是向计算环境1300提供输入的触摸输入设备,诸如键盘、小键盘、鼠标、触摸屏显示器、笔或轨迹球、语音输入设备、扫描设备或其他设备。对于音频,输入设备1350可以是接受模拟或数字形式的音频输入的声卡或类似设备,或者是向计算环境1300提供音频样本的CD-ROM读取器。输出设备1360可以是显示器、打印机、扬声器、CD刻录机、或提供来自计算环境1300的输出的另一设备。

[0168] 通信连接1370使得能够通过通信介质(例如,连接网络)与另一计算实体通信。通信介质以调制数据信号传送诸如计算机可执行指令、压缩图形信息、视频或其他数据等信息。通信连接1370不限于有线连接(例如,兆位或千兆以太网、Infiniband、光纤通道或光纤通道连接),还包括无线技术(例如,经由Bluetooth、WiFi(IEEE802.11a/b/n)、WiMax、蜂窝、卫星、激光、红外线的RF连接)以及用于为所公开的代理、网桥和代理数据消费者提供网络连接的其他合适的通信连接。在虚拟主机环境中,通信连接可以是由虚拟主机提供的虚拟化网络连接。

[0169] 所公开方法的一些实施例可以使用在计算云1390中实现所公开的技术的全部或一部分的计算机可执行指令来执行。例如,所公开的编译器和/或基于块的处理器服务器位于计算环境1330中,或者所公开的编译器可以在位于计算云1390中的服务器上执行。在一些示例中,所公开的编译器在传统的中央处理单元(例如,RISC或CISC处理器)上执行。

[0170] 计算机可读介质是可以在计算环境1300内访问的任何可用介质。作为示例而非限制,利用计算环境1300,计算机可读介质包括存储器1320和/或存储装置1340。应当容易理解,术语计算机可读存储介质包括用于数据存储的介质,诸如存储器1320和存储装置1340,而不包括诸如调制数据信号等传输介质。

[0171] XVI.所公开的技术的其他示例

[0172] 本文中根据上面讨论的示例讨论了所公开的主题的其他示例。

[0173] 在所公开的技术的一些示例中,一种处理器,可配置为根据两种或更多种模式在空间上分配一组执行资源。该模式包括第一模式,在第一模式下,处理器被配置为将该组执行资源的第一部分在空间上分配给第一上下文并且将该组执行资源的不同的第二部分在空间上分配给第二上下文,相应的第一部分和第二部分在上下文处于活动状态时不在第一上下文与第二上下文之间共享。处理器可以是CISC、RISC、基于块或EDGE ISA处理器。处理器可以以任何合适的制造技术实现,包括作为定制集成电路、ASIC、SoC、FPGA或其他合适的逻辑实现技术。

[0174] 在一些示例中，处理器包括第二模式，在第二模式下，处理器被配置为将该组执行资源全部在空间上重新分配给单个上下文。在一些示例中，处理器包括第三模式，在第三模式下，处理器被配置为将该组执行资源的一部分但不是全部在空间上重新分配给单个上下文，并且不分配该组执行资源的剩余部分。在一些示例中，在上下文之间均等地分配执行资源。在一些示例中，在上下文之间不均等地分配执行资源。上下文包括描述由处理器托管的进程或线程的状态的架构状态数据。

[0175] 与进程或线程的个体上下文相关联的架构状态数据可以包括相关联的进程或线程的优先级信息、相关联的进程或线程的调度信息、子/父进程信息、进程间通信数据、权限信息、状态信息、进程或线程的标识符、架构寄存器文件值、指令块地址指针、指令块内的各个指令指针、存储器管理信息、或与进程或线程相关联的其他合适的信息。

[0176] 在一些示例中，所分配的一组执行资源至少包括以下中的一项或多项：指令窗口、功能单元、数据高速缓存、指令高速缓存、结果缓冲器、加载/存储队列或物理寄存器文件。在一些示例中，首先设置在空间被分配的该组执行资源，并且存在处理器的在时间上被分配给第一上下文和第二上下文的第二组执行资源。在一些示例中，在时间上被分配的第二组执行资源至少包括以下中的一项或多项：分支预测器、指令高速缓存、加载/存储队列和/或数据高速缓存。在一些示例中，在空间基础上分配一个或多个执行资源，并且在时间基础上对空间分配的执行资源进行子分配。

[0177] 在一些示例中，该组执行资源的第一部分包括结果缓冲器，结果缓冲器被配置为从执行资源的第一部分的第一分片向执行资源的第一部分的第二分片传送用于原子指令块的操作数。

[0178] 在所公开的技术的一些示例中，根据任何所公开的ISA的处理器包括被配置为执行两个或更多个执行上下文的处理器指令的一组执行资源，控制单元被配置为：在两个或更多个执行上下文之间在空间上分配该组资源的第一部分，并且在两个或更多个执行上下文之间在时间上分配该组执行资源的不同的第二部分。在执行两个或更多个执行上下文中的至少一个执行上下文的执行之前确定所分配的资源的第一部分和第二部分。例如，当线程或进程启动时，资源可以在空间上，在时间上或在空间上并且在时间上分配给正在启动的线程或进程。

[0179] 在一些示例中，该组执行资源至少包括以下中的一项或多项：分支预测器、数据高速缓存、指令高速缓存或指令解码器。

[0180] 在一些示例中，该组执行资源在时钟周期的基础上被分配给执行上下文中的相应执行上下文。例如，可以分配n个时钟周期中的每个用于访问时间分配的资源。在一些示例中，时间分配在上下文之间均匀分布，而在其他示例中，分配不均匀。在一些示例中，基于对该组资源的预测需求，将该组执行资源(在空间和/或时间上)分配给相应的执行上下文。在一些示例中，基于编译器在指令块中编码的信息，将该组执行资源(在空间和/或时间上)分配给相应执行上下文。在一些示例中，分析器生成指示预计资源需求的数据，并且分配至少部分基于分析器生成的数据。

[0181] 在所公开的技术的一些示例中，处理器是基于块的处理器，其被配置为根据本文中公开的任何示例在一个或多个上下文之间在空间上，在时间上或在空间上并且在时间上分配各种执行资源。该组执行资源包括时间分配的分支预测器，该分支预测器基于以下输

入中的一个或多个生成下一指令块地址的预测:目标块地址、是否采用了指令块中的一个或多个分支指令、源指令块地址和/或执行上下文的标识符。

[0182] 在所公开的技术的一些示例中,处理器是基于块的处理器,其被配置为根据本文中公开的任何示例在一个或多个上下文之间在空间上,在时间上或在空间上并且在时间上分配各种执行资源。每个执行上下文包括用于包括两个或更多个处理器指令的原子块的线程或进程的架构状态数据。

[0183] 在所公开的技术的一些示例中,一种操作基于块的处理器的方法包括:根据预定分配将基于块的处理器的执行资源分配给处理器的一个或多个执行上下文,并且利用所分配的执行资源执行用于执行上下文的指令块。各个资源的分配可以在空间和/或时间资源上。例如,根据本文中公开的示例性方法的空间分配的资源可以包括以下中的一个或多个:指令窗口、功能单元、数据高速缓存、指令高速缓存、结果缓冲器、加载/存储队列和/或物理寄存器文件。作为另一示例,时间分配的资源可以包括以下中的一个或多个:分支预测器、指令高速缓存,加载/存储队列或数据高速缓存。在一些示例中,在空间基础上分配一个或多个执行资源,并且在时间基础上二次分配空间分配的执行资源。

[0184] 在一些示例中,该方法使用预定分配,该预定分配是执行资源到每个执行上下文的空间指派。在一些示例中,预定分配是执行资源到每个执行上下文的时间指派。在一些示例中,该方法还包括重新分配执行资源,其中直到上下文完成、暂停或停止执行才重新分配执行资源。例如,在与上下文相关联的线程或进程被终止、暂停或处于等待或休眠状态之后,可以将分配给上下文的执行资源重新分配给不同的上下文(例如,新的或同时执行的进程或线程)用于执行。在一些示例中,其他分配的执行资源的功能分片和部分的数目保持相同,而在其他示例中,针对新上下文增加或减少功能资源的量。

[0185] 在一些示例中,通过该方法提供给上下文的预定分配不基于执行上下文的运行时停滞。预定分配可以基于预定的资源指派,通过指令块或由编译器生成的其他数据中的提示或规范,通过由运行时分析器生成的提示或数据,或者通过其他合适的技术来进行。

[0186] 在所公开的技术的一些示例中,选择预定分配以将执行资源的第一部分在空间上指派给每个执行上下文,并且选择预定分配以将执行资源的不相交的第二部分在时间上指派给每个执行上下文。

[0187] 在一些示例中,该方法还包括对于第一执行上下文:将由第一执行上下文的执行资源的第一部分生成的数据临时存储在结果缓冲器中,并且将临时存储的数据从结果缓冲器发送到结果缓冲器中的第一执行上下文的执行资源的第二部分。

[0188] 一种或多种计算机可读存储介质可以存储计算机可读指令,这些计算机可读指令在由计算机执行时引起计算机执行本文中公开的时间和/或空间分配的任何方法。基于块的处理器可以被配置为执行由该方法生成的计算机可读指令。

[0189] 在所公开的技术的一些示例中,编译器和/或分析器用于至少部分确定资源在空间上和/或时间上如何分配。例如,编译器或分析器可以确定用于特定上下文的指令块可能需要更多或更少的执行资源,并且生成指示如何在空间和/或时间上分配处理器的执行资源的数据。在一些示例中,编译器可以在指令块内对各个指令进行排序以允许减少指令块的各部分之间的操作数传递。当使用结果缓冲器临时存储在折叠指令块的各部分之间传递的数据操作数和/或谓词操作数时,这可以减少所需要的开销或者满足资源约束。例如,通

过将源指令或目标指令移动到位于接收或生成相应指令操作数数据的部分内，可以使用结果缓冲器中较少的存储元件。

[0190] 鉴于可以应用所公开的主题的原理的很多可能的实施例，应当认识到，所示的实施例仅是优选示例，不应当被视为将权利要求的范围限制于那些优选示例。而是，所要求保护的主题的范围由所附权利要求限定。因此，我们将所有属于这些权利要求及其同等权利要求范围内的内容视为我们的发明。

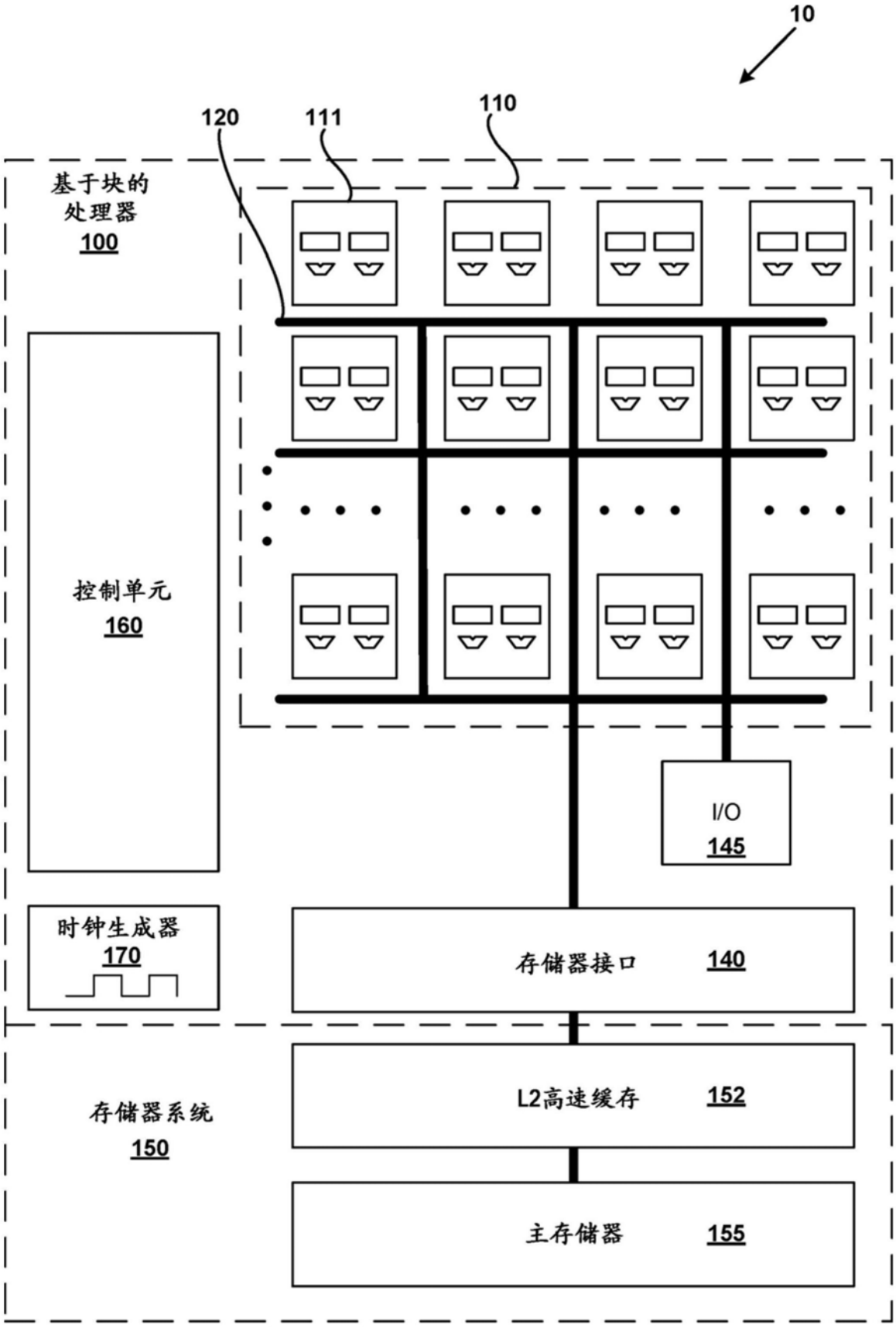

10
120
111
110
基于块的
处理器
100
控制单元
160
I/O
145
时钟生成器
170
存储器接口 140
存储器系统
150
L2高速缓存 152
主存储器 155

图1

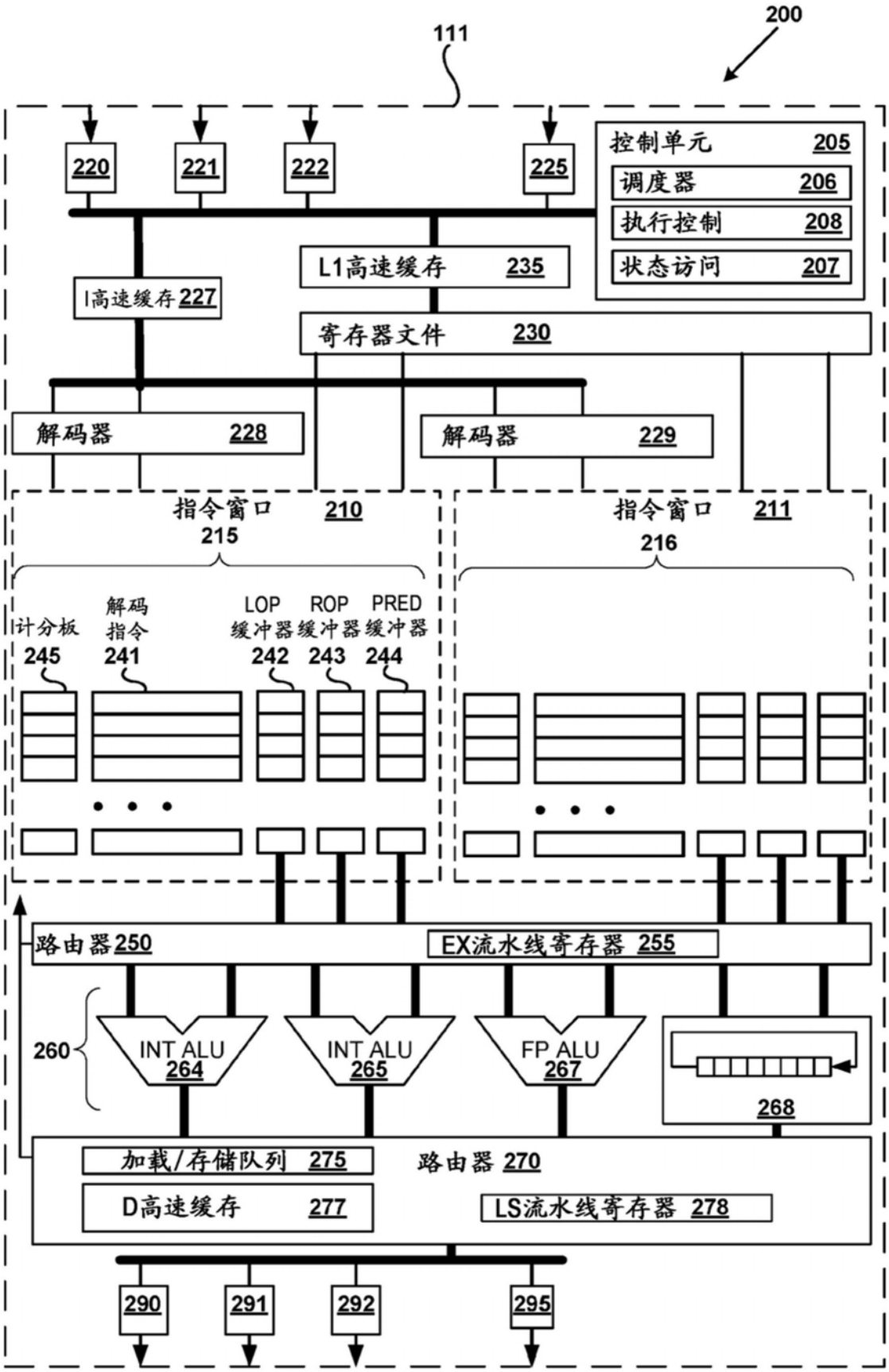

200
111
220
221
222
225
控制单元 205
调度器 206
执行控制 208
状态访问 207
L1高速缓存 235
I高速缓存 227
寄存器文件 230
解码器 228
解码器 229
指令窗口 210
215
指令窗口 211
216
计分板 245
解码指令 241
LOP缓冲器 242
ROP缓冲器 243
PRED缓冲器 244
路由器 250
EX流水线寄存器 255
260
INT ALU 264
INT ALU 265
FP ALU 267
268
加载/存储队列 275
路由器 270
D高速缓存 277
LS流水线寄存器 278
290
291
292
295

图2

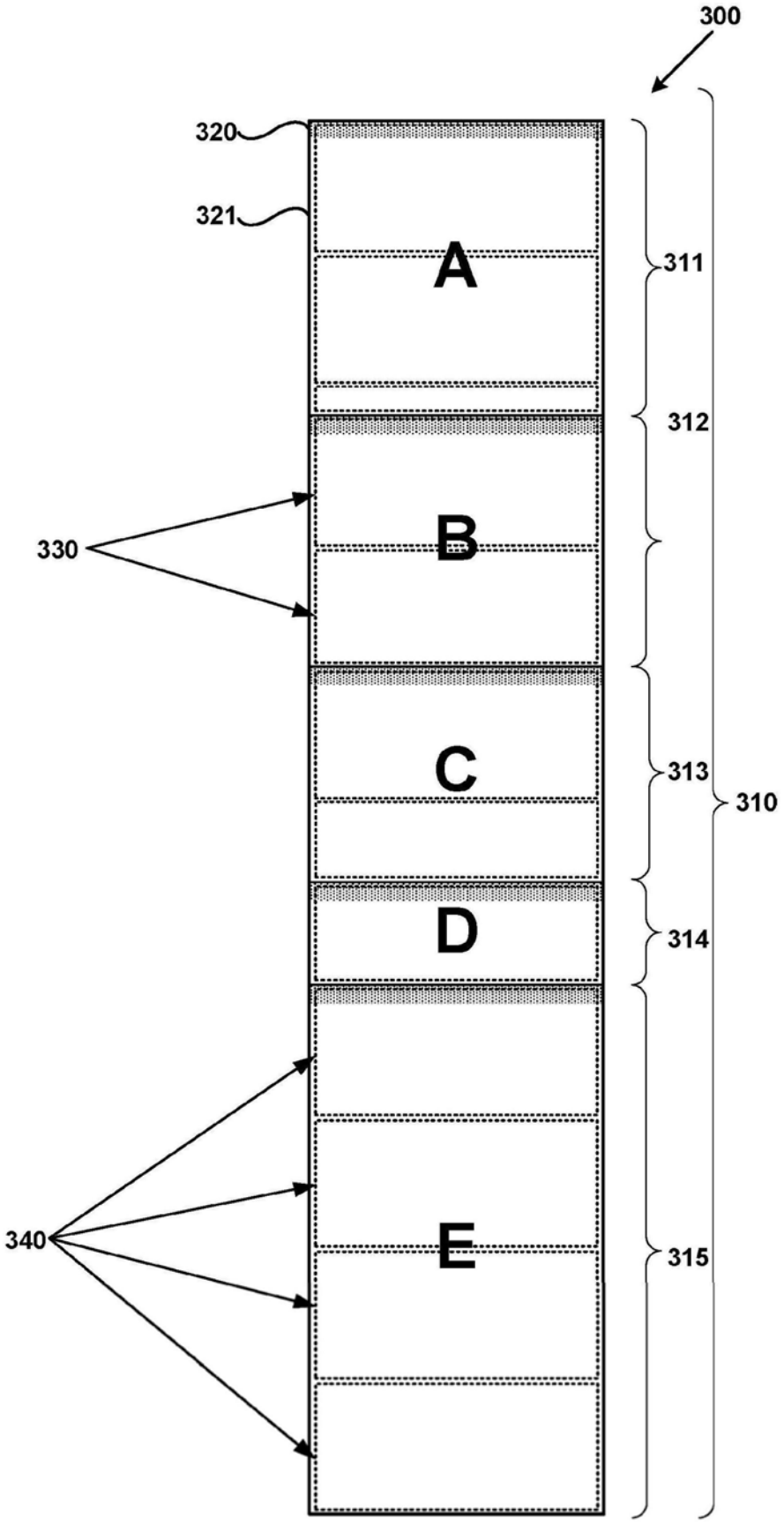


图3

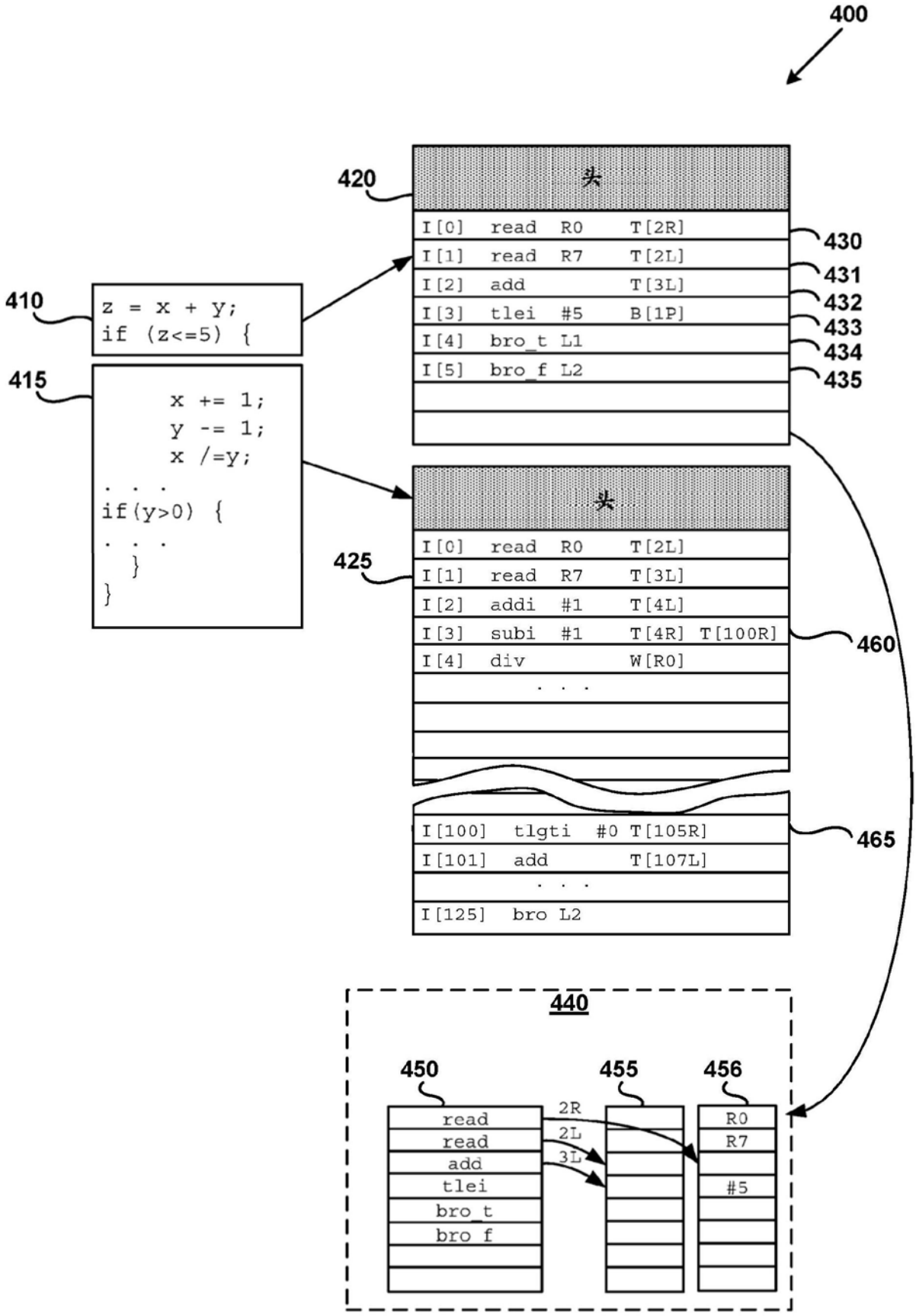

400
420
头
I[0] read R0 T[2R]
I[1] read R7 T[2L]
I[2] add T[3L]
I[3] tlei #5 B[1P]
I[4] bro_t L1
I[5] bro_f L2
430
431
432
433
434
435
410
z = x + y;
if (z<=5) {
415
x += 1;
y -= 1;
x /=y;
. . .
if(y>0) {
. . .
}
}
头
425
I[0] read R0 T[2L]
I[1] read R7 T[3L]
I[2] addi #1 T[4L]
I[3] subi #1 T[4R] T[100R]
I[4] div W[R0]
. . .
460
I[100] tlgti #0 T[105R]
I[101] add T[107L]
. . .
I[125] bro L2
465
440
450
455
456
read
read
add
tlei
bro_t
bro_f
2R
2L
3L
R0
R7
#5

图4

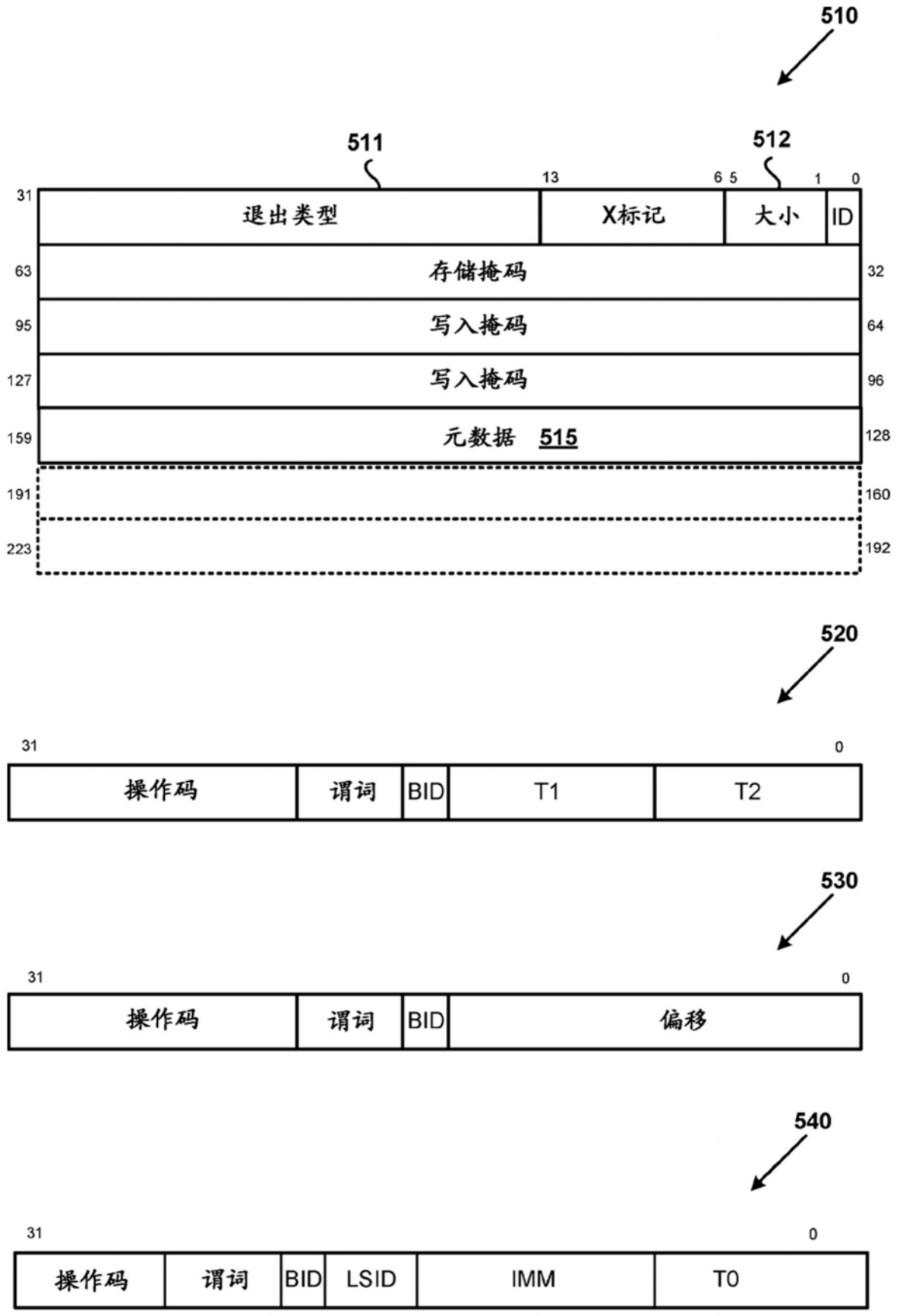

510
511
512
31
13
6 5
1
0
退出类型
X标记
大小
ID
63
存储掩码
32
95
写入掩码
64
127
写入掩码
96
159
元数据 515
128
191
160
223
192
520
31
0
操作码
谓词
BID
T1
T2
530
31
0
操作码
谓词
BID
偏移
540
31
0
操作码
谓词
BID
LSID
IMM
T0

图5

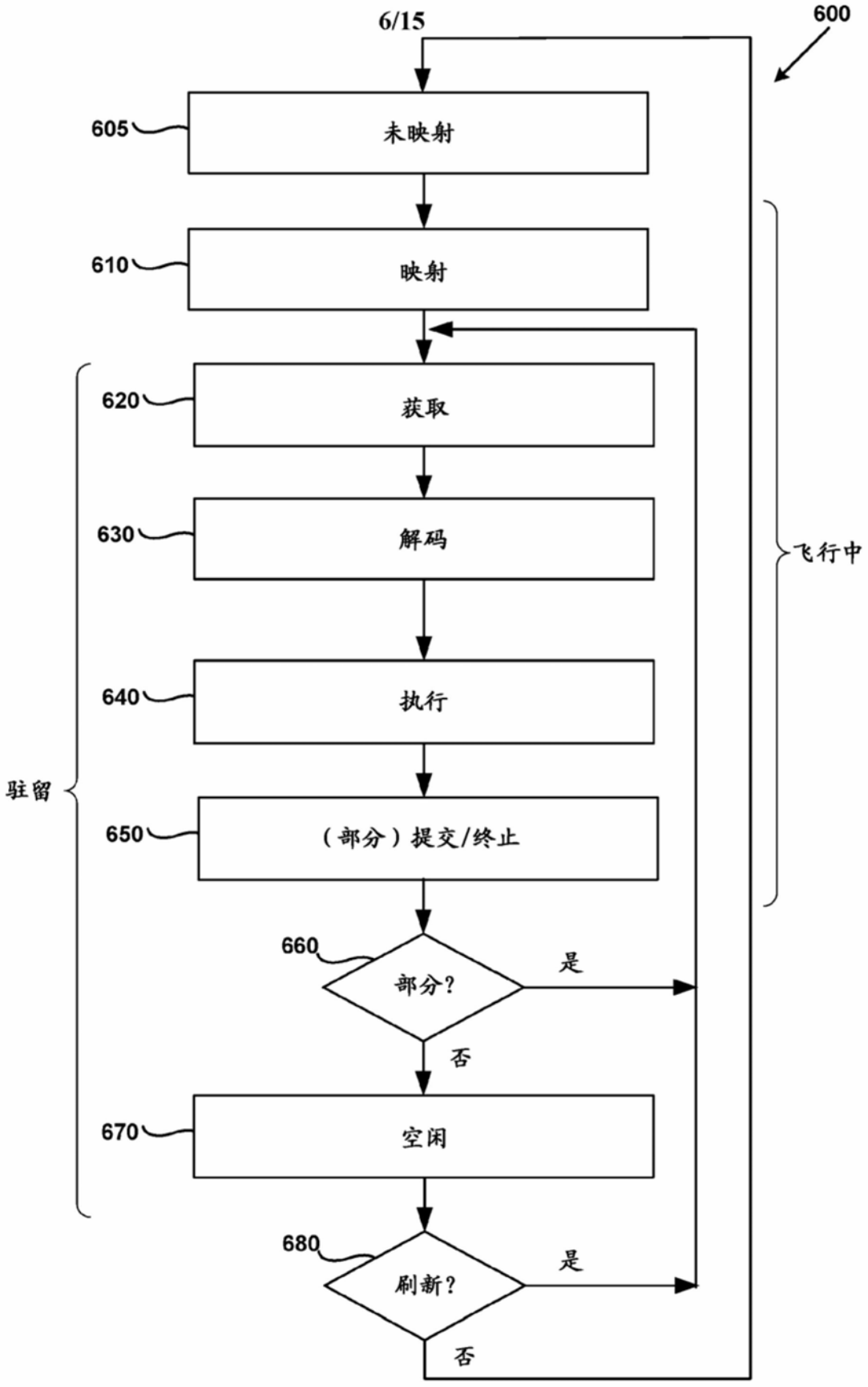

6/15
600
605
未映射
610
映射
620
获取
630
解码
640
执行
650
（部分）提交/终止
660
部分?
是
否
670
空闲
680
刷新?
是
否
驻留
飞行中

图6

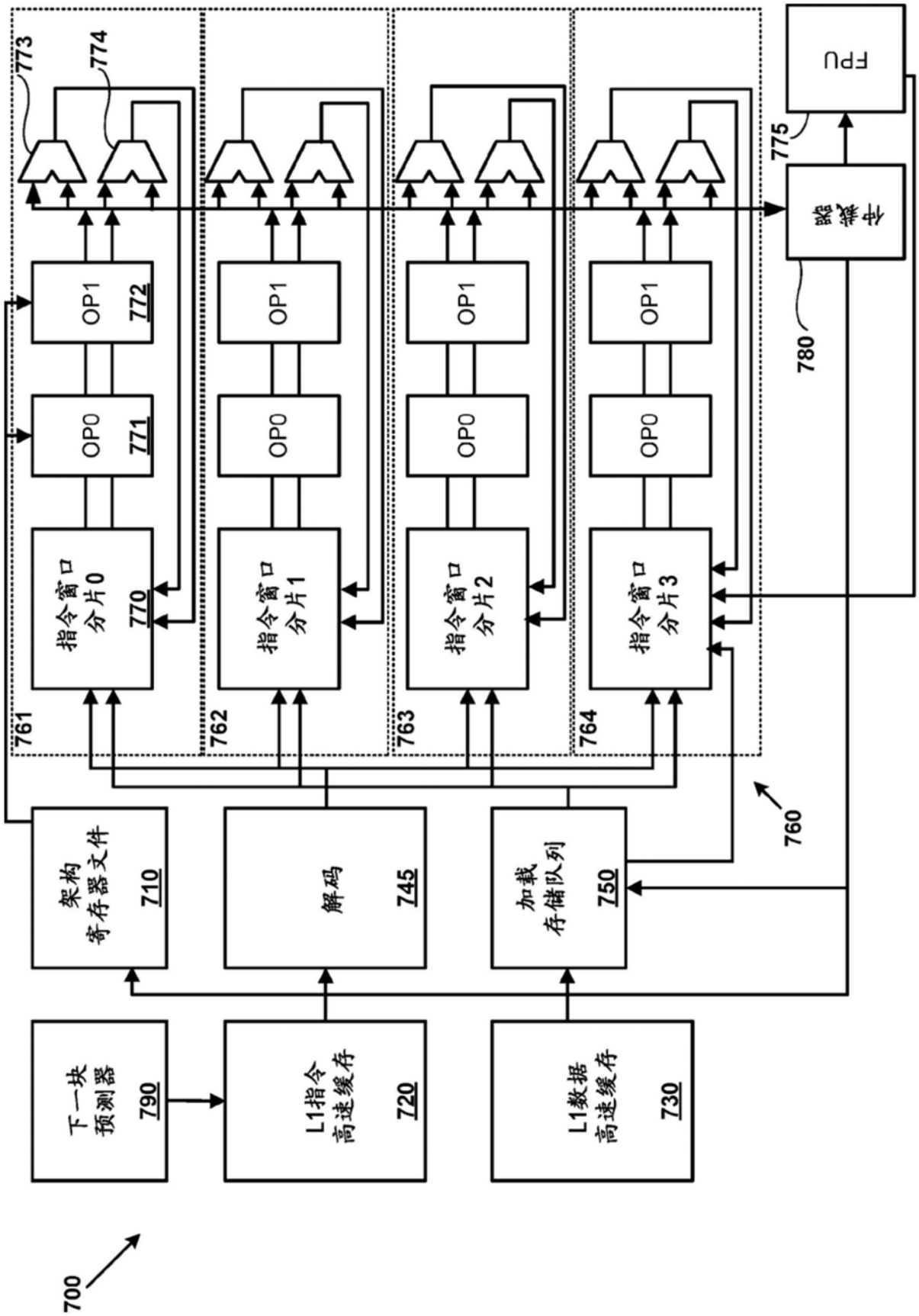

700
下一块预测器 Z90
L1指令高速缓存 Z20
L1数据高速缓存 Z30
架构寄存器文件 Z10
解码 Z45
加载存储队列 Z50
760
761
762
763
764
指令窗口分片0 Z70
指令窗口分片1
指令窗口分片2
指令窗口分片3
OP0 Z71
OP1 Z72
773
774
775
780
FPU

图7

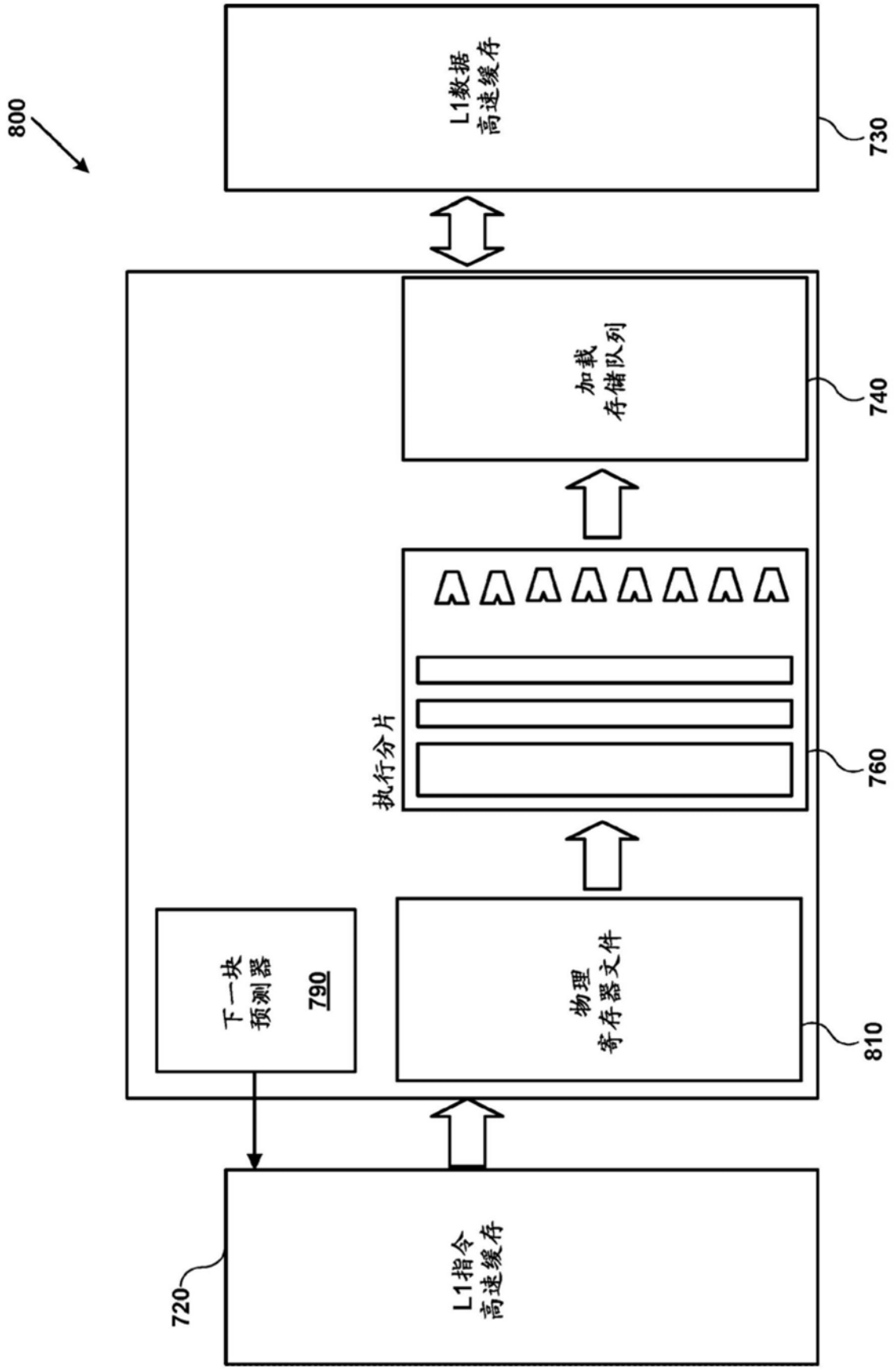

800
L1数据
高速缓存
730
加载
存储队列
740
执行分片
760
下一块
预测器
790
物理
寄存器文件
810
L1指令
高速缓存
720

图8A

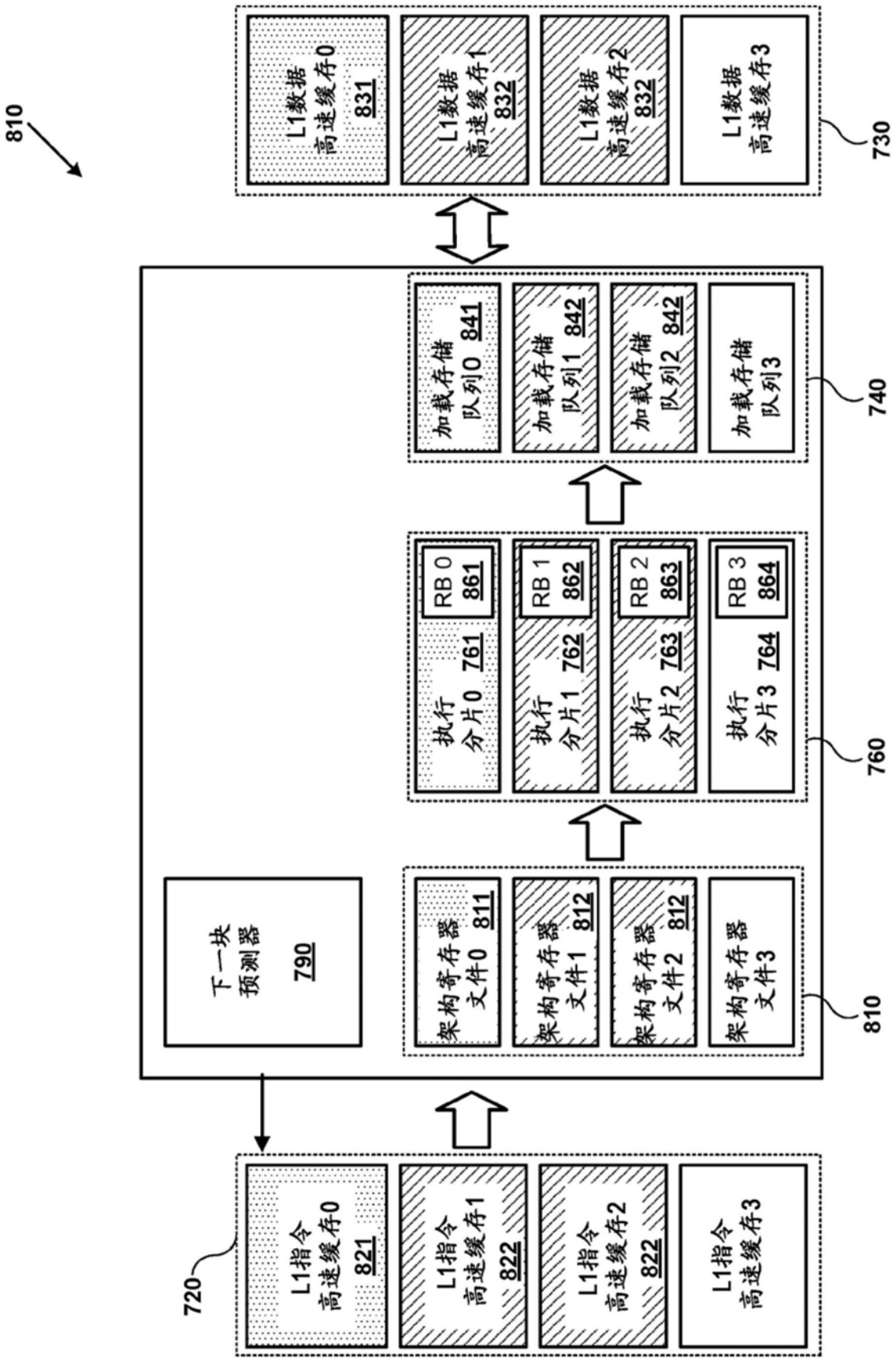

810
L1数据 高速缓存0 831
L1数据 高速缓存1 832
L1数据 高速缓存2 832
L1数据 高速缓存3
730
加载存储 队列0 841
加载存储 队列1 842
加载存储 队列2 842
加载存储 队列3
740
RB 0 861
RB 1 862
RB 2 863
RB 3 864
执行 分片0 761
执行 分片1 762
执行 分片2 763
执行 分片3 764
760
下一块 预测器 790
架构寄存器 文件0 811
架构寄存器 文件1 812
架构寄存器 文件2 812
架构寄存器 文件3
810
720
L1指令 高速缓存0 821
L1指令 高速缓存1 822
L1指令 高速缓存2 822
L1指令 高速缓存3

图8B

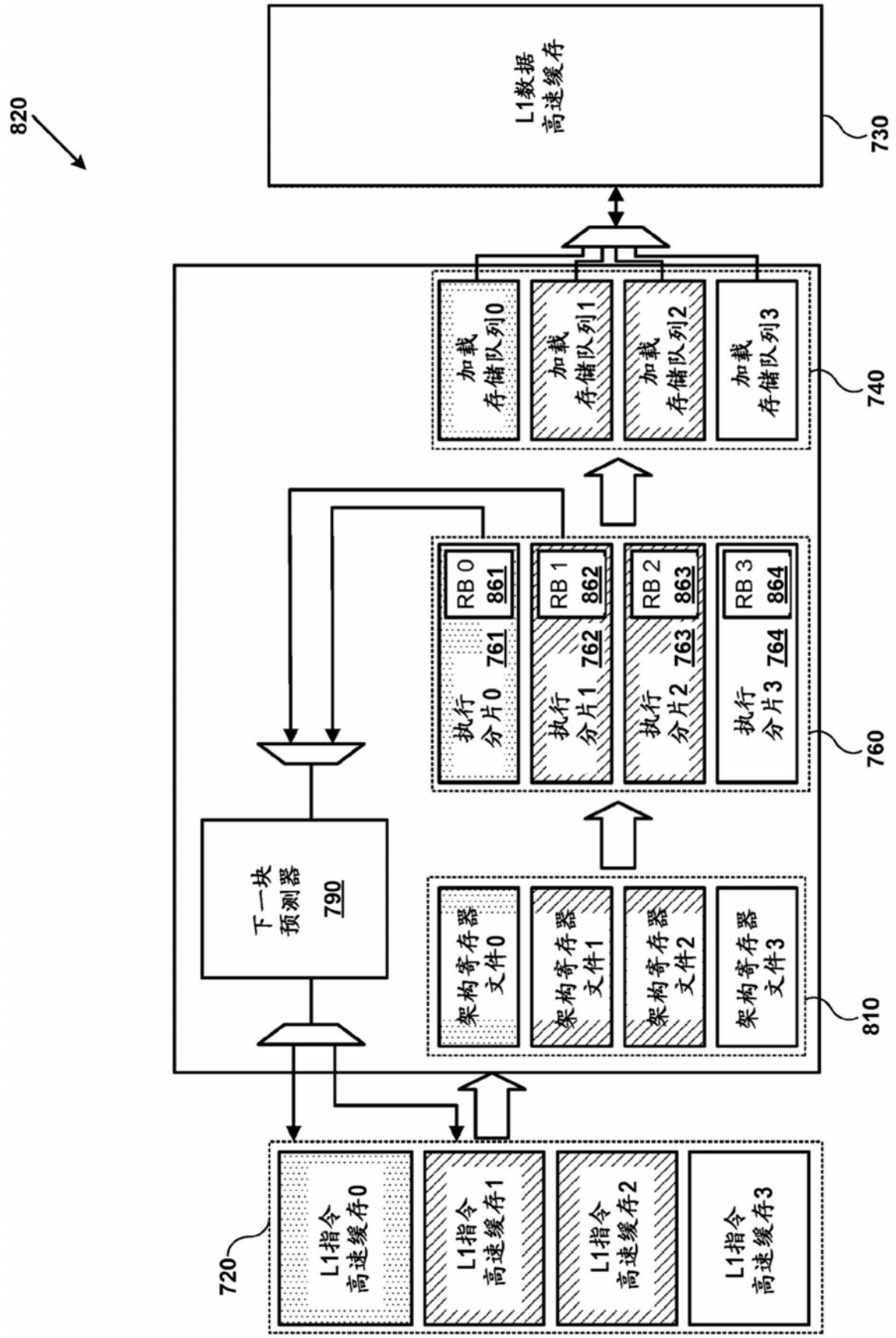

820
L1数据
高速缓存
730
加载
存储队列0
加载
存储队列1
加载
存储队列2
加载
存储队列3
740
RB 0
861
RB 1
862
RB 2
863
RB 3
864
执行
分片0
761
执行
分片1
762
执行
分片2
763
执行
分片3
764
760
下一块
预测器
790
架构寄存器
文件0
架构寄存器
文件1
架构寄存器
文件2
架构寄存器
文件3
810
720
L1指令
高速缓存0
L1指令
高速缓存1
L1指令
高速缓存2
L1指令
高速缓存3

图8C

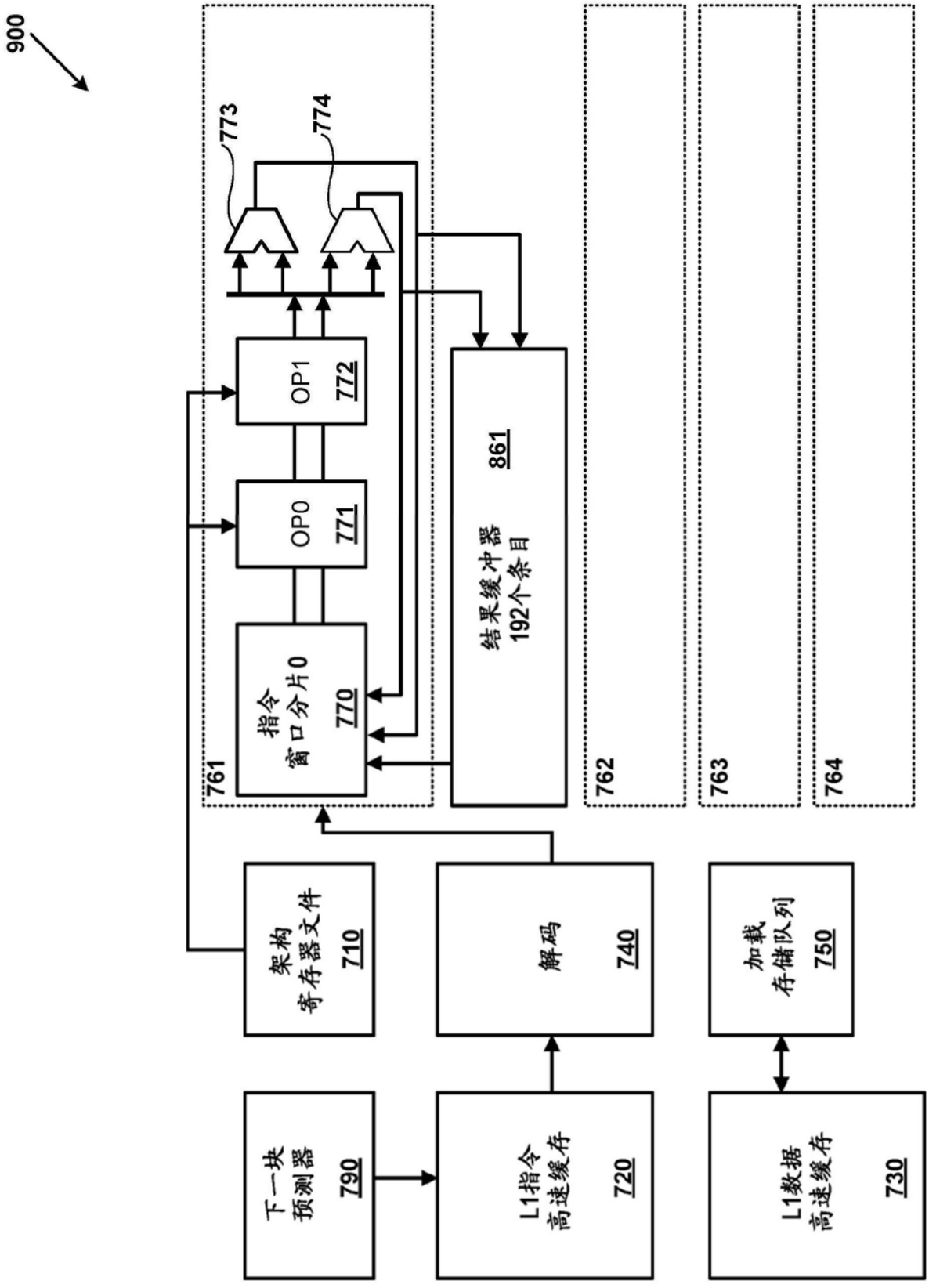

900
773
774
OP1
772
OP0
771
指令
窗口分片0
770
761
结果缓冲器
192个条目
861
762
763
764
架构
寄存器文件
710
解码
740
加载
存储队列
750
下一块
预测器
790
L1指令
高速缓存
720
L1数据
高速缓存
730

图9

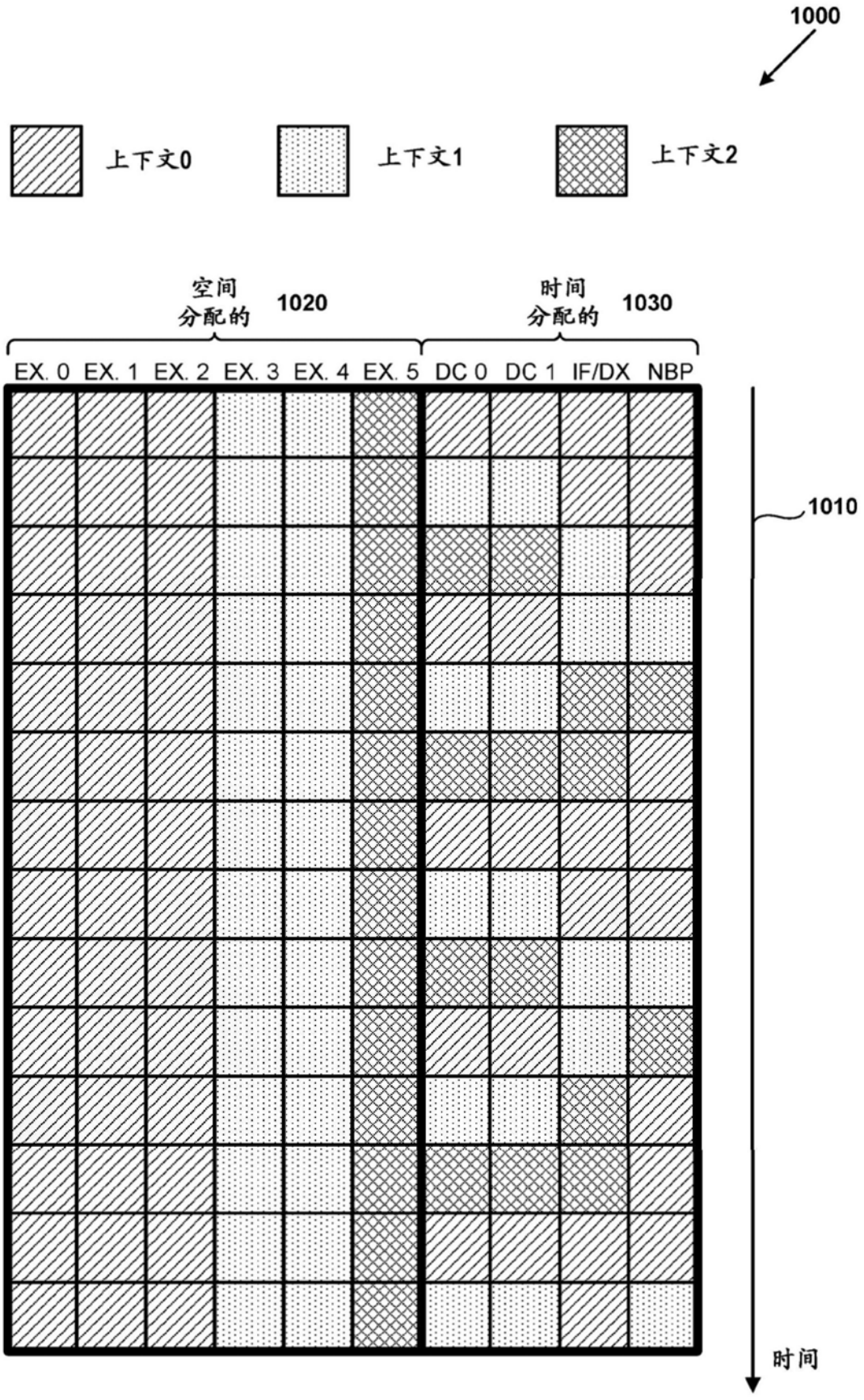

1000
上下文0
上下文1
上下文2
空间
分配的
1020
时间
分配的
1030
EX. 0
EX. 1
EX. 2
EX. 3
EX. 4
EX. 5
DC 0
DC 1
IF/DX
NBP
1010
时间

图10

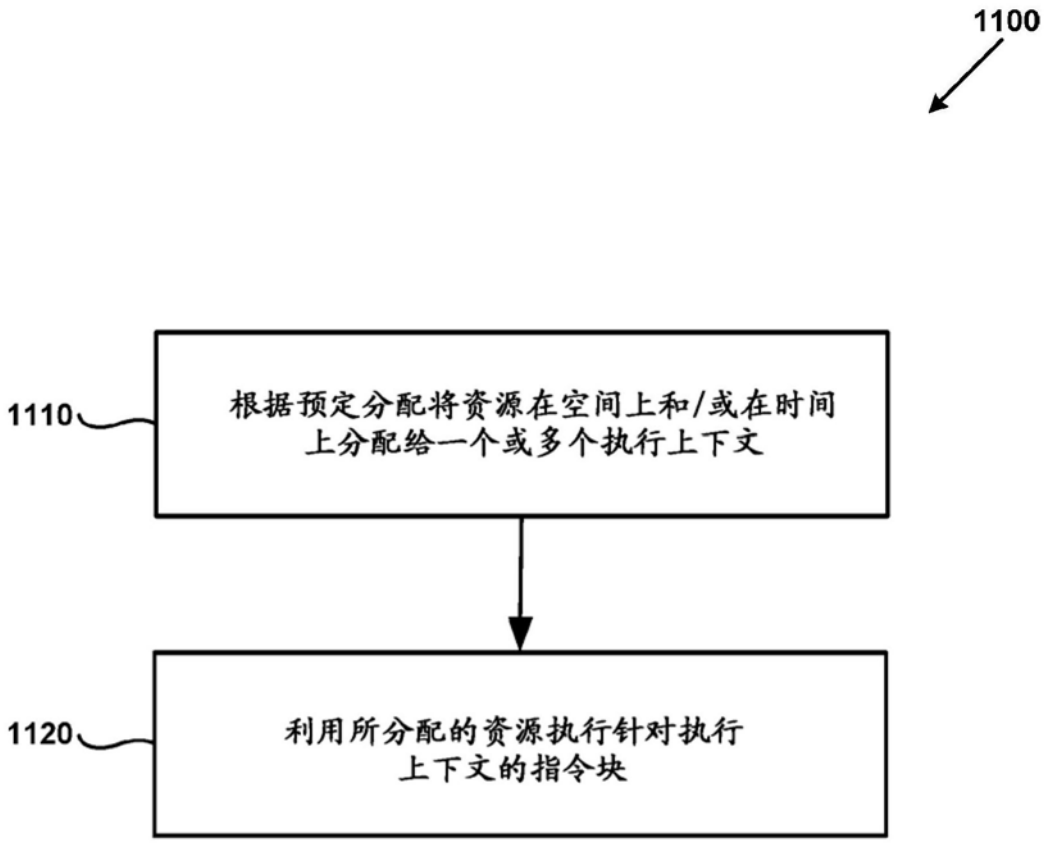

1100
1110
根据预定分配将资源在空间上和/或在时间上分配给一个或多个执行上下文
1120
利用所分配的资源执行针对执行上下文的指令块

图11

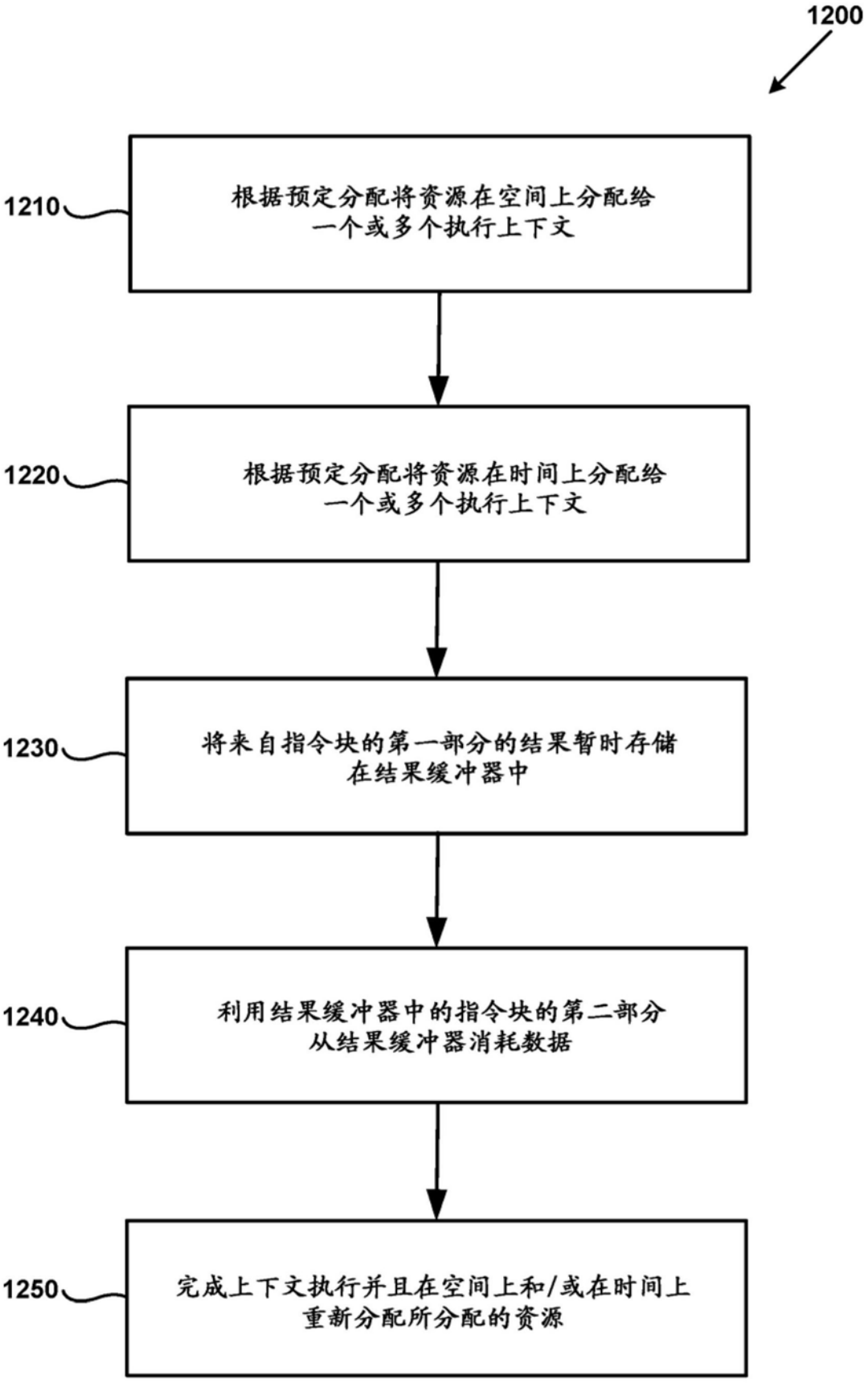

1200
1210
根据预定分配将资源在空间上分配给一个或多个执行上下文
1220
根据预定分配将资源在时间上分配给一个或多个执行上下文
1230
将来自指令块的第一部分的结果暂时存储在结果缓冲器中
1240
利用结果缓冲器中的指令块的第二部分从结果缓冲器消耗数据
1250
完成上下文执行并且在空间上和/或在时间上重新分配所分配的资源

图12

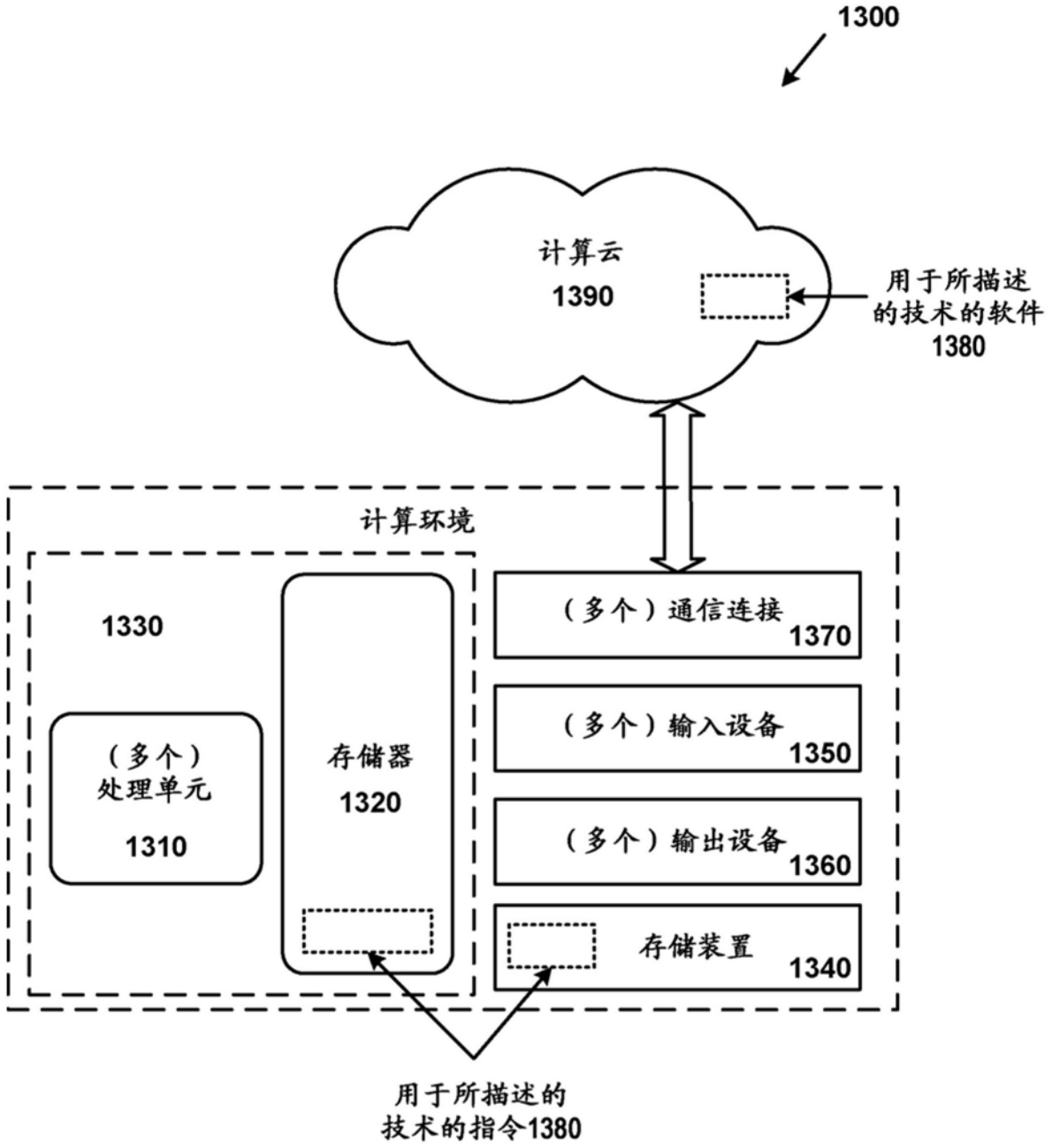

1300
计算云
1390
用于所描述
的技术的软件
1380
计算环境
1330
（多个）
处理单元
1310
存储器
1320
（多个）通信连接
1370
（多个）输入设备
1350
（多个）输出设备
1360
存储装置
1340
用于所描述的
技术的指令1380

图13