(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 108027770 B
(45)授权公告日 2022.09.23

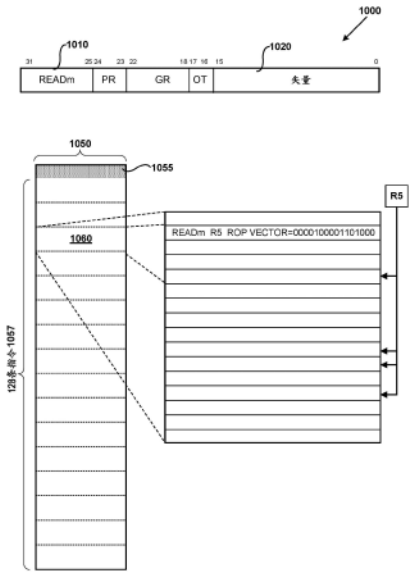
(21)申请号 201680054462.9
(22)申请日 2016.09.13
(65)同一申请的已公布的文献号
申请公布号 CN 108027770 A
(43)申请公布日 2018.05.11
(30)优先权数据
62/221,003 2015.09.19 US
15/013,842 2016.02.02 US
(85)PCT国际申请进入国家阶段日
2018.03.19
(86)PCT国际申请的申请数据
PCT/US2016/051416 2016.09.13
(87)PCT国际申请的公布数据
WO2017/048655 EN 2017.03.23
(73)专利权人 微软技术许可有限责任公司
地址 美国华盛顿州
(72)发明人 D·C·伯格 A·L·史密斯
(74)专利代理机构 北京世辉律师事务所 16093
专利代理师 王俊
(51)Int.Cl.
G06F 12/0806(2006.01)
G06F 9/38(2006.01)
(56)对比文件
US 2012204008 A1,2012.08.09
US 2009144502 A1,2009.06.04
CN 103377085 A,2013.10.30
Karthikeyan Sankaralingam,等.Distributed Microarchitectural Protocols in the TRIPS Prototype Processor”,Karthikeyan Sankaralingam.《Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture》.2009,
审查员 宁雪莹
权利要求书3页 说明书23页 附图16页

(54)发明名称

用于数据流ISA的密集读取编码

(57)摘要

公开了用于使用指令译码器来控制基于块的处理器体系架构中的存储器访问指令的执行的装置和方法,该指令译码器对具有可变数目的目标操作数的指令进行译码。在所公开的技术的一个示例中,基于块的处理器核包括指令译码器,其被配置为译码用于指令块中的指令的目标操作数,该指令被编码以将可变数目的目标操作数考虑在内;以及控制单元,其被配置为发送用于由核中的至少一个核所执行的操作的译码的目标操作数中的至少一个目标操作数的数据。在一些示例中,该指令指示具有矢量编码的目标指令。在其他示例中,可变长度格式将对一个或多个目标的指示考虑在内。



CN 108027770 B

1.一种包括主存储器和一个或多个基于块的处理器核的装置,所述核中的至少一个核包括:

指令高速缓存,存储可变长度格式指令,所述可变长度格式指令被编码为在所述指令高速缓存或所述主存储器中在第一位置处存储的第一字,所述可变长度格式指令具有操作码字段、扩展操作码字段和至少一个目标操作数字段,所述操作码字段指示所述指令的操作,所述扩展操作码字段指示所述指令是长度可变的,并且至少一个另外的目标操作数字段被编码在继所述第一字之后的不同的第二位置处的第二字中;

指令译码器,其被配置为:

从所述第一位置取回所述第一字,并且从所述操作码字段译码所述指令的操作;并且

基于所述扩展操作码字段的存在,从继所述第一字之后的在所述指令高速缓存或主存储器中的所述第二位置处取回和译码所述至少一个另外的目标操作数字段;以及

控制单元,其被配置为发送针对经译码的所述目标操作数中的每个目标操作数的指令所指定的操作的数据,至少一个指令接收针对每个目标操作数而发送的所述数据。

2.根据权利要求1所述的装置,其还包括执行单元,其被配置为使用经译码的所述操作数中的至少一个操作数来执行所述操作。

3.根据权利要求1所述的装置,其中所述指令使用指示用于所述目标操作数的指令标识符的可变数目的指令目标而被编码。

4.根据权利要求1所述的装置,其中所述指令使用指示所述指令目标的位矢量而被编码。

5.根据权利要求1所述的装置,其中所述目标操作数的一种或多种类型选自由以下各项组成的组:另一指令的标识符、全局寄存器、广播信道以及存储器地址。

6.根据权利要求1所述的装置,其中所述目标操作数中的至少一个目标操作数指示由目标指令执行的操作的断言操作数、右操作数或左操作数。

7.根据权利要求1所述的装置,其中所述译码器被配置为至少部分地基于所述指令高速缓存或主存储器内的所述指令的位置来确定目标指令以接收目标操作数数据。

8.根据权利要求1所述的装置,其中所述译码器被配置为基于在所述指令高速缓存或主存储器的不同字中被编码的扩展操作码来译码所述目标操作数。

9.根据权利要求1所述的装置,其中所述译码器被配置为至少部分地基于在所述指令高速缓存或主存储器中编码的所述指令的操作码来将所述指令译码为固定长度指令或可变数目的目标指令。

10.根据权利要求1所述的装置,其中所述目标操作数字段中的至少一个目标操作数字段是存储器地址。

11.一种控制存储器访问指令的执行的方法,所述方法包括:

取回和译码在指令高速缓存或被耦合到处理器的主存储器中的第一地址处存储的第一指令,所述第一指令被编码在所述指令高速缓存或主存储器的一个字中,所述指令高速缓存或主存储器的所述一个字包括:操作码字段,编码指定由所述指令执行的操作的操作码;和至少一个目标操作数字段,指定用于发送执行所指定的操作的结果的目标;

通过执行所述操作和向由所述至少一个目标操作数字段指定的目标发送所述结果,来执行所述第一指令;

取回和译码在所述指令高速缓存或耦合到所述处理器的主存储器中的不同第二地址处存储的第二指令，所述第二指令被编码在所述指令高速缓存或主存储器的两个字中，所述两个字中的第一字包括：操作码字段，编码与所述第一指令相同的、指定与由所述第一指令执行的操作相同的操作的操作码；和扩展操作码，指示在所述指令高速缓存或主存储器的所述两个字中的第二字中编码的目标操作数，所述第二字指示用于发送与所述第二指令相关联的操作的结果的至少一个扩展目标；

基于译码所述扩展操作码，译码在所述第二字中编码的所述目标操作数；以及

通过执行所述操作和向所指定的所述指示一个扩展目标发送所述结果，来执行所述第二指令。

12.根据权利要求11所述的方法，其中所述译码包括：

至少部分地基于经译码的所述指令在所述指令高速缓存或主存储器内的相对位置和/或经译码的所述指令在所述指令高速缓存或主存储器的一部分内的相对位置，来确定所述至少一个目标。

13.根据权利要求11所述的方法，其中所述译码包括：

至少部分地基于向量确定所指定的所述至少一个目标。

14.根据权利要求11所述的方法，其中所述译码包括：

译码指示具有特定类型的若干个目标操作数的数据；以及

确定发送用于所述特定类型的所述目标操作数的数据的目标位置。

15.根据权利要求11所述的方法，其中所述译码包括：确定目标指令操作数的类型，所述类型选自由以下各项组成的组：目标指令、广播信道、寄存器和存储器。

16.根据权利要求11所述的方法，其中所述指令是寄存器读取。

17.根据权利要求11所述的方法，其中不同的所述第二字能够可选地针对具有相同操作码的指令进行编码。

18.根据权利要求11所述的方法，其中所述第一指令包括第一字段，所述第一字段指定所述第一指令中的目标操作数字段的数目，并且所述第二指令包括第二字段，所述第二字段指定所述第二指令中的目标操作数字段的数目，所述第一字段和所述第二字段指定不同数目的目标操作数字段。

19.一个或多个存储用于指令块的计算机可读指令的计算机可读存储介质，所述计算机可读指令在由处理器执行时使得所述处理器执行方法，所述计算机可读指令包括：

用于分析源代码和/或目标代码以确定针对所述指令块的操作数依存性的指令；以及

用于将所述源代码和/或目标代码转换成针对所述指令块的计算机可执行码的指令，所述计算机可执行代码包括：

第一指令，所述第一指令以可变长度格式被编码，作为具有预定位数的单个字被存储在所述指令块中，所述第一指令具有至少一个被编码的目标操作数字段和在固定操作码字段位置中编码的操作码；以及

第二指令，所述第二指令以可变长度格式被编码，作为多个字被存储在所述指令块中，所述字中的每个字具有相同的预定位数，所述第二指令具有在所述多个字中的不同数目的至少两个被编码的目标操作数字段和相同的在固定操作码字段位置中编码的操作码。

20.根据权利要求19所述的计算机可读存储介质，其中所述指令还包括：

用于为所述第二指令的所述目标操作数指派优先级的指令;以及

用于布置所述第二指令的所述目标操作数的指令,以使较高优先级目标操作数将在较低优先级目标操作数之前被发送到较高优先级目标操作数的目标指令。

21.根据权利要求19所述的计算机可读存储介质,其中所述指令还包括:

用于重新排序目标指令的指令,使得针对所述第二指令的所有目标指令将全部位于能够由进行发送的所述指令寻址的所述指令块的一部分内。

22.根据权利要求19所述的计算机可读存储介质,其中所述第二指令还包括扩展操作码字段,所述扩展操作码字段具有的目标操作数字段的数目不同于所述第一指令目标操作数字段。

用于数据流ISA的密集读取编码

背景技术

[0001] 由于摩尔定律所断定的持续的晶体管扩展，微处理器已经从晶体管数的持续增加、集成电路成本、制造资本、时钟频率、以及能量效率中收益，而相关的处理器指令集架构(ISA)却很小变化。然而，从在过去40年里驱动半导体工业的光刻扩展实现的益处正在放缓或者甚至反转。精简指令集计算(RISC)架构已经成为处理器设计中的主导典范很多年。乱序超标量实现尚未在面积或性能方面展现出持续改进。因此，存在对于扩展性能改进的处理器ISA改进的足够机会。

发明内容

[0002] 公开了用于配置、操作和编译用于包括显式数据图执行(EDGE)体系架构的基于块的处理器体系架构(BB-ISA)的代码的方法、装置和计算机可读存储设备。所描述的用于例如改进处理器性能和/或减少能量消耗的解决方案的技术和工具可以分开地实现，或者以彼此的各种组合实现。如将在下文更充分地描述的，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。本领域普通技术人员将容易理解，所公开的技术可以在各种计算平台中实现，这些计算平台包括但不限于服务器、大型机、手机、智能手机、PDA、手持式设备、手持式计算机、PDA、触摸屏平板设备、平板电脑、可穿戴式电脑和膝上型计算机。

[0003] 在所公开的技术的一些示例中，一种基于块的处理器包括指令译码器，其被配置为译码用于指令块内的指令的目标操作数。该指令被编码以将可变数目的目标操作数考虑在内，例如，用于可变数目的相关联的目标指令，其将接收通过执行由执行指令所指定的操作而生成的一个或多个值。处理器核还包括控制单元，其被配置为向目标指令发送由指令核执行的操作的经译码的目标操作数中的至少一个目标操作数的数据。在一些示例中，基于块的处理器译码用指示目标操作数的可变数目的指令标识符编码的指令。在一些示例中，经译码的指令用指令矢量进行编码，其指示目标操作数要发送到的一个或多个目标指令。

[0004] 提供本发明内容是为了以简化形式介绍下文在具体实施方式中进一步描述的概念选择。本发明内容并非旨在标识所要求保护的主题的关键特征或基本特征，也并非旨在用于限制所要求保护的主题的范围。从下面参照附图进行的具体实施方式中，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

[0005] 图1图示了如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。

[0006] 图2图示了如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。

[0007] 图3图示了根据所公开的技术的某些示例的若干个指令块。

[0008] 图4图示了如可以在所公开的技术的一些示例中使用的源代码及指令块的部分。

[0009] 图5图示了如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部和指令。

[0010] 图6是图示了当指令块被映射、执行和引退时指派给指令块的若干个状态的状态图。

[0011] 图7图示了如可以在所公开的技术的一些示例中使用的可变长度指令的示例。

[0012] 图8图示了如可以在所公开的技术的某些示例中使用的示例目标操作数格式。

[0013] 图9图示了如可以在所公开的技术的某些示例中使用的示例可变长度指令。

[0014] 图10图示了如可以在所公开的技术的某些示例中使用的示例目标矢量指令。

[0015] 图11图示了如可以在所公开的技术的某些示例中使用的示例目标矢量指令。

[0016] 图12图示了如可以在所公开的技术的某些示例中使用的基于块的处理器配置。

[0017] 图13是概述了如可以在所公开的技术的某些示例中执行的译码和执行指定可变数目的目标操作数的指令的示例方法的流程图。

[0018] 图14A和图14B图示了如可以在所公开的技术的某些示例中使用的源代码和汇编代码的示例。

[0019] 图15是概述了如在所公开的技术的某些示例中可以执行的将源代码和/或目标代码转换成用于基于块的处理器的计算机可执行代码的示例方法的流程图。

[0020] 图16是图示了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

[0021] I.总体考虑

[0022] 本公开在不旨在以任何方式进行限制的代表性实施例的上下文中进行阐述。

[0023] 如本申请中所使用的，除非上下文另外明确指出，否则单数形式“一”、“一种”和“该”包括复数形式。附加地，术语“包含(includes)”意味着“包括(comprises)”。进一步地，术语“耦合”涵盖机械的、电的、磁性的、光学的以及将多个项耦合或链接在一起的其他实际方式，并且不排除耦合项之间的中间元件的存在。更进一步地，如本文中所使用的，术语“和/或”是指短语中的任一项或多项的组合。

[0024] 本文中所描述的系统、方法和装置不应以任何方式被解释为限制性的。相反，本公开涉及以单独和彼此的各种组合与子组合方式的各种公开实施例的所有新颖且非显而易见的特征和方面。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，所公开的内容和方法也不要求任何一个或多个特定优点存在或者问题被解决。更进一步地，所公开的实施例的任何特征或者方面可以彼此以各种组合和子组合被使用。

[0025] 尽管为了便于呈现而以特定顺序的次序描述了所公开的方法中的一些方法的操作，但是应当理解，除非特定排序由下面阐述的特定语言所要求，，否则说明书的这种方式涵盖重新布置。例如，顺序地描述的操作可以在一些情况下被重新布置或并行执行。而且，为了简单起见，附图可能并未示出所公开的内容和方法可以与其他内容和方法结合使用的各种方式。附加地，说明书中有时使用诸如“产生”、“生成”、“显示”、“接收”、“发射”、“验证”、“执行”和“发起”之类的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层描述。与这些术语相对应的实际操作将根据具体实现方式而变化，并且容易被本领域的普通技术人员辨别。

[0026] 为了更好理解的目的，已经提供了本文中参照本公开的装置或方法给出的操作理论、科学原理或其他理论描述，并且不旨在对范围的限制。所附权利要求书中的装置和方法不限于以由这些操作理论所描述的方式实现的那些装置和方法。

[0027] 所公开的方法中的任一方法被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))并且在计算机(例如，任何商业可用计算机，包括智能电话或包括计算硬件的其他移动设备的)上执行的计算机可执行指令。用于实现所公开的技术的计算机可执行指令中的任一计算机可执行指令以及在实现所公开的实施例期间创建和使用的任何数据可以被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经由网络浏览器或其他软件应用(诸如远程计算应用)访问或下载的专用软件应用或软件应用的一部分。这样的软件可以例如使用一个或多个网络计算机在单个本地计算机上(例如，具有在任何合适的商业上可用的计算机上执行的通用处理器和/或基于块的处理器)或者在网络环境中(例如，经由互联网、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)中执行。

[0028] 为了清楚起见，仅描述了基于软件的实现方式的某些选定方面。本领域公知的其他细节被省略。例如，应当领会，所公开的技术不限于任何特定计算机语言或程序。比如，所公开的技术可以通过用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众所周知的，并且不需要在本公开中进行详细阐述。

[0029] 更进一步地，可以通过合适的通信手段来上载、下载或远程访问基于软件的实施例(包括例如用于使得计算机执行所公开的方法中的任一方法的计算机可执行指令)中的任一基于软件的实施例。这种合适的通信手段包括例如因特网、万维网、内联网、软件应用、电缆(包括光缆)、磁通信、电磁通信(包括RF、微波和红外通信)、电子通信、或其他这样的通信手段。

[0030] II.对所公开的技术的介绍

[0031] 超标量乱序微体系架构采用大量的电路资源来重命名寄存器，按数据流次序调度指令，在误推测后清理，并且针对精确异常而按序引退结果。这包括昂贵的电路，诸如深的多端口寄存器文件、用于数据流指令调度唤醒的多端口内容可访问存储器(CAM)、以及许多宽总线多路复用器和旁路网络，所有的这些全部是资源密集型的。例如，多读取多写入RAM的基于FPGA的实现方式通常要求复制、多循环操作、时钟加倍、组交错、实况值表和其他昂贵技术的混合。

[0032] 所公开的技术可以通过应用包括高指令级并行性(ILP)、乱序(OoO)、超标量执行的技术来实现性能增强，同时避免处理器硬件和相关软件中的大量的复杂性和开销。在所公开的技术的一些示例中，基于块的处理器使用被设计用于面积和能量效率高的ILP执行的EDGE ISA。在一些示例中，使用EDGE体系架构和相关编译器使用巧妙处理，远离寄存器重命名、CAM和复杂性中的很多种。

[0033] 在所公开的技术的某些示例中，EDGE ISA可以消除对包括寄存器重命名、数据流分析、误推测恢复和按序引退在内的一个或多个复杂体系架构特征的需要，同时支持诸如C和C++之类的主流编程语言。在所公开技术的某些示例中，基于块的处理器执行多个两个或

更多个指令作为原子块。基于块的指令可以被用于以更加明确的方式来表达程序数据流和/或指令流的语义，从而改善编译器和处理器的性能。在所公开的技术的某些示例中，显式数据图执行指令集体系架构(EDGE ISA)包括关于程序控制流的信息，其被用来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/或节省能量。

[0034] 在所公开的技术的一些示例中，指令块内组织的指令被原子地提取、执行和提交。块内部的指令以数据流次序执行，其减少或消除了寄存器重命名的使用，并且提供功率有效的OoO执行。编译器可以被用来通过ISA显式地编码数据依存性，从而减少或消除处理器核控制逻辑在运行时间重新发现依存性的负担。使用所断言的执行，块内分支可以被转换为数据流指令，并且除了存储器依存性之外的依存性可以被限制于直接数据依存性。所公开的目标形式编码技术允许块内的指令经由操作数缓冲器直接传达它们的操作数，从而减少对耗电的多端口物理寄存器文件的访问。

[0035] 在指令块之间，指令可以使用存储器和寄存器进行通信。因此，通过利用混合数据流执行模型，EDGE体系架构仍然可以支持命令式编程语言和顺序的存储器语义，但是期望地还享受具有近按序功率效率和复杂性的乱序执行的好处。

[0036] 公开了用于生成并且使用用于基于块的处理器的基于块的分支元数据的装置、方法和计算机可读存储介质。在所公开的技术的某些示例中，指令块包括指令块头部和多个指令。换句话说，指令块所执行的指令影响状态，或者不会作为一单元来影响状态。

[0037] 在所公开的技术的一些示例中，硬件结构存储指示针对若干个存储器访问指令(包括存储器加载和存储器存储指令)要粘附的执行顺序的数据。耦合到处理器核的控制单元至少部分地基于存储在硬件结构中的数据来控制存储器访问指令的发出。因此，可以避免存储器读取/写入危险，同时允许指令块中的指令在其依存性可用时立即执行。

[0038] 如相关领域的普通技术人员将容易理解到，所公开的技术的实现方式的范围可能存在各种领域和性能的折衷的情况。

[0039] III.示例基于块的处理器

[0040] 图1是如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体系架构(ISA)执行原子指令块，该指令集体系架构描述处理器操作的若干方面，包括寄存器模型、由基于块的指令执行的若干个定义的操作、存储器模型、中断和其他体系架构特征。基于块的处理器包括多个处理核110，其包括处理器核111。

[0041] 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之间传递数据并且控制信号。核互连120可以使用电的、光学的、磁性的或其他合适的通信技术传送和接收信号，并且可以依据特定的期望配置提供根据若干个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结构。在一些示例中，核110中的任一核可以连接到其他核中的任一核，而在其他示例中，一些核仅连接到其他核的子集。例如，每个核只能连接到最近的4个、8个或20个相邻核。核互连120可以被用来将输入/输出数据传送至核以及从核传送输入/输出数据，以及将控制信号和其他信息信号传送至核以及从核传送控制信号和其他信息信号。例如，核110中的每个核可以接收并且传送指示各个核中的每个核当前正在执行的指令的执行状态的信号量。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

接线，而在其他示例中，核互连可以包括用于多路复用一条或多条互连接线上的数据信号的电路；以及开关和/或路由组件，其包括有源信号驱动器和中继器；或其他合适的电路。在所公开的技术的一些示例中，在处理器100内传送并且向处理器100/从处理器100传送的信号不限于全摆幅电数字信号，而是处理器可以被配置为包括差分信号、脉冲信号或用于传送数据和控制信号的其他合适的信号。

[0042] 在图1的示例中，处理器的存储器接口140包括接口逻辑，其被用来连接到附加存储器，例如，位于除了处理器100之外的另一集成电路上的存储器。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，而主存储器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包括直接存储器访问(DMA)控制器，其允许在存储器中传送数据块，而不使用一个或多个寄存器文件和/或处理器100。在一些示例中，存储器接口管理虚拟存储器的分配，从而扩展可用主存储器155。

[0043] I/O接口145包括电路，其用于从其他组件接收和向其他组件发送输入和输出信号，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信号可以是同步的也可以是异步的。在一些示例中，I/O接口的全部或一部分使用存储器映射I/O技术结合存储器接口140来实现。

[0044] 基于块的处理器100还可以包括控制单元160。控制单元160监督处理器100的操作。可以由控制单元160执行的操作可以包括对核的分配和解除分配，以用于执行指令处理；控制核中的任何核、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；修改执行流；以及验证控制流中的分支指令、指令头部和其他改变的目标位置。控制单元160可以根据表示出口点的控制流和元数据信息以及用于指令块的控制流概率来生成和控制处理器。

[0045] 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(例如，存储在一个或多个寄存器文件中的程序计数器)的读取和写入。在所公开的技术的一些示例中，控制单元160至少部分地使用处理核110中的一个或多个处理核来实现，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至少部分地使用以下各项中的一项或多项来实现：硬连线有限状态机、可编程微代码、可编程门阵列或其他合适的控制电路。在备选示例中，可以由核110中的一个或多个核来执行控制单元功能。

[0046] 控制单元160包括调度器165，其被用来向处理器核110分配指令块。如本文中所使用的，调度器块分配是指指令块的引导操作，其包括：发起指令块映射、提取、译码、执行、提交、中止、空闲和刷新指令块。进一步地，指令调度是指在指令块内调度指令的发出和执行。例如，基于指令依存性和指示用于存储器访问指令的相对排序的数据，控制单元160可以确定准备好发出指令块中的哪个或哪些指令，并且发起指令的发出和执行。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。调度器165调度指令流，其包括分配和解除分配用于执行指令处理的核；控制核中的任何核、寄存

器文件、存储器接口140和/或I/O接口之间的输入数据和输出数据。如下文进一步详细讨论的，控制单元160还包括存储器访问指令硬件结构167，其可以被用来存储包括指示用于执行存储器访问指令的相对排序的数据，该存储器访问指令硬件结构167诸如存储掩码或存储装置计数器和存储矢量寄存器，其存储指示哪些指令(例如，哪些加载和/或存储指令)已经在指令块内执行的数据。

[0047] 基于块的处理器100还包括时钟发生器170，其将一个或多个时钟信号分发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，时钟的一部分被选通以当处理器组件中的一些处理器组件未被使用时允许功率节省。在一些示例中，使用锁相环(PLL)生成时钟信号以生成具有固定的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缘(例如，上升沿)上被触发，而在其他示例中，接收电路中的至少一些电路被上升时钟边缘和下降时钟边缘被触发。在一些示例中，时钟信号可以光学地或无线地传送。

[0048] IV.基于块的处理器核示例

[0049] 图2是进一步详述用于基于块的处理器100的示例微体系架构(以及具体地、如可以在所公开的技术的某些示例中使用的、基于块的处理器核中的一个基于块的处理器核的实例)的框图。为了便于解释，示例性的基于块的处理器核被图示为具有五个阶段明：取指(IF)，译码(DC)，操作数取回，执行(EX)和存储器/数据访问(LS)。然而，相关领域的普通技术人员将容易理解可以修改对所图示的微体系架构的修改(诸如添加/移除阶段，执行操作的添加/移除单元以及其他实现细节)，以使特定应用适于基于块的处理器。

[0050] 如图2所示，处理器核111包括控制单元205，其生成控制信号以调节核操作并且使用指令调度器206调度核内的指令流。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：生成并且使用存储器访问指令编码；分配和解除分配用于执行指令处理的核；控制核中的任何核、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。

[0051] 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，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，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指令调度器206。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优先级、预测和其他数据，其可以至少部分地用于用指令调度器206来调度指令块。如相关领域的普通技术人员将容易地理解的，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可以被用来实现用于指令调度器206的硬件。

[0052] 控制单元205还包括用于存储控制流信息和元数据的存储器(例如，SRAM或寄存器中的)。例如，指示存储器访问指令相对排序的数据可以被存储在诸如存储指令数据存储装置207之类的硬件结构中。存储指令数据存储装置207可以存储用于存储掩码的数据(例如，通过复制在指令块中编码的数据生成的或者当译码指令时由指令译码器生成的)，存储计数数据(例如，指示在特定的下一个指令能够发出之前要被执行的若干个存储器存储装置的数据)，存储装置计数器(例如，存储指示已经发出或执行的被存储的指令数目的数据的计数器)，存储矢量寄存器(例如，存储指示已经执行了哪些类型的存储器访问指令以及什

么类型的存储器访问指令的数据)，以及经掩码的存储矢量寄存器数据(例如，通过将存储掩码应用于存储矢量寄存器生成的数据)。在一些示例中，存储指令数据存储装置207包括计数器，其跟踪已经执行的存储器访问指令的数目和类型。在一些示例中，存储指令数据存储装置207将指示指令标识符、断言、断言路径、加载/存储的数目或可以被用来确定存储器访问指令的执行的相对排序的其他信息的数据存储在表(例如，多维表)中。

[0053] 指令译码器228和229可以指定用于发出和执行块内的加载和存储指令的相对顺序。例如，数值加载/存储标识符(LSID)可以被指派给每个存储器访问指令，或仅被指派给存储器存储指令。编号较高的LSID指示该指令应当在较低编号的LSID之后执行。在一些示例中，处理器可以确定两个加载/存储指令不冲突(例如，基于用于指令的读取/写入地址)并且可以以不同的顺序执行指令，尽管机器的结果状态不应与犹如指令已经在指定的LSID排序中执行的结果状态不同。在一些示例中，具有互斥断言值的加载/存储指令可以使用相同的LSID值。例如，如果第一加载/存储指令基于值p为真而被断定，而第二加载/存储指令基于值p为假而被断定，则每条指令可以具有相同的LSID值。

[0054] 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(例如，被存储在一个或多个寄存器文件中的程序计数器)的读取和写入。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和/或指令调度器206至少部分地使用硬连线有限状态机、可编程微代码、可编程门阵列或其他合适的控制电路中的一个或多个来实现。

[0055] 示例性处理器核111包括两个指令窗口210和211，每个指令窗口可以被配置为执行指令块。在所公开的技术的一些示例中，指令块是基于块的处理器指令的原子集合，其包括指令块头部和多个一个或多个指令。如下文将进一步讨论的，指令块头部包括可以被用来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。依据所使用的特定ISA和处理器硬件，还可以在指令的执行期间使用指令块头部，并且通过例如允许指令和/或数据的早期取回，改进的分支断定，推测的执行，提高能源效率以及改善代码紧凑性来提高执行指令块的性能。在其他示例中，不同数目的指令窗口是可能的，诸如一个、四个、八个或其他数目的指令窗口。

[0056] 指令窗口210和211中的每个指令窗口都可以从连接到互连总线和指令高速缓存227的输入端口220，221和222中的一个或多个输入端口接收指令和数据，该互连总线和指令高速缓存227又连接到指令译码器228和229。附加的控制信号还可以在附加的输入端口225上被接收。指令译码器228和229中的每个指令译码器译码指令块的指令头部和/或指令，并且将译码后的指令存储在位于每个相应的指令窗口210和211中的存储器存储装置215和216内。进一步地，译码器228和229中的每个译码器可以将数据发送到控制单元205，例如，以根据在指令块头部中或者在指令中指定的执行标志来配置处理器核111的操作。指令译码器228和229中的每个指令译码器被配置为生成针对指示指令块中的一个或多个存储器访问指令的相对排序的标识符。这些标识符可以被用来确定所有要针对指令块执行的存储器访问指令已经被执行。例如，指令译码器可以分析指令(例如，通过构建控制流程图或等同物)以确定与块中的存储器访问指令相关联的断言。基于断言，确定某些存储器访问指令必须在其他存储器访问或跳转指令之前执行，以便允许正确的指令块实现。

[0057] 处理器核111还包括耦合到L1(第一级)高速缓存器235的寄存器文件230。寄存器

文件230存储用于在基于块的处理器体系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将数据存储在寄存器文件中的两个或更多个写入端口，以及具有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，寄存器中的每个寄存器保持32位数据的字。(为了便于解释，除非另有说明，否则本申请将32位数据作为字来表示。根据所公开的技术的合适处理器可以使用8位字、16位字、64位字、128位字、256位字或另一数字位字来操作)。在一些示例中，寄存器文件230内的寄存器中的一些寄存器可以被分配用于特殊目的。例如，寄存器中的一些寄存器可以专用为系统寄存器示例，其示例包括存储常量值(例如，全零字)、一个或多个程序计数器(PC)(其指示正在被执行的程序线程的当前地址)、物理核数目、逻辑核数目、核分配拓扑、核控制标志、执行标志、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、一个或每个程序计数器，以允许在一个或多个处理器核和/或处理器上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，而非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能受到操作系统或其他监督式计算机指令的约束。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装置来实现。针对给定处理器(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。

[0058] 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，根据处理器ISA和配置，与处理器核相关联的各个寄存器文件可以被静态地或动态地组合以形成更大的文件。

[0059] 如图2所示，指令窗口210的存储器存储装置215包括若干个译码指令241、左操作数(LOP)缓冲器242、右操作数(ROP)缓冲器243、断言缓冲器244、三个广播信道245以及指令记分板247。如图2所示，在所公开的技术的一些示例中，指令块的每个指令被分解成一行译码后的指令、左操作数和右操作数以及记分板数据。译码后的指令241可以包括作为位级控制信号存储的指令的部分译码版本或完全译码版本。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、由先前发出的指令计算的操作数、或其他操作数值)，直到它们各自的译码指令已准备好执行。指令操作数和断言分别从操作数缓冲器242和243以及断言缓冲器244而非寄存器文件读取。指令记分板245可以包括用于针对指令的断言的缓冲器，其包括用于组合由多个指令发送到指令的断言的线或(wire-OR)逻辑。

[0060] 第二指令窗口211的存储器存储装置216存储与存储器存储装置215相似的指令信息(译码后的指令、操作数和记分板)，但为了简单起见未在图2中示出。指令块可以由第二指令窗口211同时或顺序地相对于第一指令窗口执行，这受ISA约束并且如由控制单元205指导。

[0061] 在所公开的技术的一些示例中，前端流水线阶段IF和DC可以从后端流水线阶段(IS、EX、LS)运行去耦合。控制单元可以每个时钟周期将两个指令取回并且译码到指令窗口210和211中的每个指令窗口中。控制单元205使用计分板245来提供指令窗口数据流调度逻

辑以监视每个译码指令的输入(例如,每个相应的指令的一个或多个断言和一个或多个操作数)的就绪状态。控制单元205进一步监视指示存储器访问指令的相对排序的数据(例如,使用由指令译码器生成的加载/存储标识符)以及指示已经执行了哪些指令的数据(例如,通过跟踪每条指令和/或维持已经发出的若干个存储器存储指令的计数)。当用于特定译码指令的所有输入操作数和一个或多个断言就绪时,以及任何先前排序后的存储器访问指令(例如,先前排序后的存储器存储指令)已经发出和/或被执行时,指令的发出就绪。控制单元205然后在每个周期中发起(发出)一个或多个下一指令(例如,编号最小的就绪指令)的执行,并且基于译码后的指令和指令的输入操作数的控制信号被发送到功能单元260中的一个或多个功能单元以供执行。译码后的指令还可以对若干个就绪事件进行编码。控制单元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的就绪零输入指令开始,执行继续处理以零输入指令作为目标的指令。

[0062] 译码后的指令241不需要按照它们被布置在指令窗口210的存储器存储装置215内的相同顺序来执行。相反,指令记分板245被用来跟踪译码后的指令的依存性,并且当依存性已经被满足时,相关联的个体译码指令被调度以供执行。例如,当针对相应指令已经满足依存性时,相应指令的引用可以被推送到就绪队列上,并且就绪指令可以从就绪队列中以先进先出(FIFO)顺序被调度。对于与生成的加载存储标识符(LSID)相关联的指令,执行顺序也将遵循所生成的指令LSID中所枚举的优先级,或者按照指令看起来好像按指定顺序执行的顺序执行。

[0063] 在记分板245中存储的信息可以包括但不限于一个或多个相关联的指令的执行断言(诸如指令是否正在等待断言位被计算,以及当断言位是真还是假时是否执行指令),操作数对指令的可用性,或者在发出和执行相关联的个体指令之前所需的其他先决条件。在每个指令窗口中存储的指令的数目通常与指令块内的指令的数目相对应。在一些示例中,在一个或多个广播信道上接收操作数和/或断言,该一个或多个广播信道允许将相同的操作数或断言发送到大量的指令。在一些示例中,指令块内的指令的数目可以是32、64、128、1024、或另一数字。在所公开的技术的一些示例中,指令块跨处理器核内的多个指令窗口分配。可以根据指定一个或多个操作模式的数据来控制乱序操作和存储器访问。

[0064] 在一些示例中,对处理器施加约束(例如,根据体系架构定义或通过处理器的可编程配置),以禁止不按照其中指令被布置在指令块中的顺序来执行指令。在一些示例中,可获取的编号最小的指令被配置为要被执行的下一指令。在一些示例中,控制逻辑遍历指令块中的指令并且执行准备执行的下一指令。在一些示例中,一次只能发出和/或执行一条指令。在一些示例中,指令块内的指令以确定性顺序(例如,按照指令被布置在块中的顺序次序)被发出和执行。在一些示例中,当通过用户使用软件调试器来调试在基于块的处理器上执行的程序时,可以配置对指令排序的约束。

[0065] 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协调从存储器取回指令、译码指令、一旦指令被加载到相应的指令窗口时对指令的执行、流入/流出处理器核111的数据、以及控制由处理器核输入和输出的信号。例如,控制单元205可以包括如上文所描述的就绪队列,用于调度指令。存储在位于每个相应的指令窗口210和211中的存储器存储装置215和216中的指令可以原子地执行。因此,对受执行的指令影响的

可见体系架构状态(诸如寄存器文件230和存储器)的更新可以被本地缓冲在核200内,直到指令被提交为止。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,并且发出提交信号。例如,用于指令块的提交阶段可以当所有寄存器写入被缓存时开始,对存储器的所有写入都被缓存,并且计算分支目标。指令块可以当对可见体系架构状态的更新完成时提交。例如,当寄存器写入作为寄存器文件被写入时,可以提交指令块,存储被发送到加载/存储单元或存储器控制器,并且生成提交信号。控制单元205还至少部分地控制功能单元260到相应指令窗口中的每个指令窗口的分配。

[0066] 如图2所示,具有若干个执行流水线寄存器255的第一路由器250被用来将数据从指令窗口210和211中的任一个指令窗口发送到功能单元260中的一个或多个功能单元,该功能单元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)或可以包括图形功能、物理功能和其他数学操作在内的其他合适的执行单元。然后,来自功能单元260的数据可以依据正在执行的特定指令的要求,通过第二路由器270被路由到输出290,291和292、路由回到操作数缓冲器(例如,LOP缓冲器242和/或ROP缓冲器243)、或者被馈送回到另一功能单元。第二路由器270可以包括:加载/存储队列275,其可以被用来发出存储器指令;数据高速缓存277,其存储正从核输入存储器或从核输出到存储器的数据;以及加载/存储流水线寄存器278。

[0067] 核还包括控制输出295,其被用来指示例如对于指令窗口210或211中的一个或多个指令窗口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“已提交”,并且来自控制输出295的信号可以进而被基于块的处理器100内的其他核和/或被控制单元160用于发起调度、取回和执行其他指令块。第一路由器250和第二路由器270二者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)。在一些示例中,指示相对排序和执行状态的数据被用来确定指令块是否可以被提交。

[0068] 如相关领域的普通技术人员将容易理解的,个体核200内的部件不限于图2中所示的那些,但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,单个指令译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取决于基于块的处理器的特定目标应用而变化。在利用指令核选择和分配资源时需要考虑的其他因素包括性能要求、能量使用要求、集成电路芯片、处理技术和/或成本。

[0069] 对于相关领域的普通技术人员而言,容易理解的是,可以通过在指令窗口(例如,指令窗口210)和处理器核110的控制单元205内的资源的设计和分配来在处理器性能中进行折衷。面积、时钟周期、能力和限制实质上确定了各个核110的实现性能和基于块的处理器100的吞吐量。

[0070] 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,指令调度器是高并发的。例如,每个周期,一个或多个译码器将指令的译码就绪状态和译码指令写入一个或多个指令窗口,选择要发出的下一指令,并且作为响应,后端发送就绪事件——以特定指令的输入槽(断言、左操作数、右操作数等)为目标的目标就绪事件或以所有指令为目标的的广播就绪事件。每条指令就绪状态位与译码就绪状态一起可以被用来确定指令已准备好发出。

[0071] 在一些情况下,调度器206接受目标指令的事件,其尚未被译码,并且还必须禁止

已发出的就绪指令的重发。在一些示例中，指令可以是非断言的或断言的(基于真条件或假条件)。断言的指令直到其被另一指令的断言结果作为目标并且该结果与断言条件相匹配时，才会变得就绪。如果相关联的断言不匹配，则指令永远不会发出。在一些示例中，可以推测地发出和执行断言指令。在一些示例中，处理器可以随后检查被推测地发出和执行的指令是否被正确地推测。在一些示例中，可能重新执行误推测的发出的指令和消耗其输出的块中的指令的特定传递闭包，或者误推测的副作用被取消。在一些示例中，对误推测的指令的发现导致整个指令块的完全回滚和重新执行。

[0072] 在分支到新指令块时，一个或多个相应的指令窗口就绪状态被清除(块重置)。然而，当指令块分支回自己(块刷新)时，只有活动就绪状态被清除。因此，可以保留指令块的译码就绪状态，使得不需要重新取回和译码该块的指令。因此，块刷新可以被用来节省循环中的时间和能量。

[0073] V.示例指令块流

[0074] 现在转到图3的图300，图示了基于块的指令流的一部分310，包括若干个可变长度指令块311至314。指令流可以被用来实现用户应用、系统服务或任何其他合适的用途。指令流可以被存储在存储器中，从存储器中的另一进程接收，通过网络连接接收，或以任何其他合适的方式存储或接收。在图3所示的示例中，每个指令块以指令头部开始，然后是不同数目的指令。例如，指令块311包括头部320和二十条指令321。所图示的特定指令头部320包括部分地控制指令块内的指令的执行的若干个数据字段，并且还允许改进的性能增强技术，其包括例如分支预测、推测执行、惰性评估和/或其他技术。指令头部320还包括指令块大小的指示。指令块的大小可以处于多于一个的指令的组块中，例如，指令块内包含的数目为4的指令组块。换句话说，为了压缩被分配以指定指令块大小的头部空间，块的大小被移位4位。因此，0的大小值指示尺寸最小的指令块，其是块头部，后面是四条指令。在一些示例中，指令块大小被表达为若干个字节、若干个字、若干个n字数据块、地址、地址偏移量、或者使用用于描述指令块大小的其他合适的表达式来表达。在一些示例中，指令块大小由指令块头部和/或脚部中的终止位模式指示。

[0075] 指令块头部320还可以包括指示用于执行指令块的一个或多个操作模式的一个或多个执行标志。例如，操作模式可以包括核融合操作、矢量模式操作、存储器依存性断定、和/或按序或确定性指令执行。

[0076] 在所公开的技术的一些示例中，指令头部320包括指示编码数据是指令头部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，不同的位编码可以被用于一个或多个标识位。在一些示例中，指令头部320包括指示相关联的指令块被编码所针对的ISA的特定版本的信息。

[0077] 块指令头部还可以包括用于例如分支断定、控制流确定和/或分支处理的若干个块退出类型。退出类型可以指示分支指令的类型是什么，例如:顺序分支指令，其指向存储器中的下一连续指令块;偏移指令，其在相对于偏移量计算的存储器地址处的另一指令块的分支;子例程调用或子例程返回。通过编码指令头部中的分支退出类型，分支预测器可以在相同指令块内的分支指令已被取回和/或译码之前至少部分地开始操作。

[0078] 所图示的指令块头部320还包括存储掩码，其指示在块指令中编码的加载存储队

列标识符中的哪个加载存储队列标识符被指派给存储操作。指令块头部还可以包括写入掩码，其标识相关联的指令块将写入哪个或哪些全局寄存器。在一些示例中，存储掩码通过例如指令译码器(例如，译码器228或229)被存储在存储矢量寄存器中。在其他示例中，指令块头部320不包括存储掩码，但是存储掩码由指令译码器当指令块被译码时通过分析指令依存性而动态生成。例如，译码器可以生成用于指令块指令的加载存储标识符以确定存储掩码并且将存储掩码数据存储在存储矢量寄存器中。类似地，在其他示例中，写入掩码没有被编码在指令块头部中，而是由指令译码器动态生成(例如，通过分析由指令块中的指令引用的寄存器)并且被存储在写入掩码寄存器中。写入掩码可以被用来确定何时指令块的执行已完成，从而发起指令块的提交。在指令块可以完成之前，相关联的寄存器文件必须接收对每个条目的写入。在一些示例中，基于块的处理器体系架构不仅可以包括标量指令，而且还可以包括单指令多数据(SIMD)指令，其允许在单个指令内用更大数目的数据操作数进行操作。

[0079] 可以用于指令321的合适的基于块的指令的示例可以包括用于执行整数和浮点算术运算、逻辑运算、类型转换、寄存器读取和写入、存储器加载和存储、分支和跳转执行的指令，以及其他合适的处理器指令。在一些示例中，指令包括用于配置处理器以通过例如基于控制流和关于存储器访问指令的数据的推测性执行根据操作中的一个或多个操作进行操作的指令，该存储器访问指令被存储在诸如存储指令数据存储装置207之类的硬件结构中。在一些示例中，存储指令数据存储装置207在体系架构上不可见。在一些示例中，对存储指令数据存储装置207的访问被配置为限于在处理器的监督模式或其他受保护模式下的处理器操作。

[0080] VI.示例块指令目标编码

[0081] 图4是描绘C语言源代码的两个部分410和415及其相应的指令块420和425的示例的图400，其图示了基于块的指令如何可以显式地编码其目标。在这个示例中，前两个READ指令430和431相应地以ADD指令432的右(T[2R])操作数和左(T[2L])操作数为目标(2R指示以指令编号2的右操作数为目标；2L指示指令编号2的左操作数)。在所图示的ISA中，读取指令是从全局寄存器文件(例如，寄存器文件230)读取的唯一指令；然而，任何指令都可以以全局寄存器文件为目标。当ADD指令432接收到两个寄存器读取的结果时，它将变成就绪并且执行。应当指出，本公开有时将右操作数称为OP0，将左操作数称为OP1。

[0082] 当TLEI(测试小于等于立即(test-less-than-equal-immediate))指令433从ADD接收其单个输入操作数时，它将准备好发出并且执行。然后，测试产生断言操作数，其在信道一(B[1P])上向在用于断言的广播信道上监听的所有指令广播，这些指令在该示例中是两个断言的分支指令(BRO_T 434和BRO_F 435)。接收匹配断言的分支指令将激发(执行)，但是用互补断言编码的另一指令不会激发/执行。

[0083] 还图示了用于指令块420的依存性图440，作为指令节点阵列450及其对应的操作数目标455和456。这图示了块指令420、对应的指令窗口条目和由指令表示的底层数据流图之间的对应关系。这里译码指令READ 430和READ 431已经准备好发出，因为它们不具有输入依存性。当它们发出并且执行时，从寄存器R0和R7读取的值被写入到ADD432的右操作数缓冲器和左操作数缓冲器中，从而将ADD432的左操作数和右操作数标记为“就绪”。结果，ADD 432指令变为就绪，发出到ALU，执行，并且其和被写入到TLEI指令433的左操作数。

[0084] VII.示例基于块的指令格式

[0085] 图5是图示了指令头部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)的指令格式的通用示例的图。指令格式可以用于根据在指定操作模式的指令头部中指定的若干个执行标志而被执行的指令块。指令头部或指令中的每个指令头部或指令均根据位数而被标记。例如,指令头部510包括四个32位字并且从其最低有效位(lsb)(位0)直到其最高有效位(msb)(位127)被标记。如所示出的,指令头部包括写入掩码字段、若干个退出类型字段、若干个执行标志字段、指令块大小字段和指令头部ID位(指令头部的最低有效位)。在一些示例中,指令头部510包括元数据515和/或516,其可以被用来存储用于指示目标操作数的附加信息。在一些示例中,若干个目标操作数被编码在指令头部510中并且与标识符相关联(例如,基于指令的位置的指令标识符,该指令生成的结果被用于为可变数目的目标指令生成目标操作数)。在这样的实施例中,可以通过在头部中编码这样的目标操作数和指令信息并且引用该头部来避免使用可变长度指令。在一些示例中,元数据是指相关联的源指令。

[0086] 在图5中描绘的执行标志字段占据指令块头部510的6至13位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融合操作、矢量模式操作、分支断定器抑制、存储器依存性断定器禁止、块同步、块之后中断、块之前中断、块落空失败、和/或按序或确定性指令执行。

[0087] 退出类型字段包括可以被用来指示在指令块内被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各项中的一项或多项: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包括相对和/或绝对地址的指令块之间传送控制流并且使用条件断言或无条件断言的任何控制流指令。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断定和推测执行。

[0088] 所图示的通用块指令520被存储为一个32位的字并且包括操作码字段、断言字段、广播ID字段(BID)、矢量操作字段(V)、单指令多数据(SIMD)字段、第一目标字段(T1)和第二目标字段(T2)。对于具有比目标字段更大的消费方的指令,编译器可以使用移动指令构建扇出树,它可以将高扇出指令指派给广播。广播支持通过轻量级网络将操作数发送到核中的任意数目的消费方指令。

[0089] 尽管由通用指令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全部指令,但是本领域技术人员将容易理解,即使对于ISA的特定示例,指令字段中的一个或多个指令字段可能偏离用于特定指令的通用格式。操作码字段指定由指令520执行的一个或多个操作,诸如存储器读取/写入、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。断言字段指定指令在其下执行的条件。例如,断言字段可以指定值“真”,并且只有当对应的条件标志匹配指定的断言值时,才会执行该指令。在一些示例中,断言字段至少部分地指定哪个用于比较断言,而在其他示例中,执行在由先前指令(例如,指令块中的先前指令)设置的标志上被断定。在一些示例中,断言字段可以指定该指令总是或永远不会被执行。因此,通过减少分支指令的数目,使用断言字段可以实现更密集的目标代码,提高能量效率,并且改善处理器性能。

[0090] 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，其包括指定操作数槽(例如，左操作数、右操作数或断言操作数)。取决于特定指令和ISA，所图示的目标字段中的一个或两个目标字段可以被其他信息取代，例如，第一目标字段T1可以被立即操作数、附加操作码、指定两个目标等来取代。

[0091] 分支指令530包括操作码字段、断言字段、广播ID字段(BID)和偏移字段。操作码字段和断言字段在如关于通用指令中描述的格式和功能方面是类似的。偏移可以以四个指令组为单位被表达，从而扩展了在其上可以执行分支的存储器地址范围。利用通用指令520和分支指令530所示的断言可以被用来避免指令块内的附加分支。例如，可以根据先前指令的结果(例如，两个操作数的比较)来断定特定指令的执行。如果断言为假，则指令不会提交由特定指令计算的值。如果断言值与所需断言不匹配，则指令不会发出。例如，BRO_F(断言假)指令将发出，如果其被发送假断言值。

[0092] 应当容易理解的是，如本文中所使用的，术语“分支指令”不限于将程序执行改变为相对存储器位置，而是还包括跳转到绝对或符号存储器位置、子例程调用和返回以及可以修改执行流的其他指令。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，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处的值来改变执行流。在一些示例中，跳转寄存器分支指令被用来跳转到存储在寄存器中的存储器位置。在一些示例中，子例程调用和返回分别使用跳转和链接以及跳转寄存器指令来实现。

[0093] 存储器访问指令540格式包括操作码字段、断言字段、广播ID字段(BID)、直接字段(IMM)偏移字段和目标字段。操作码、广播、断言字段在如关于通用指令所描述的格式和功能方面类似。例如，可以根据先前指令的结果(例如，两个操作数的比较)来断言特定指令的执行。如果断言为假，则指令不会提交由特定指令计算的值。如果断言值与所需断言不匹配，则指令不会发出。直接字段(例如，并且移位若干位)可以用作发送到加载或存储指令的操作数的偏移量。操作数加(移位的)立即偏移被用作用于加载/存储指令(例如，从存储器读取数据或将数据存储到存储器中的地址)的存储器地址。

[0094] VIII.示例处理器状态图

[0095] 图6是图示当指令块被映射、执行和引退时指派给指令块的状态的数目的状态图600。例如，根据一个或多个执行标志，在执行指令期间可以指派状态中的一个或多个状态。应当容易理解的是，图6所示的状态是所公开的技术的状态的示例，但是在其他示例中，指令块可以具有附加的或更少的状态，并且具有与状态图600中所描绘的状态不同的状态。在状态605处，指令块未被映射。指令块可以驻留在耦合到基于块的处理器的存储器中，存储在诸如硬盘驱动器或闪存驱动器之类的计算机可读存储设备上，并且可以位于处理器本地或者位于远程服务器处并且可以使用计算机网络来访问。未映射的指令也可以至少部分地驻留在耦合到基于块的处理器的高速缓存存储器中。

[0096] 在指令块映射状态610处，诸如指令调度器之类的基于块的处理器的控制逻辑可以被用来监视基于块的处理器的处理核资源，并且将指令块映射到一个或多个处理核。

[0097] 控制单元可以将指令块中的一个或多个指令块映射到特定处理器核的处理器核和/或指令窗口。在一些示例中，控制单元监视先前已经执行特定指令块的处理器核，并且可以针对仍驻留在“已预热”处理器核上的指令块重新使用经译码的指令。一旦一个或多个

指令块已被映射到处理器核，指令块就可以前进到取回状态620。

[0098] 当指令块处于取回状态620(例如，取指)时，映射的处理器核从基于块的处理器的存储器系统取回计算机可读块指令，并且将它们加载到与特定处理器核相关联的存储器中。例如，针对指令块的经取回的指令可以被取回并且被存储在处理器核内的指令高速缓存中。可以使用核互连将指令传达到处理器核。一旦指令块的至少一条指令已经被取回，指令块就可以进入指令译码状态630。

[0099] 在指令译码状态630期间，取出的指令的各个位被译码成可以由处理器核用来控制特定指令的执行的信号，其包括生成指示存储器访问指令的相对排序的标识符。例如，经译码的指令可以被存储在上文在图2中所示出的存储器存储装置215或216中的一个存储器存储装置中，译码包括生成译码的指令的依存性、用于译码的指令的操作数信息以及译码的指令的目标。一旦指令块的至少一条指令已经被译码，指令块就可以进入执行状态640。

[0100] 在执行状态640期间，使用例如以上关于图2所讨论的功能单元260来执行与该指令相关联的操作。如上文所讨论的，所执行的功能可以包括算术功能、逻辑功能、分支指令、存储器操作和寄存器操作。与处理器核关联的控制逻辑监视指令块的执行，并且一旦确定指令块可以被提交，或者指令块将被中止，则指令块状态被设置为提交/中止650。在一些示例中，控制逻辑针对指令块使用写入掩码和/或存储掩码来确定执行是否已经充分进行以提交指令块。

[0101] 在提交/中止状态650处，处理器核控制单元确定可以完成由指令块执行的操作。例如，存储器加载存储操作、寄存器读取/写入、分支指令和其他指令将根据指令块的控制流程被明确地执行。可替代地，如果指令块将被中止，例如，因为指令的依存性中的一个或多个依存性不被满足，或者在针对不被满足的指令块的断言上该指令被推测性地执行，所以指令块被中止，使得不会影响存储器或寄存器文件中的指令序列的状态。无论指令块是否已经提交或中止，指令块进入状态660以确定是否应当刷新指令块。如果指令块被刷新，则处理器核通常使用新数据值(具体地，由刚刚提交的块的执行更新的寄存器和存储器)来重新执行指令块，并且直接前进到执行状态640。因此，可以避免在映射、读取和译码指令块中花费的时间和精力。可替代地，如果指令块没有要被刷新，则指令块进入空闲状态670。

[0102] 在空闲状态670中，执行指令块的处理器核可以通过例如对处理器核内的硬件断电，同时维持指令块的译码指令的至少一部分而被空闲。在某个时刻，控制单元确定680处理器核上的空闲指令块是否要被刷新。如果空闲指令块要被刷新，则指令块可以在执行状态640处恢复执行。可替代地，如果指令块不要被刷新，则指令块未被映射，并且处理器核可以被冲刷并且随后指令块可以被映射到经冲刷后的处理器核。

[0103] 虽然状态图600图示了为了便于解释而在单个处理器核上执行的指令块的状态，但是相关领域的普通技术人员应当容易理解，在某些示例中，多个处理器核可以被用来同时执行给定指令块的多个实例。

[0104] IX.可变数目的目标操作数的示例指令编码

[0105] 图7是图示了可以用于基于块的处理器的各种基于块的指令的可变长度指令格式的通用示例的图700。在一些示例中，指令操作码710与指令的固定长度版本的操作码相同，而在其他示例中，不同的操作码用于用信号通知该指令具有可变长度。所描绘的指令具有两个目标操作数字段T1和T2，这两个目标操作数字段T1和T2指定用于发送来自执行由该指

令指定的操作的结果的目标操作数。在所图示的示例中，每个目标操作数使用9位格式进行编码，这将在下文关于图8进一步讨论，该9位格式包括目标指令(或广播ID)和目标类型(例如，左操作数、右操作数或断言操作数)的指示。图7所图示的指令是寄存器READ指令，并且包括用于指示指令的执行是否被断定的断言字段(PR)、用于指示从中读取值的源操作数寄存器的通用寄存器字段(GR)、以及两个目标操作数T1和T2。因为指令是可变长度指令，所以在包含操作码710的字715之后的指令数据的后续字(例如，字720和725)中可以指定附加的目标操作数。在可变长度指令使用相同的操作码作为固定长度指令的示例中，在下一字开头编码的扩展操作码(REXT)指示该指令是长度可变的。该第二字720包括三个目标操作数：T3、T4和T5。可变长度指令可以被扩展为在第三字中包括另外的目标操作数(725)，如所指示的：T6、T7和T8。因此，对于由多于两个目标指令或其他目标使用的指令产生的结果，单个指令可以被用来生成目标操作数。这对于基于块的处理器体系架构尤其有利，其中每个指令块作为单个事务执行，并且在指令块的执行的单个实例中执行的不同指令之间，寄存器值的改变是不可见的。因此，传统的RISC和CISC体系架构中不存在需要两个以上目标操作数的问题。

[0106] 图8是描绘了如可以在所公开的技术的某些示例中使用的用于编码目标操作数的示例格式的图800。如所示出的，目标格式的位7和8被用于对目标的一般类型进行编码，其可以是通用寄存器写入(到全局寄存器文件)、广播信道(其可以由指令块中的任意数目的指令读取)、针对由INST ID字段指定的特定指令的断言槽、或者由INSTID字段指定的特定指令的分别被称为OP0和OP1的左操作数槽或右操作数槽。因此，单个指令可以被编码以多于一种类型的目标操作数为目标：全局寄存器、经由广播或指令标识符的指令数据操作数、或者经由广播或指令标识符的指令断言。图7中描绘的可变长度指令格式可以例如被编码以如下为目标：具有目标操作数T1和T2的通用寄存器、具有目标操作数T3的广播信道以及以目标操作数T4和T5为目标的特定指令(断言、左操作数或右操作数)。

[0107] 图9是图示了如可以在本公开技术的某些示例中使用的可变长度指令格式900的另一示例的图。如所示出的，该指令包括操作码910，其可以是被用于相同指令的固定长度版本的相同操作码，或者可以是不同的操作码，这取决于正在采用的特定处理器体系架构。类似于图7的可变长度指令，该指令包括断言字段和通用寄存器字段。下一字段NLOP指示数据将从指令发送到的若干个左目标操作数：LOP1，LOP2和LOP3。操作数中的每个操作数都使用7位进行编码以指示指令标识符，但不包括用于指示字段类型的额外位，因为这是基于指令的NLOP字段。类似地，该指令也正在针对两个右操作数，其将针对字段NROP被编码为整数值2，随后是要将目标操作数ROP1和ROP2发送到的指令标识符。最后，该指令包括NPRED字段，其指示通过该指令被写入的断言目标操作数的数目。在这个示例中，断言目标操作数的数目是4。进一步地，如所示出的，因为位数与32位字大小没有完美地对齐，所以在指令的第二字和第三字的末尾处的一些位未被使用。

[0108] 图9还图示了如可以在所公开的技术的某些示例中使用的可变长度指令格式950的另一个示例。在950的可变长度指令示例中，操作码960用于加法立即指令(ADDI)，而非寄存器读取指令。因此，显然所公开的可变长度指令格式不限于寄存器读取，而是可以包括其他合适的指令。在所图示的指令格式950中，该指令编码三个左操作数：LOP1、LOP2和LOP3，但不包括任何右目标操作数或断言目标操作数，因此ROP字段和PRED字段被设置为零。

[0109] 图10图示了可以使用矢量格式向多个目标指令发送指令数据的示例指令格式1000。该指令包括操作码1010，其可以是与对应的固定长度指令相同的操作码，或者在其他示例中是不同的操作码。该指令还包括位17和16处的操作数类型(OT)字段，其指示指令将发送数据的一个或多个目标操作数的类型。在所图示的示例中，OT字段被编码以指示右操作数。其他指令可以用OT字段来编码，其指示断言、左操作数或其他合适的目标操作数类型。该指令还包括矢量字段1020，其包括16位，每个位与要将目标操作数数据发送到的目标指令相关联。图10还图示了示例指令块1050，其包括如虚线所指示的被分组为16位组块的指令头部1055和128条指令1057。示例指令被包括在第三个16位组块1060的第二字中。所图示的指令是来自寄存器编号5的寄存器读取指令，并且OT字段被设置为ROP。因此，寄存器5的值将从寄存器文件中被读取并且被发送到由矢量指示的每个指令。在所图示的示例中，矢量是0000100001 101000。因此，寄存器5的值将被发送到指令编号4、指令编号9、指令编号10和指令编号12。因此，多于两个指令可以使用所图示的指令格式以单个指令为目标。然而，该指令是受限制的，因为仅组块内的指令才可以以可变长度目标指令为目标(例如，仅具有第三指令组块1060的指令才包含具有目标操作数矢量的指令)。

[0110] 在备选示例中，矢量格式指令被重新编码，使得用于矢量的16位可以与OT字段中使用的位组合。在这种情况下，指令块可能被布置在6位的组块中，并且每个类型最多6条指令(断言、左操作数、右操作数)可以从单个矢量指令接收数据。在一些示例中，指令被布置为使得断言目标指令从矢量目标指令偏移一固定量，右操作数目标指令偏移额外的6个指令，并且左操作数目标指令偏移额外的6个指令，从而允许矢量格式将结果发送到大量的目标指令。

[0111] 图11图示了如可以在所公开的技术的某些示例中实现的目标矢量指令的备选实现方式。在所图示的示例中，指令块1110的16位指令组块内的指令的相对位置确定指令矢量将发送哪个组块的目标操作数。因此，第一指令1120位于指令块1110的第三组块1130的第二字中。因此，与指令矢量相关联的四个指令将被发送到位于第二组块内的指令。类似地，第八指令1125也是目标矢量指令，并且因为它位于第八字处，所以由读取指令(这里是寄存器8的读取)产生的所有值将被发送到由指令块的指令的第八组块1135内的矢量所指示的各个指令。因此，可以至少部分地基于指令块的指令组块内的指令的位置来将指令作为目标。

[0112] X.示例基于块的处理器和存储器配置

[0113] 图12是图示了包括基于块的处理器1210的装置的图1200，该基于块的处理器包括被配置为执行包括具有可变数目的目标操作数的指令的指令块的控制单元1220。控制单元1220包括核调度器。核调度器1225调度指令块的流程，其包括用于执行指令处理的核的分配和解除分配、对核中的任何核、寄存器文件、存储器接口和/或I/O接口之间的输入数据和输出数据的控制。

[0114] 基于块的处理器1210还包括被配置为取回并且执行指令块的一个或多个处理器核1240至747。这些核中的每个核包括指令译码器(例如，译码器1249)，其对指令操作码、扩展操作码和其他字段进行译码，以确定指令是否指定可变数目和/或目标操作数的类似。所图示的基于块的处理器1210具有多达8个核，但是在其他示例中，可以具有64个、512个、1024个或其他数目个的基于块的处理器核。基于块的处理器1210耦合到存储器1250，该存

储器1250包括若干个指令块，这些指令块包括指令块A和B，该指令块A和B包括具有可变数目的目标操作数的指令(1255和1256)；并且耦合到计算机可读存储媒体盘1260，其存储具有可变数目的目标操作数的指令1265。

[0115] XI.用变量目标操作数译码和执行指令的示例方法

[0116] 图13是如可以在本公开技术的某些示例中执行的概述了译码和执行具有可变数目的目标操作数的指令的示例方法的流程图1300。例如，图1的基于块的处理器包括上文在图2中描述的基于块的处理器核111，可以被用来执行所概述的方法。在一些示例中，基于块的处理器的指令译码器被配置为对具有可变数目的目标操作数的指令进行译码并且执行经译码的指令，其包括将数据发送到指定的目标操作数进而发送到指定的目标指令。

[0117] 在过程框1310处，来自基于块的处理器指令块的一个或多个指令被译码。经译码的指令中的至少一个经译码的指令指定用于发送与相应指令相关联的操作的结果的两个或更多个目标。取决于所采用的特定指令格式，相同的指令格式可以被用来指定一个目标、两个目标、三个目标或更多个目标。

[0118] 在过程框1320处，通过执行由指令操作码指定的操作并且通过将结果发送到在过程框1310处经译码的指定目标来执行经译码的指令。在一些示例中，可变数目的目标操作数可以通过使用指定的操作码、指定目标操作数的数目和/或类型的字段、随后是指示目标指令的若干个标志符来指定。在其他示例中，目标指令使用目标矢量用多个目标操作数进行编码，该目标矢量包括指定每条指令的位以发送由指令执行的操作的结果。在一些示例中，至少部分地基于译码的指令在指令块内的相对位置来标识将向其发送目标操作数目标指令。在一些示例中，指令编码多个目标操作数，但是多个目标操作数全部用于相同类型，例如，断言、左操作数或右操作数。在其他示例中，指定的目标操作数可以用于多种类型。在一些示例中，具有多个目标的指令是寄存器读取指令，而在其他示例中，处理器允许产生值的任何操作具有可变数目的目标操作数。在一些示例中，控制单元(例如，控制单元205)包括硬件结构，该硬件结构可以在连续数目的时钟周期中缓冲目标操作数中的一个或多个目标操作数。在一些示例中，每个指令包括可以存储其相应的左操作数、右操作数和断言操作数直到指令发出和执行为止的缓冲器。在其他示例中，控制单元的硬件结构中的缓冲器存储目标操作数，直到一个或多个接收目标指令准备好发出并且执行为止。

[0119] XII.示例源代码和对象代码

[0120] 图14A至图14B图示了如可以用于本公开技术的某些示例中的用于基于块的处理器的源代码1410和对应的汇编代码1420和1430的示例。源代码1410包括if/else语句。if/else语句的每个部分内的语句包括对数组a[]和b[]的若干个存储器读取和存储器写入。当源代码1410被转换为汇编代码(并且随后，被存储为目标代码的机器代码)时，将生成若干个加载和存储汇编指令。

[0121] 用于源代码部分1410的汇编代码1420包括编号为0到27的28个指令。汇编指令不包括可变目标操作数，因此没有指令可以以多于两个的目标指令为目标。在一些情况下，广播信道(例如，B1R和B2P)被用来向多于一个或两个目标指令发送值。汇编代码指示若干个字段，例如，指令操作码、由指令指定的源数据(例如，广播标识符或立即参数)、以及目标指定。汇编代码包括寄存器读取指令(0至3)、算术指令(例如，指令3和4)以及向多个目标(例如，移动指令5和6)发送数据的移动指令。汇编代码1420还包括测试指令11，该测试指令11

测试是否大于将在广播信道2(B2P)上生成断言值的指令。进一步地,汇编代码包括两个不可断言的存储器加载指令7和8以及断言的存储器加载指令16和23。汇编代码1420还包括若干个存储器存储指令,其将数据存储到存储器地址,例如,断言的存储指令12,13,18和26。

[0122] 除了使用广播信道之外,汇编代码还使用MOV指令复制值。例如,LD指令7执行从地址a[i]的存储器读取,并且将该值发送到MOV指令8。然后,该值被复制到指令13、左操作数以及指令11、左操作数。这个额外的MOV指令被使用,因为32位固定长度的指令格式只允许用于LD指令的一个目标操作数。

[0123] 转到图14B,用于源代码部分1410的汇编代码部分1430已被重新编码以使用可变长度目标操作数格式。这使总指令数量减少了7条指令,并且仅为扩展的目标部分使用附加的两个指令代码字。指令ID没有被改变以便于与汇编代码部分1420进行比较,但是指令可以被压缩以占用较少的空间。例如,指令2已经被READV指令取代,该READV指令对两个左操作数目标(指令ID 15L和22L)和两个右操作数目标(指令ID 19R和24R)进行编码。上文关于图9所讨论的指令格式可以被用来使用单个指令和一个附加的指令数据字对这两种不同的目标操作数类型进行编码。在其他示例中,图8中讨论的READX指令可以被用来对指令2进行编码。汇编代码部分1420中所示的相关联的MOV指令14和21也已经被消除。

[0124] 指令3和11已经使用上文关于图10所讨论的矢量格式进行编码。应当指出,所有的操作数都是相同的类型,其将在指令的经编码的OT字段中被指定。这可以节省若干个指令,例如,指令11以具有七个目标指令的矢量格式被编码。重新编码的汇编代码部分1430也消除了广播信道的使用。在一些示例中,可以通过消除对广播信道的使用并且使用可变目标操作数指令和相关联的硬件来优化处理器核。进一步地,广播id位的使用可以被分配给其他功能,或者扩展在其他指令字段中所使用的位(例如,用于对立即值进行编码)。

[0125] 汇编代码部分1420和1430可以被转换为机器代码以用于由基于块的处理器的实际执行。例如,依据特定机器体系架构和编译器参数,汇编代码可以使用与上文关于图5所讨论的那些格式类似的固定长度指令格式和上文关于图7至图11所讨论的可变操作数指令格式进行编码。在一些示例中,可以基于性能相对代码密度目标来选择一些指令格式。

[0126] XIII.转换代码的示例方法

[0127] 图15是概述如在本公开技术的某些示例中可以执行的生成用于基于块的处理器的计算机可执行代码的示例方法的流程图1500。例如,图1的基于块的处理器包括上文在图2中描述的基于块的处理器核111,可以被用来执行所概述的方法。在其他示例中,通用RISC或CISC处理器被用来生成将由基于块的处理器执行的指令。在一些示例中,代码由编译器转换并且被存储为可以由基于块的处理器执行的对象代码。在一些示例中,即时编译器或解释器在运行时生成计算机可执行代码。

[0128] 在过程框1510处,分析源代码和/或目标代码以确定要由编译器或解释器接纳的指令块的操作数依存性。例如,编译器可以分析以确定相同的目标操作数将被发送到多个不同的指令。在一些示例中,结合源代码来分析诸如代码大小、执行速度、广播信道可用性、指令地址空间和其他合适的参数的参数,以确定操作数依存性。在分析操作数依存性之后,该方法继续进行到过程框1520。

[0129] 在过程框1520处,在过程框1510处分析的源代码和/或目标代码被转换成用于一个或多个指令块的计算机可执行代码,计算机可执行代码包括根据可变长度格式对目标操

作数进行编码的指令。在一些示例中，与上文关于图7所讨论的格式类似的指令格式通过指定用于每个目标操作数的指令标识符来对可变数目的目标操作数进行编码。在其他示例中，可变长度目标操作数使用与上文关于图10和图11所讨论的类似的矢量来指定。一旦代码已经被转换成基于块的处理器代码，其可以被存储在计算机可读存储介质中，或者经由计算机网络被传送到另一位置以供基于块的处理器执行。

[0130] 在方法的一些示例中，编译器将优先级指派给以具有可变数目的目标操作数的单个可变长度指令为目标的目标操作数。然后，编译器可以布置目标操作数的顺序，使得较高优先级的目标操作数在较低优先级的目标操作数之前被设置为其目标指令。例如，如果某些目标指令具有不同的延迟时间或其他执行要求，则可以相应地对其目标操作数进行排序，以便改进整个指令块的执行。在一些示例中，编译器对扩展指令和/或目标指令进行重新排序，使得用于发送多个目标操作数的指令的所有目标指令将位于可以由降序指令寻址的指令块的一部分内。在方法的一些示例中，该方法还包括：确定转换后的代码的指令块内的指令的位置，该位置将导致执行指令的处理器以至少部分地基于发送指令的位置来向指令组块发送可变数目的目标操作数。

[0131] 一旦代码已被转换成基于块的处理器代码，该代码可以被存储在计算机可读存储介质中，或者经由计算机网络被传送到另一位置以供基于块的处理器执行。

[0132] XIV.示例性计算环境

[0133] 图16图示了其中可以实现所描述的包括配置基于块的处理器的实施例、方法和技术的合适的计算环境1600的一般性示例。例如，如本文中所描述的，计算环境1600可以实现用于配置处理器以生成用于一个或多个指令块的相对排序数据的所公开的技术，或将代码编译成用于执行这些操作的计算机可执行指令。

[0134] 计算环境1600并不旨在对该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包括手持式设备、多处理器系统、可编程消费电子产品、网络PC、小型计算机、大型计算机等在内的其他计算机系统配置来实现。所公开的技术还可以在分布式计算环境中实践，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)可以位于本地存储器存储设备和远程存储器存储设备中。

[0135] 参考图16，计算环境1600包括至少一个基于块的处理单元1610和存储器1620。在图16中，这个最基本的配置1630被包括在虚线之内。基于块的处理单元1610执行计算机可执行指令，并且可以是实际处理器或虚拟处理器。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，如此，多个处理器可以同时运行。存储器1620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)或这两者的某个组合。存储器1620存储软件1680、图像和视频，其可以例如实现本文中所描述的技术。计算环境可能具有附加特征。例如，计算环境1600包括存储器1640、一个或多个输入设备1650、一个或多个输出设备1660以及一个或多个通信连接1670。诸如总线、控制器或网络之类的互连机构(未示出)使计算环境1600的组件互连。通常，操作系统软件(未示出)为在计算环境1600中执行的其他软件提供操作环境，并且协调计算环境1600的组件的活动。

[0136] 存储器1640可以是可移除的或不可移除的，并且包括磁盘、磁带或磁带盒、CD-ROM、CD-RW、DVD或可以被用来存储信息并且可以在计算环境1600内被访问的任何其他介

质。存储装置1640存储用于软件1680的指令、插件数据和消息，其被可以用来实现本文中所描述的技术。

[0137] 一个或多个输入设备1650可以是诸如键盘、小键盘、鼠标、触摸屏显示器、笔或轨迹球之类的触摸输入设备；话音输入设备；扫描设备；或向计算环境1600提供输入的另一设备。对于音频，一个或多个输入设备1650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，或者向计算环境1600提供音频样本的CD-ROM读取器。一个或多个输出设备1660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1600的输出的另一设备。

[0138] 一个或多个通信连接1670启用通过通信介质(例如，连接网络)到另一计算实体的通信。通信介质传送诸如计算机可执行指令、压缩图形信息、视频、或经调制的数据信号中的其他数据之类的信息。一个或多个通信连接1670不限于有线连接(例如，兆比特或千兆比特以太网、无线宽带、电连接或光纤连接上的光纤信道)，还包括无线技术(例如，经由蓝牙、WiFi(IEEE 802.11a/b/n)、WiMax、蜂窝、卫星、激光、红外的RF连接)和用于为所公开的方法提供网络连接的其他合适的通信连接。在虚拟主机环境中，一个或多个通信连接可以是由虚拟主机提供的虚拟化网络连接。

[0139] 所公开的方法的一些实施例可以使用在计算云1690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中，或者所公开的编译器可以在位于计算云1690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。

[0140] 计算机可读介质是可以在计算环境1600内被访问的任何可用介质。作为示例而非限制，利用计算环境1600，计算机可读介质包括存储器1620和/或存储装置1640。应当容易理解，术语计算机可读存储介质包括用于诸如存储器1620和存储装置1640之类的数据存储介质，而非诸如经调制的数据信号之类的传输介质。

[0141] XV.所公开的技术的附加示例

[0142] 根据上文所讨论的示例，本文中对所公开的主题的附加示例进行讨论。相关领域的普通技术人员参考以下附录A可以进一步理解附加示例。

[0143] 在所公开的技术的一些示例中，一种装置包括存储器和一个或多个基于块的处理器核。处理器核中的一个或多个处理器核包括指令译码器，其被配置为对指令块中的指令的目标操作数进行译码，该指令被编码以将可变数目的目标操作数考虑在内；以及控制单元，其被配置为发送用于由目标指令指定的操作的译码的目标操作数中的每个目标操作数的数据。

[0144] 在一些示例中，处理器核包括执行单元，其被配置为使用译码的操作数中的至少一个译码的操作数来执行操作。在一些示例中，该指令用指示用于目标操作数的指令标识符的可变数目的指令目标而被编码。在一些示例中，指令目标还指示目标操作数中的每个目标操作数的类型。在其他示例中，该指令用指示指令目标的位矢量进行编码。在一些示例中，一个或多个类型的目标操作数可以是以下各项中的一项或多项：指令块内的另一指令的标识符、全局寄存器、广播信道、另一指令块的标识符、以及存储器地址。在一些示例中，编码目标操作数的字段支持一种类型、多种类型或全部类型。在一些示例中，不同的指令操作码用于不同的操作数类型。在一些示例中，目标操作数中的至少一个目标操作数指示由目标指令执行的操作的断言操作数、右操作数或左操作数。

[0145] 在一些示例中，指令译码器被配置为至少部分地基于指令块内的指令的相对位置来确定接收目标操作数数据的目标指令。在一些示例中，指令译码器被配置为基于在指令块的不同字中被编码的扩展操作码来译码目标操作数。在一些示例中，指令译码器被配置为至少部分地基于在指令块中编码的指令的操作码来将指令译码为固定长度指令或可变数目的目标指令。

[0146] 在所公开技术的一些示例中，一种操作处理器以执行指令块的方法包括：译码来自指令块的一个或多个指令，译码的指令中的至少一个译码的指令指定用于发送与相应指令相关联的操作的结果的两个或更多个目标；通过执行操作来执行译码的指令；以及将结果发送到指定的两个或更多个目标。

[0147] 在一些示例中，译码包括：至少部分地基于经译码的指令在指令块内的相对位置或经译码的指令在指令块的一部分内的相对位置来确定所指定的两个或更多个目标。在一些示例中，使用经译码的指令的两个相对位置。

[0148] 在一些示例中，译码包括：至少部分地基于矢量来确定所指定的两个或更多个目标。在一些示例中，译码包括：译码指示具有特定类型的若干个目标操作数的数据，以及确定发送用于特定类型的目标操作数的数据的目标位置。

[0149] 在一些示例中，译码包括：确定目标指令操作数的类型。合适的操作数类型的示例包括左操作数、右操作数和断言操作数。在一些示例中，目标操作数类型可以以不同方式(例如，作为数字或字母)而被指定，或者包括不同类型(例如，用于整数与定点值的不同类型)。在一些示例中，指定了复杂的断言操作数，或者没有断言操作数被指定。在一些示例中，译码包括：确定目标指令操作数的类型，并且类型可以是以下各项中的任一项或多项：目标指令、广播信道、寄存器和存储器。在一些示例中，广播信道不是所支持的类型。在一些示例中，寄存器不是所支持的类型。在一些示例中，存储器不是所支持的类型。在一些示例中，目标指令不是所支持的类型。在一些示例中，目标指令是唯一支持的类型，或者是针对某些操作码类别唯一支持的类型。

[0150] 在一些示例中，该指令是寄存器读取。在一些示例中，附加指令可以支持可变数目的目标指令，例如，算术指令、逻辑指令和/或移动指令。

[0151] 在所公开的技术的一些示例中，一种编译用于基于块的处理器的源目标代码和/或链接目标代码的方法包括：分析源代码和/或目标代码以确定用于指令块的操作数依存性，并且将源代码和/或目标代码转换成用于指令块的计算机可执行代码，该计算机可执行代码包括根据可变长度格式对目标操作数进行编码的指令。在一些示例中，通过在基于块的处理器上执行计算机可读指令来执行该方法。在其他示例中，通过在通用处理器或专用处理器(例如，CISC或RISC CPU；或GPU或其他专用协处理器)上执行计算机可读指令来执行该方法。

[0152] 在一些示例中，编译方法包括：向目标操作数指派优先级并且安排目标操作数，以使较高优先级的目标操作数将在较低优先级的目标操作数之前被发送到其目标指令。在一些示例中，该方法包括：对目标指令进行重新排序，使得用于发送多个目标操作数的指令的所有目标指令将全部位于可以由发送指令寻址的指令块的一部分内。

[0153] 在一些示例中，该方法包括：确定转换后的代码的指令块内的指令的位置，该位置将使得执行指令的处理器以至少部分地基于该位置来向指令的组块发送可变目标操作数。

[0154] 在一些示例中，计算机可读存储介质存储用于指令块的计算机可读指令，该指令块当由处理器执行时，使得处理器执行所公开的方法中的任一个或多个方法。

[0155] 鉴于可以应用所公开的主题的原理的许多可能的实施例，应当认识到，所说明的实施例仅是优选示例，并且不应被视为将权利要求的范围限制为那些优选的示例。相反，所要求保护的主题的范围由以下权利要求限定。我们因此要求落入这些权利要求范围之类的全部内容作为本发明。

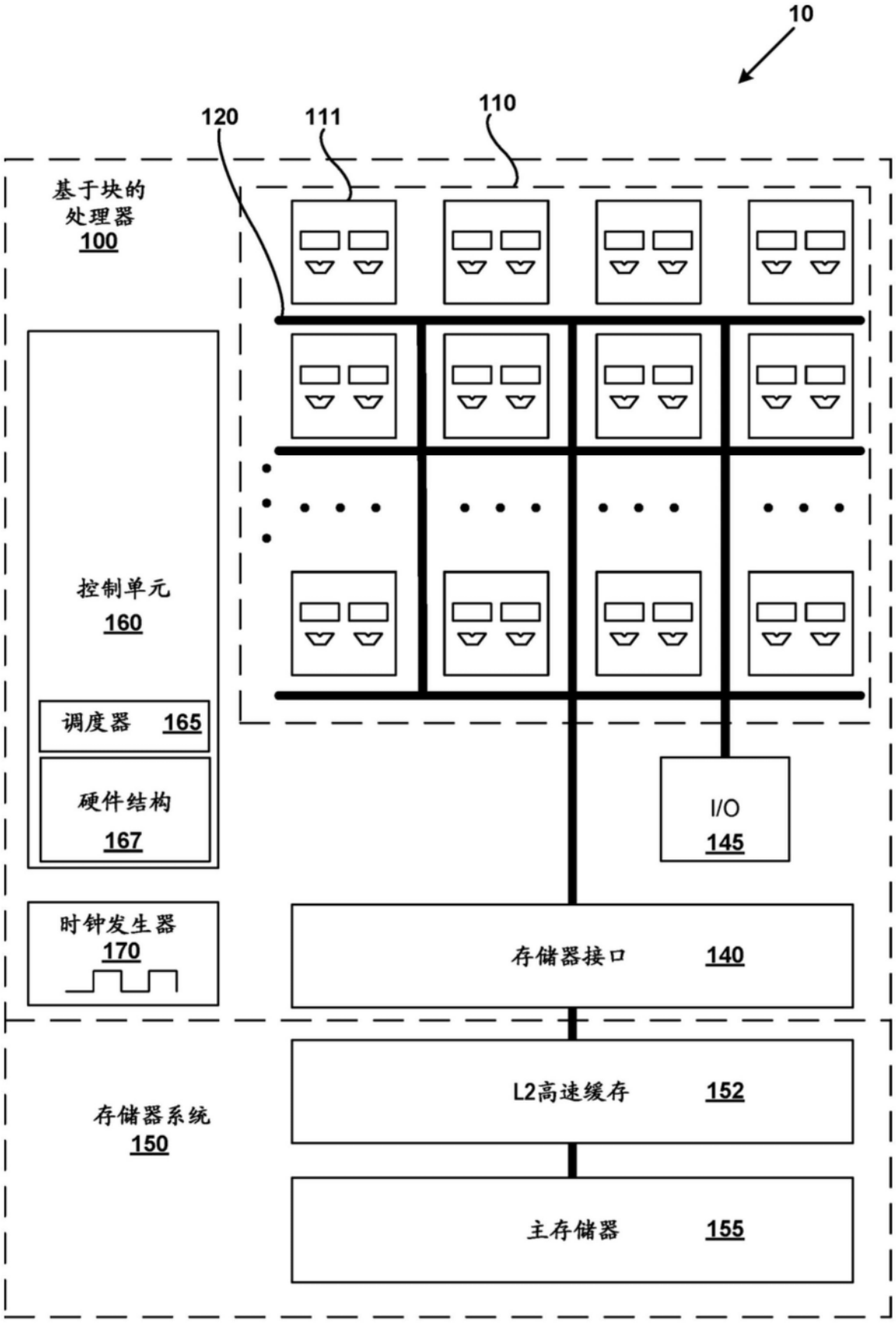

10
120
111
110
基于块的
处理器
100
控制单元
160
调度器 165
硬件结构
167
时钟发生器
170
I/O
145
存储器接口 140
存储器系统
150
L2高速缓存 152
主存储器 155

图1

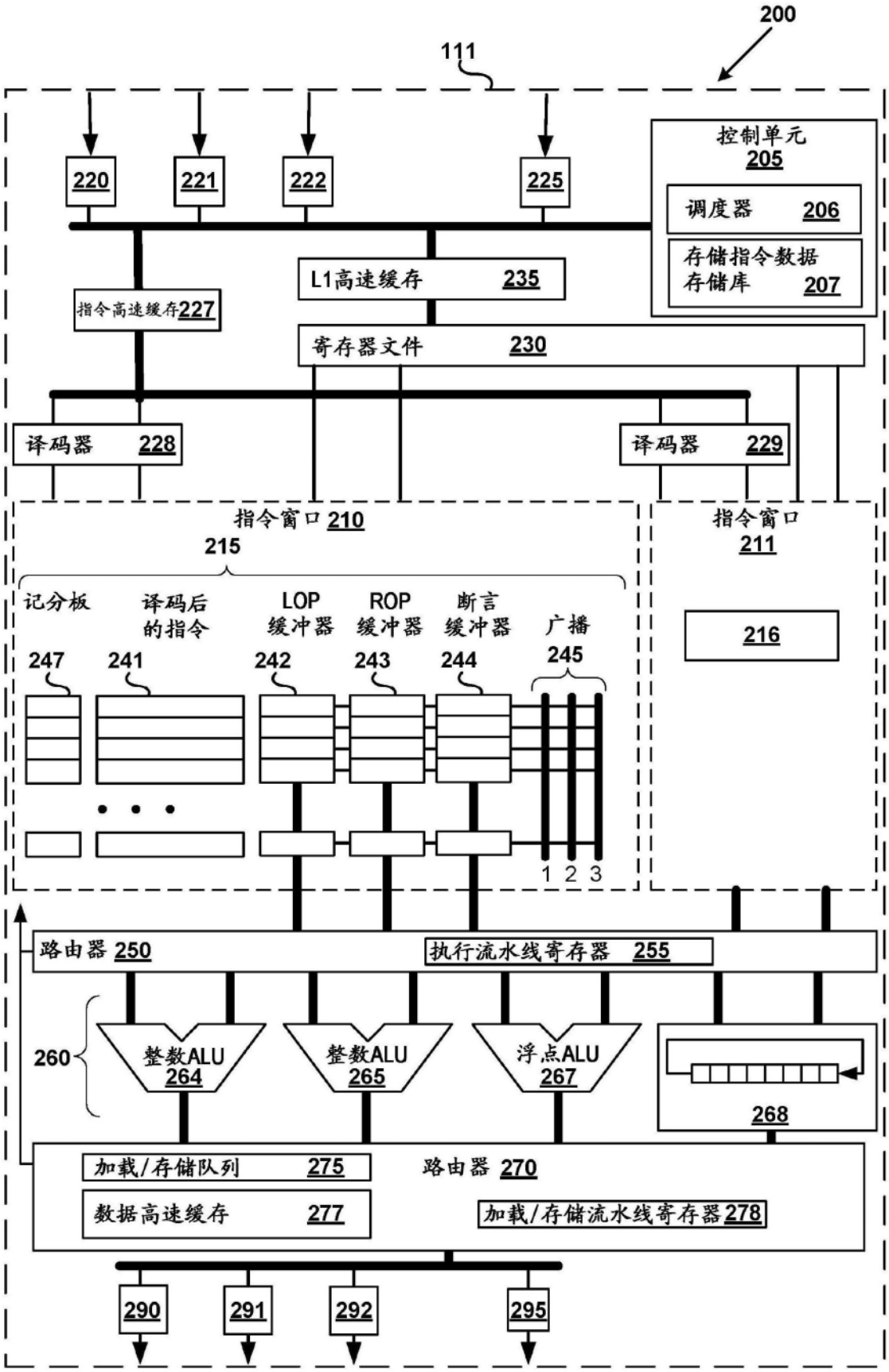

200
111
220
221
222
225
控制单元
205
调度器 206
存储指令数据
存储库 207
L1高速缓存 235
指令高速缓存227
寄存器文件 230
译码器 228
译码器 229
指令窗口 210
215
指令窗口
211
记分板
译码后
的指令
LOP
缓冲器
ROP
缓冲器
断言
缓冲器
广播
247
241
242
243
244
245
216
1 2 3
路由器 250
执行流水线寄存器 255
260
整数ALU
264
整数ALU
265
浮点ALU
267
268
加载/存储队列 275
路由器 270
数据高速缓存 277
加载/存储流水线寄存器278
290
291
292
295

图2

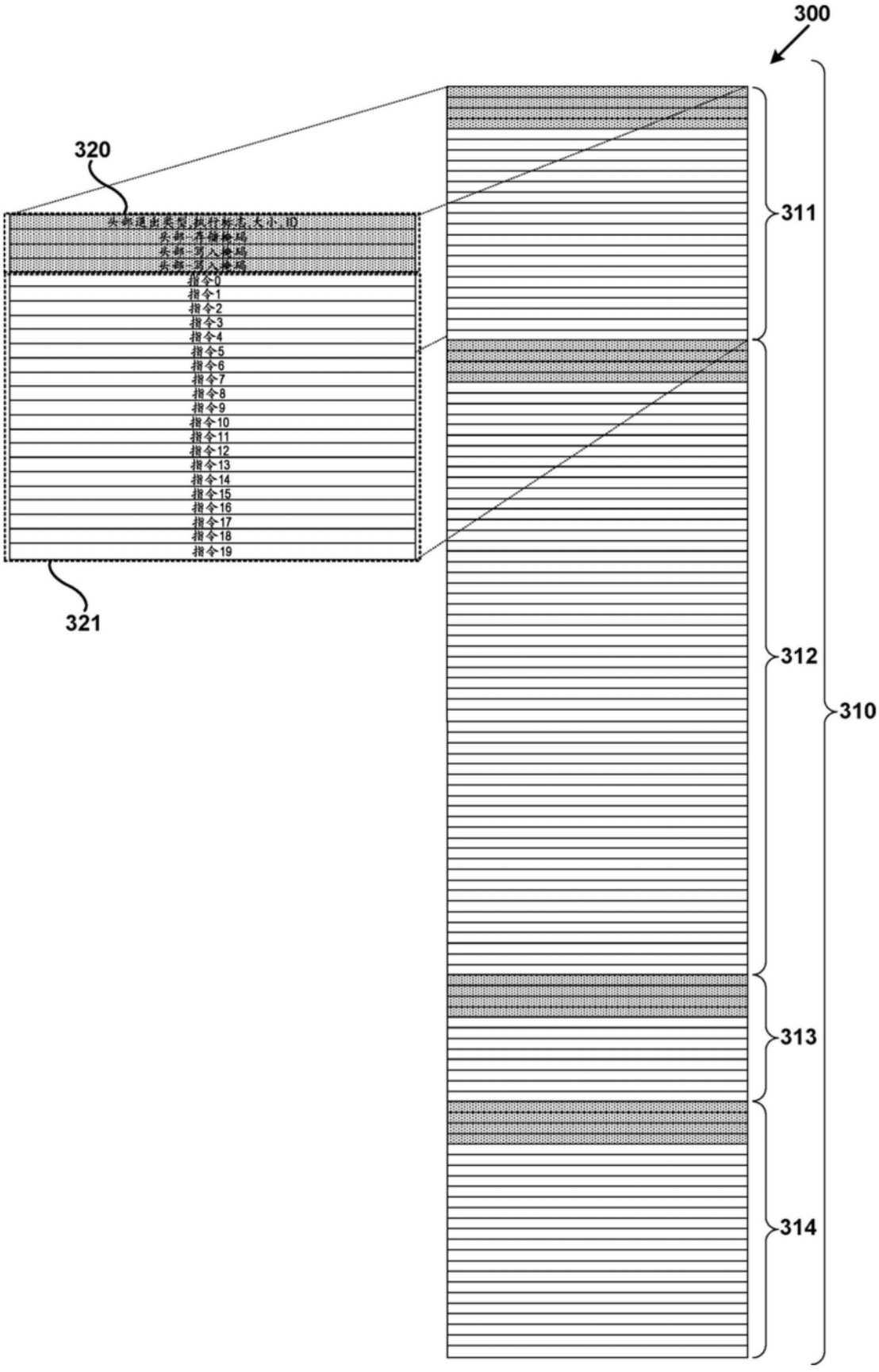

300
320
头部:通道类型,执行标志,大小,ID
头部-存储地址
头部-第1地址
头部-第2地址
指令0
指令1
指令2
指令3
指令4
指令5
指令6
指令7
指令8
指令9
指令10
指令11
指令12
指令13
指令14
指令15
指令16
指令17
指令18
指令19
321
311
312
313
314
310

图3

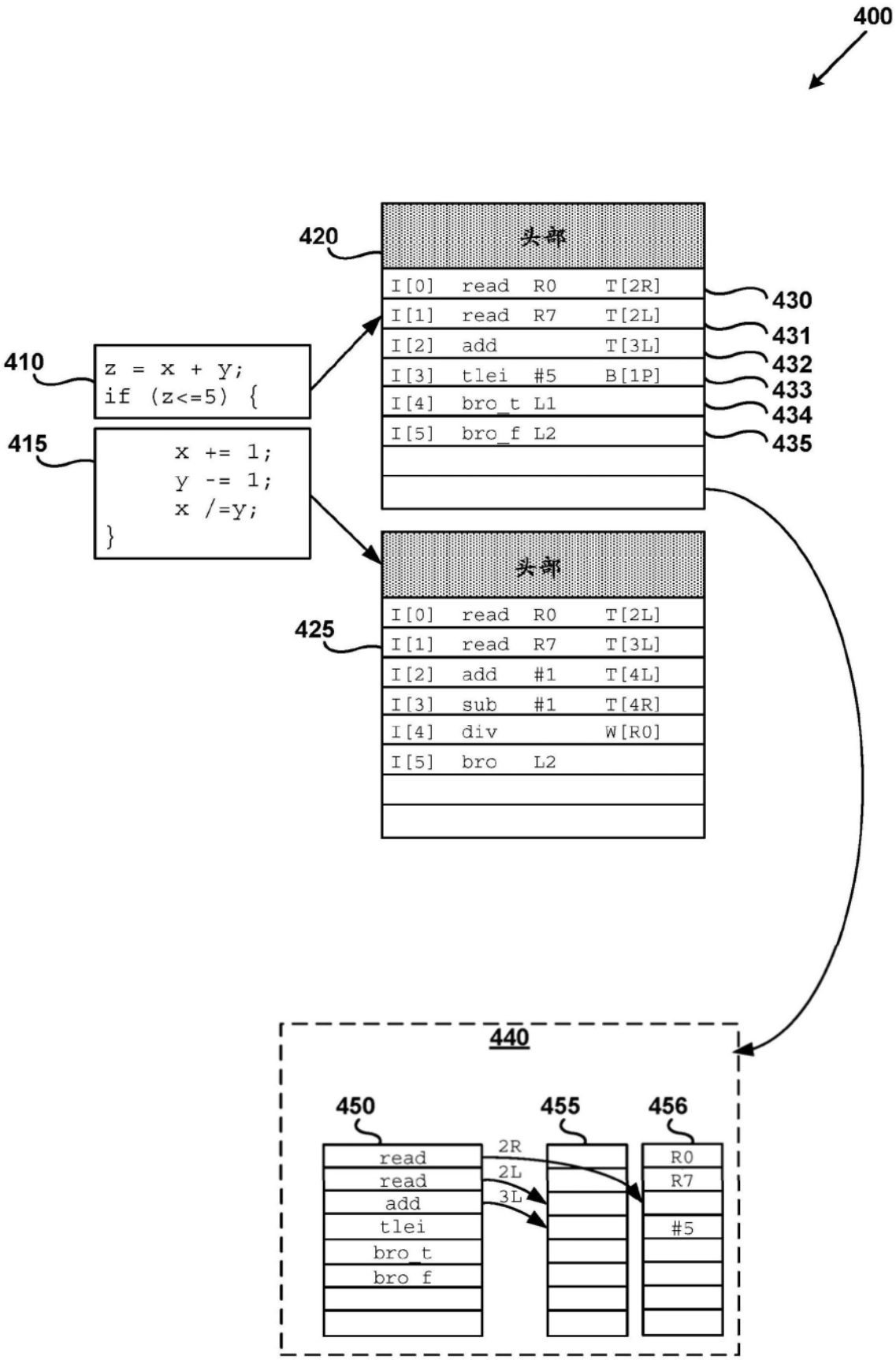

400
420
头部
I[0] read R0 T[2R]
I[1] read R7 T[2L]
I[2] add T[3L]
I[3] tlei #5 B[1P]
I[4] bro_t L1
I[5] bro_f L2
430
431
432
433
434
435
410
z = x + y;
if (z<=5) {
415
x += 1;
y -= 1;
x /=y;
}
头部
425
I[0] read R0 T[2L]
I[1] read R7 T[3L]
I[2] add #1 T[4L]
I[3] sub #1 T[4R]
I[4] div W[R0]
I[5] bro L2
440
450
455
456
read
read
add
tlei
bro_t
bro_f
2R
2L
3L
R0
R7
#5

图4

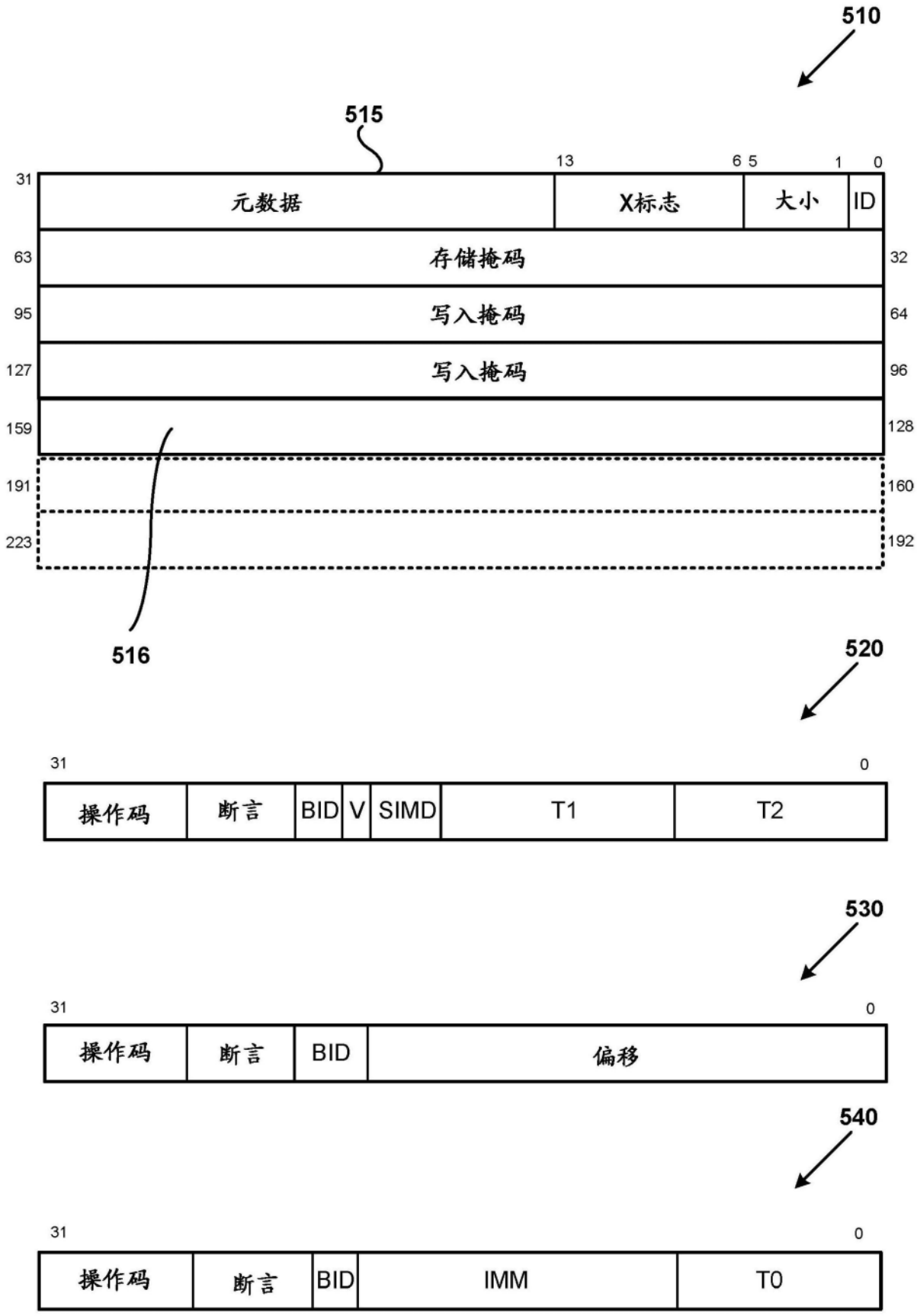

510
515
31
13
6 5
1
0
元数据
X标志
大小
ID
63
存储掩码
32
95
写入掩码
64
127
写入掩码
96
159
128
191
160
223
192
516
520
31
0
操作码
断言
BID
V
SIMD
T1
T2
530
31
0
操作码
断言
BID
偏移
540
31
0
操作码
断言
BID
IMM
T0

图5

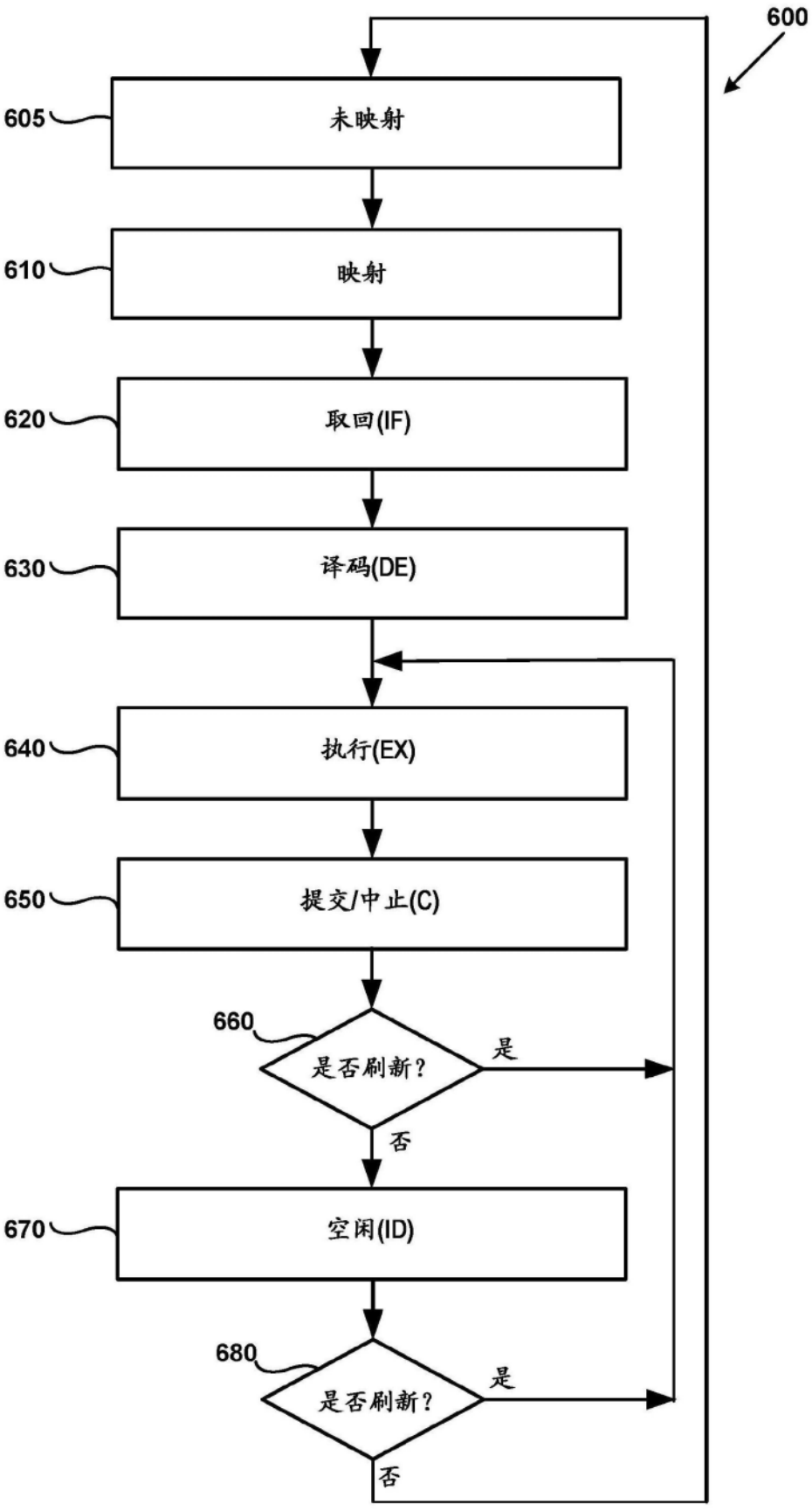

600
605
未映射
610
映射
620
取回(IF)
630
译码(DE)
640
执行(EX)
650
提交/中止(C)
660
是否刷新?
是
否
670
空闲(ID)
680
是否刷新?
是
否

图6

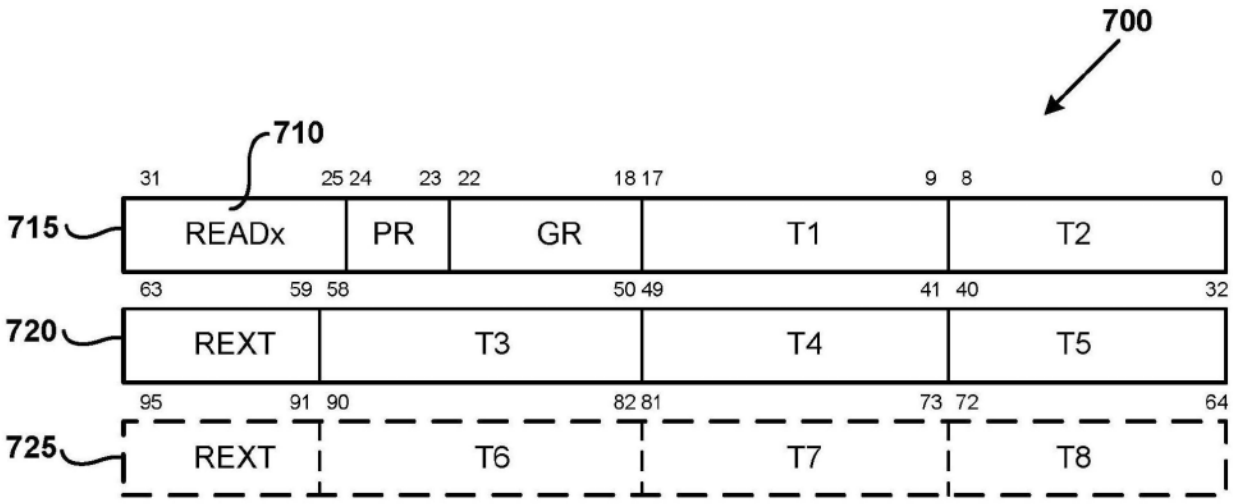

700
710
31
25 24
23 22
18 17
9 8
0
715
READx
PR
GR
T1
T2
63
59 58
50 49
41 40
32
720
REXT
T3
T4
T5
95
91 90
82 81
73 72
64
725
REXT
T6
T7
T8

图7

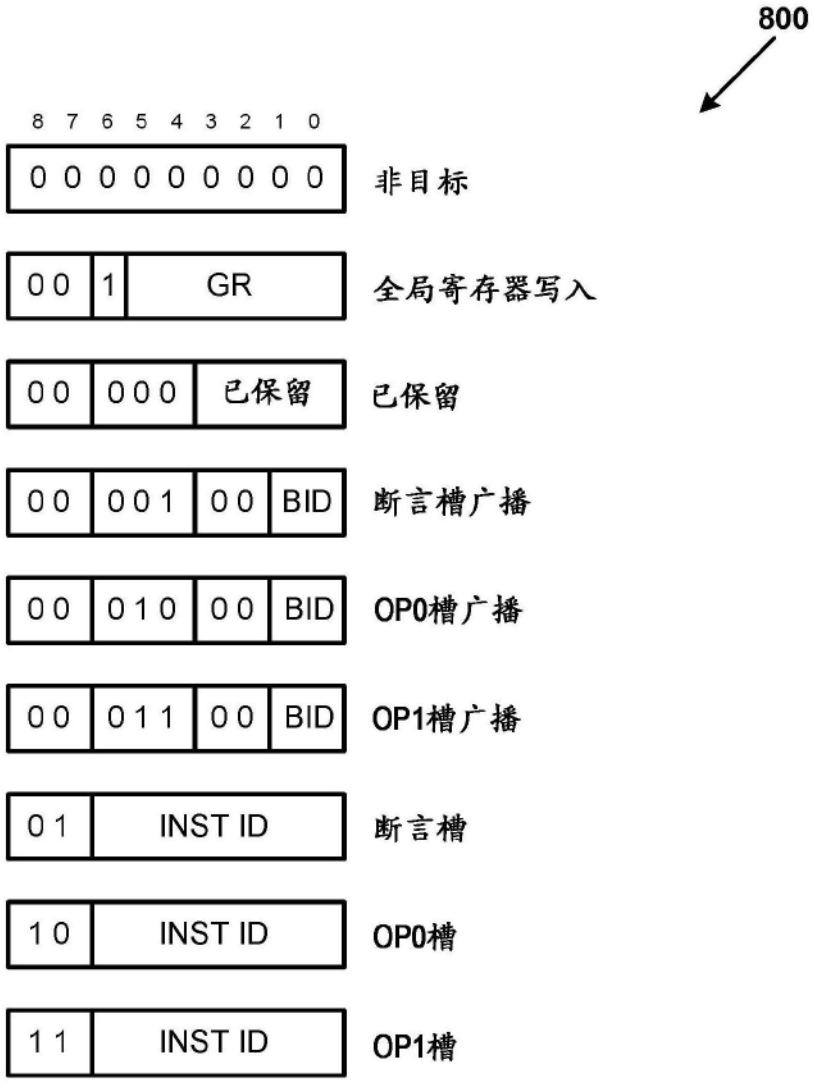
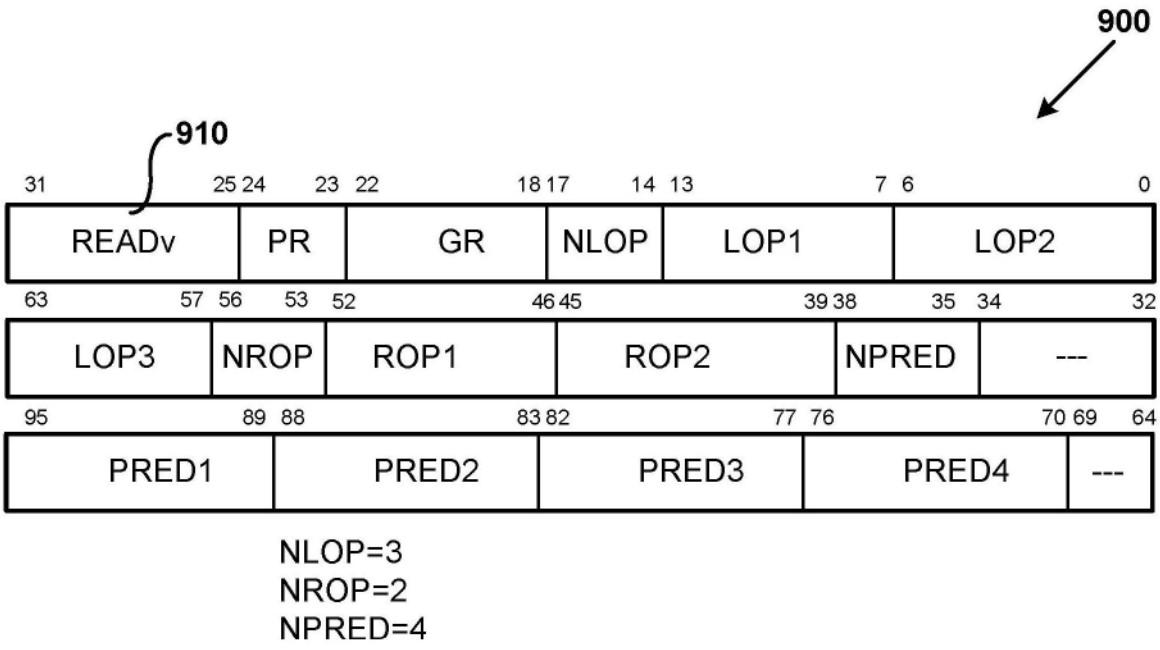

800
8 7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0 0
非目标
0 0
1
GR
全局寄存器写入
0 0
0 0 0
已保留
已保留
0 0
0 0 1
0 0
BID
断言槽广播
0 0
0 1 0
0 0
BID
OP0槽广播
0 0
0 1 1
0 0
BID
OP1槽广播
0 1
INST ID
断言槽
1 0
INST ID
OP0槽
1 1
INST ID
OP1槽

图8


900
910
31 25 24 23 22 18 17 14 13 7 6 0
READv PR GR NLOP LOP1 LOP2
63 57 56 53 52 46 45 39 38 35 34 32
LOP3 NROP ROP1 ROP2 NPRED ---
95 89 88 83 82 77 76 70 69 64
PRED1 PRED2 PRED3 PRED4 ---
NLOP=3
NROP=2
NPRED=4

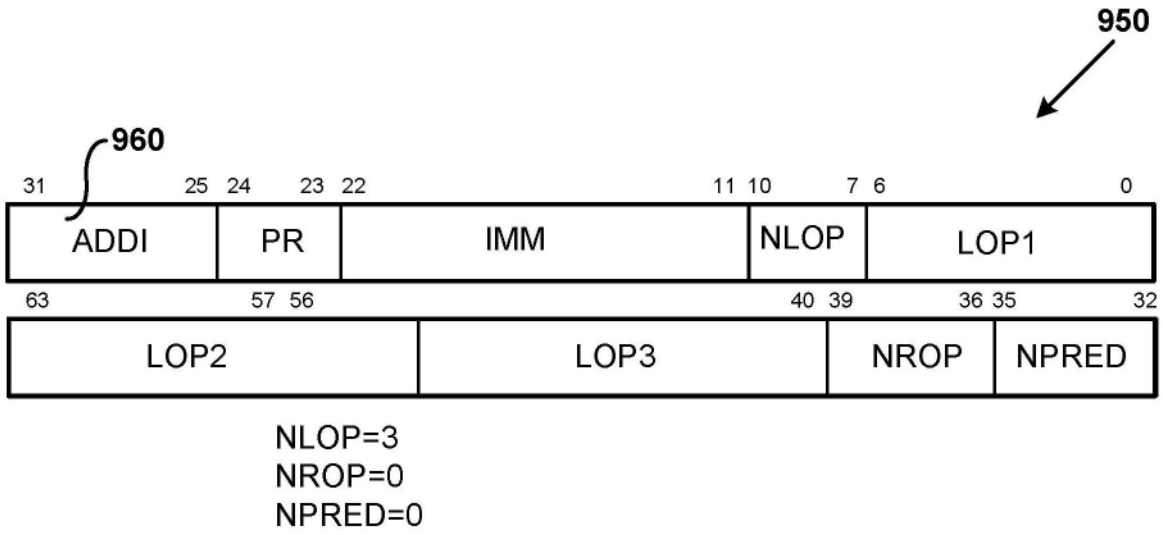

950
960
31 25 24 23 22 11 10 7 6 0
ADDI PR IMM NLOP LOP1
63 57 56 40 39 36 35 32
LOP2 LOP3 NROP NPRED
NLOP=3
NROP=0
NPRED=0

图9

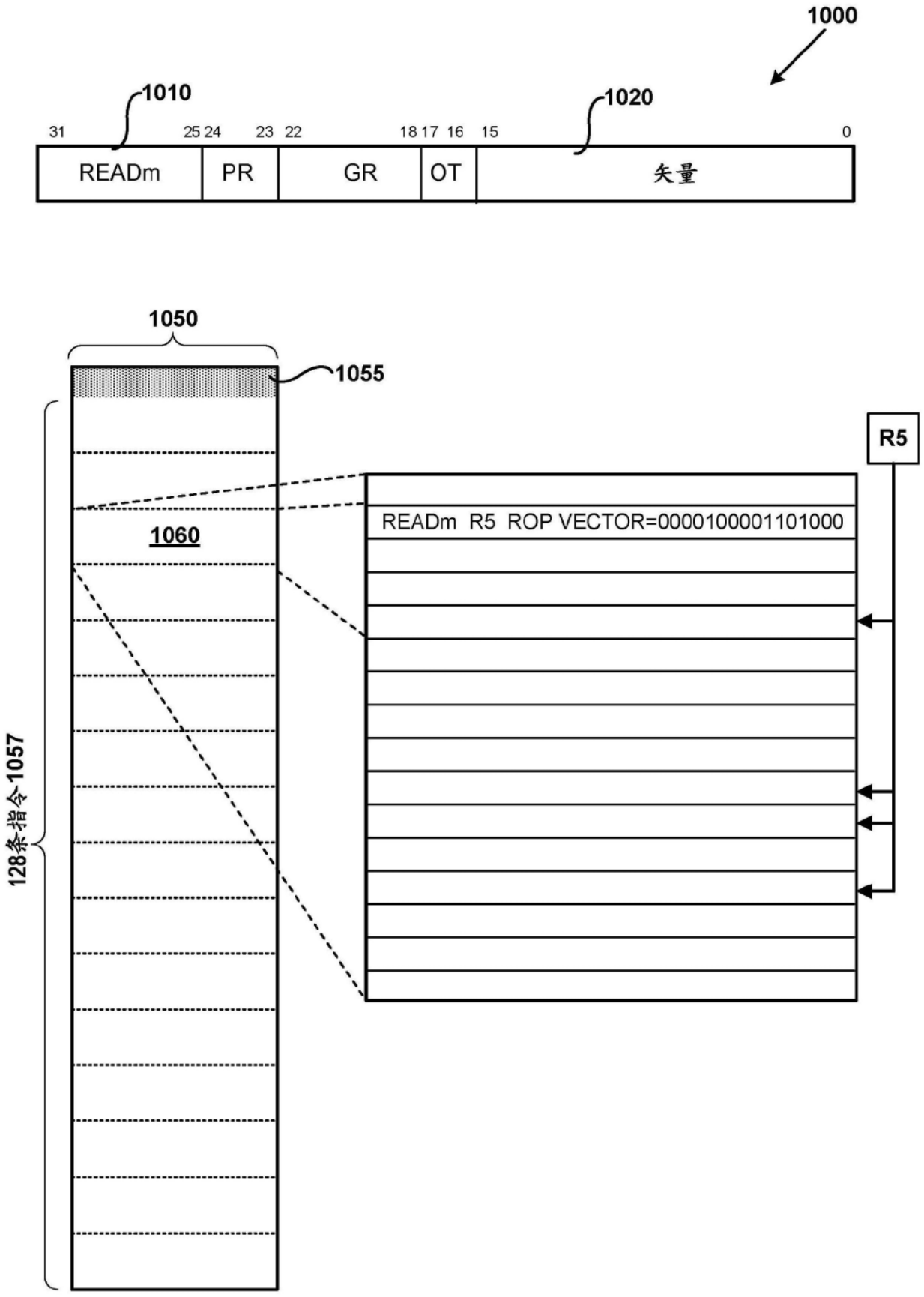

1000
1010
1020
31
25 24
23 22
18 17 16
15
0
READm
PR
GR
OT
矢量
1050
1055
R5
READm R5 ROP VECTOR=0000100001101000
1060
128条指令1057

图10

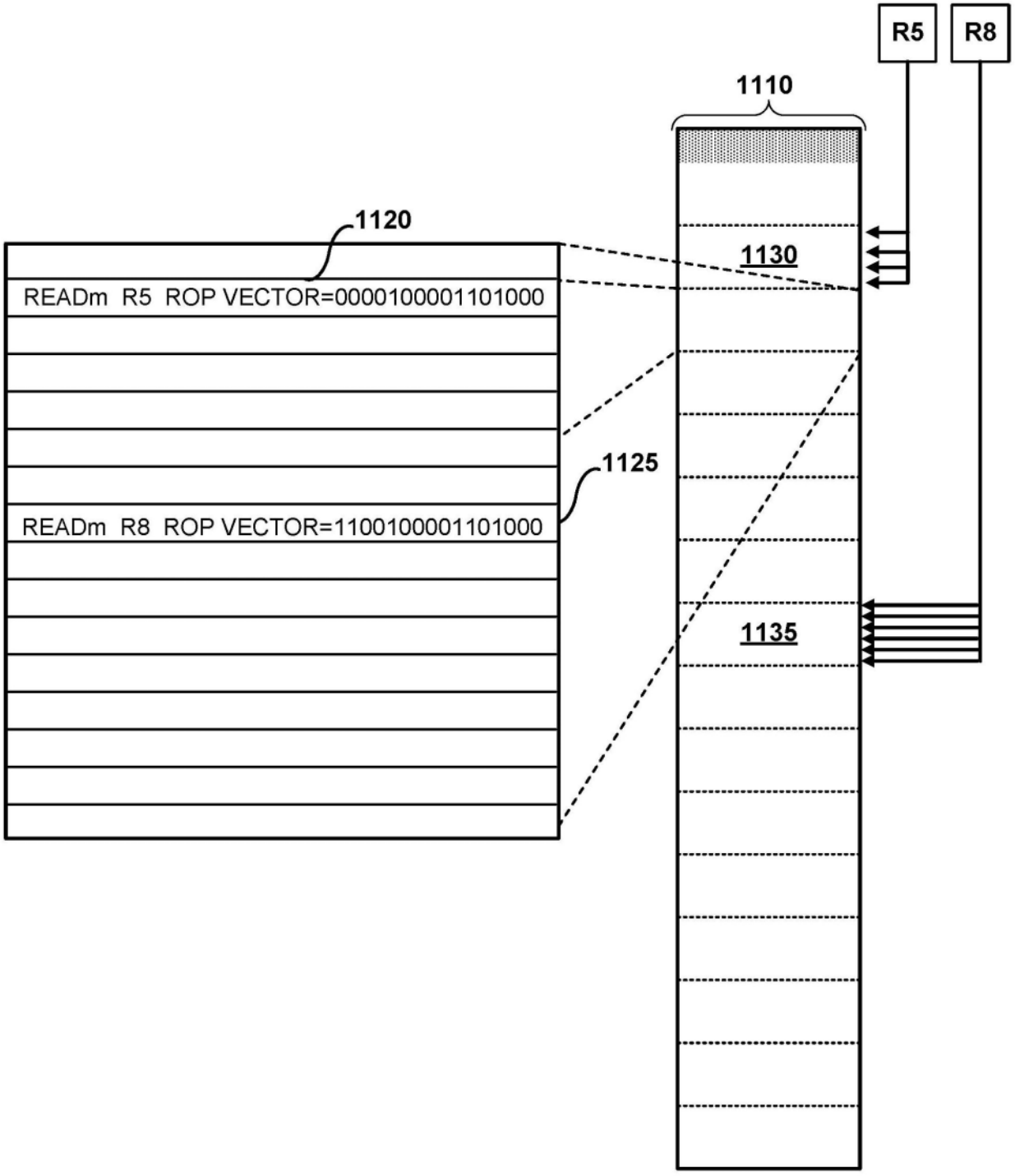

R5
R8
1110
1120
1130
READm R5 ROP VECTOR=0000100001101000
1125
READm R8 ROP VECTOR=1100100001101000
1135

图11

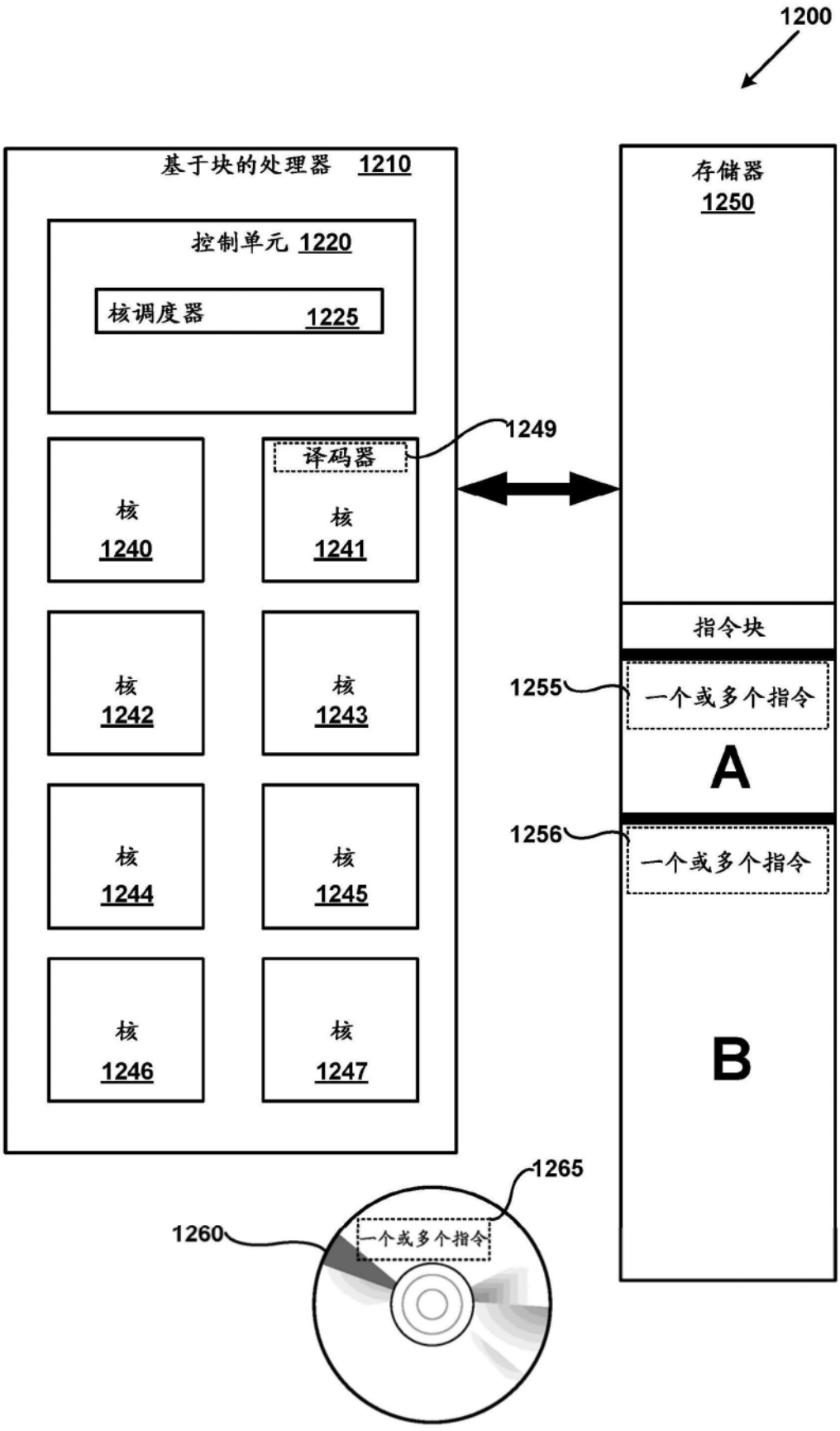

1200
基于块的处理器 1210
控制单元 1220
核调度器 1225
译码器
1249
核 1240
核 1241
核 1242
核 1243
核 1244
核 1245
核 1246
核 1247
存储器 1250
指令块
1255
一个或多个指令
A
1256
一个或多个指令
B
1265
1260
一个或多个指令

图12

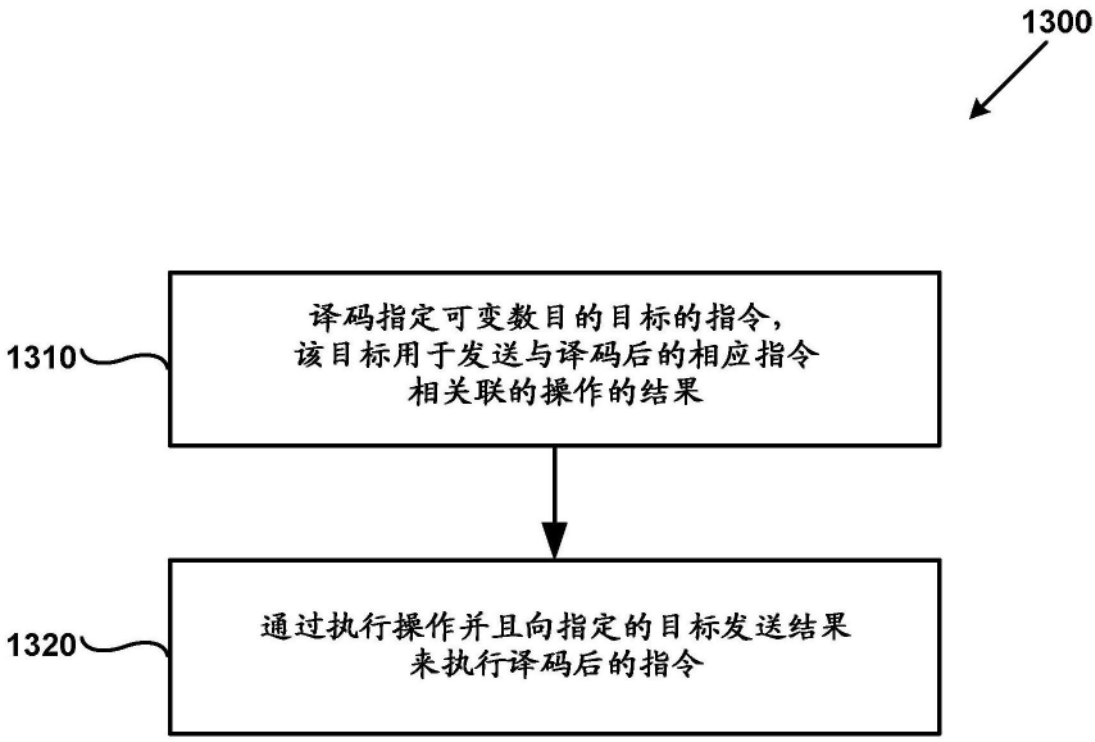

1300
译码指定可变数目的目标的指令，
该目标用于发送与译码后的相应指令
相关联的操作的结果
1310
通过执行操作并且向指定的目标发送结果
来执行译码后的指令
1320

图13

1410

```
++i;
if(a[i]>a[i+1] {
  t      = a[i];
  a[i]   = a[i+1];
  a[i+1] = t;
} else {
  b[i]=b[i]+1;
  if(i>100) {
    x=b[i];
  } else {
    b[5]=0xff;
  }
}
```

1420

```
//  opcd  src bid     target(s)
 0: readl  R1         T[3R]  T[19R]   // i
 1: readl  R2         T[3L]           // a
 2: readl  R3         T[14R] T[21R]   // b
 3: addi   #1         B1R             // ++i
 4: add       B1R     T[5R]  T[6R]    // &a[i] (i.e., a+i)
 5: mov               T[7R]  T[8R]
 6: mov               T[12R] T[13R}
 7: ld     #0         T[9R]           // a[i]
 8: ld     #1         T[10R]          // a[i+1]
 9: mov               T[13L] T[11L]   // a[i]
10: mov               T[12L] T[11R]   // a[i+1]
11: tgt               B2P    T[14P]   // a[i] > a[i+1] ?
12: sd_t     B2P                      // a[i]   = a[i+1]
13: sd_t     B2P                      // a[i+1] = t (i.e., a[i])
14: mov               T[15L] T[19R]   // b
15: add      B1R      T[16R] T[18R]   // &b[i]
16: ld_f   #0 B2P     T[17R]          // b[i]
17: addi_f #1 B2P     T[18L]          // b[i]+1
18: sd_f     B2P                      // b[i] = b[i]+1
19: addi   #1         T[20R] T[22R]   // ++i (duplicate)
20: tgt_f  #100 B2P   T[23P] T[26P]   // i > 100 (i<=100)
21: mov               T[22L] T[24R]   // b
22: add      B1R      T[23R] T[25R]   // &b[i]
23: ld_t   #0         R5              // x=b[i]
24: addi   #5           T[26R]           // &b[5]
25: movi   #ff        T[26L]          // 0xff
26: sd_f   #0                         // b[5]=0xff
27: bro    NEXTBLOCK                  // jump to next instruction block
```

图14A

1410

```
++i;
if(a[i]>a[i+1] {
  t      = a[i];
  a[i]   = a[i+1];
  a[i+1] = t;
} else {
  b[i]=b[i]+1;
  if(i>100) {
    x=b[i];
  } else {
    b[5]=0xff;
  }
}
```

1430

```
//  opcd  src    target(s)
 0: readl  R1     T[3R]  T[19R]                    // i
 1: readl  R2     T[4L]                            // a
 2: readv  R3     T[15L] T[19R] T[22L] T[24R]      // b
 3: addim  #1     T[4R]  T[15R] T[20R] T[22R]      // ++i    (was: B1R)
 4: addx          T[7R]  T[8R]  T[12R] T[13R]      // &a[i] (i.e., a+i)
 7: ld     #0     T[13L] T[11L]                    // a[i]
 8: ld     #1     T[12L] T[11R]                    // a[i+1]
11: tgtm          T[12P] T[13P] T[14P] T[16P] T[17P] T[18P] T[20P]
                                              // a[i] > a[i+1] ?   /(was: B2P)
12: sd_t                                         // a[i]   = a[i+1]
13: sd_t                                         // a[i+1] = t (i.e., a[i])
15: add           T[16R] T[18R]                  // &b[i]
16: ld_f  #0      T[17R]                         // b[i]
17: addi_f #1     T[18L]                         // b[i]+1
18: sd_f                                         // b[i] = b[i]+1
20: tgt_f #100    T[23P] T[26P]                  // i > 100 (i<=100)
22: add           T[23R] T[25R]                  // &b[i]
23: ld_t  #0      R5                             // x=b[i]
24: addi  #5      T[26R]                         // &b[5]
25: movi  #ff     T[26L]                         // 0xff
26: sd_f  #0                                     // b[5]=0xff
27: bro   NEXTBLOCK                          // jump to next instruction block
```

图14B

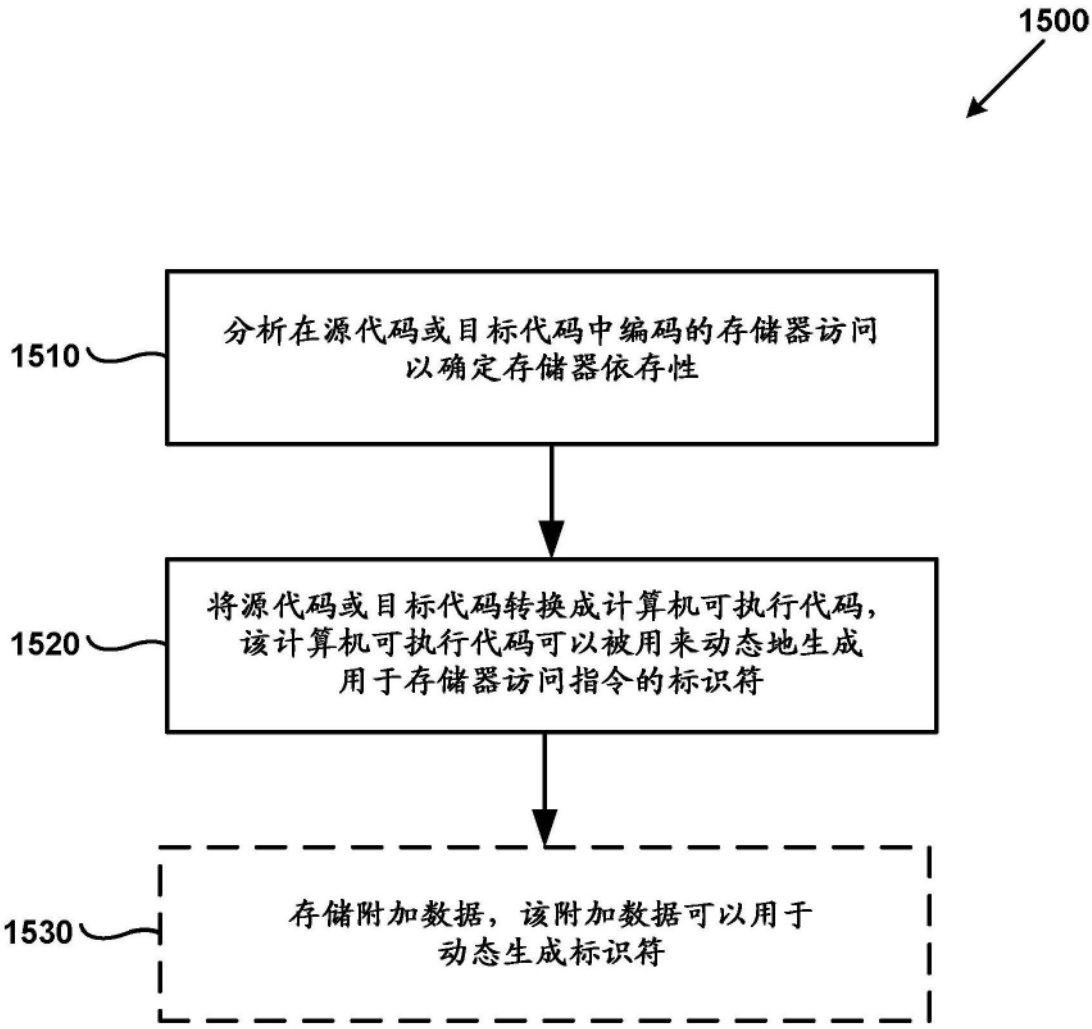

1500
1510
分析在源代码或目标代码中编码的存储器访问
以确定存储器依存性
1520
将源代码或目标代码转换成计算机可执行代码，
该计算机可执行代码可以被用来动态地生成
用于存储器访问指令的标识符
1530
存储附加数据，该附加数据可以用于
动态生成标识符

图15

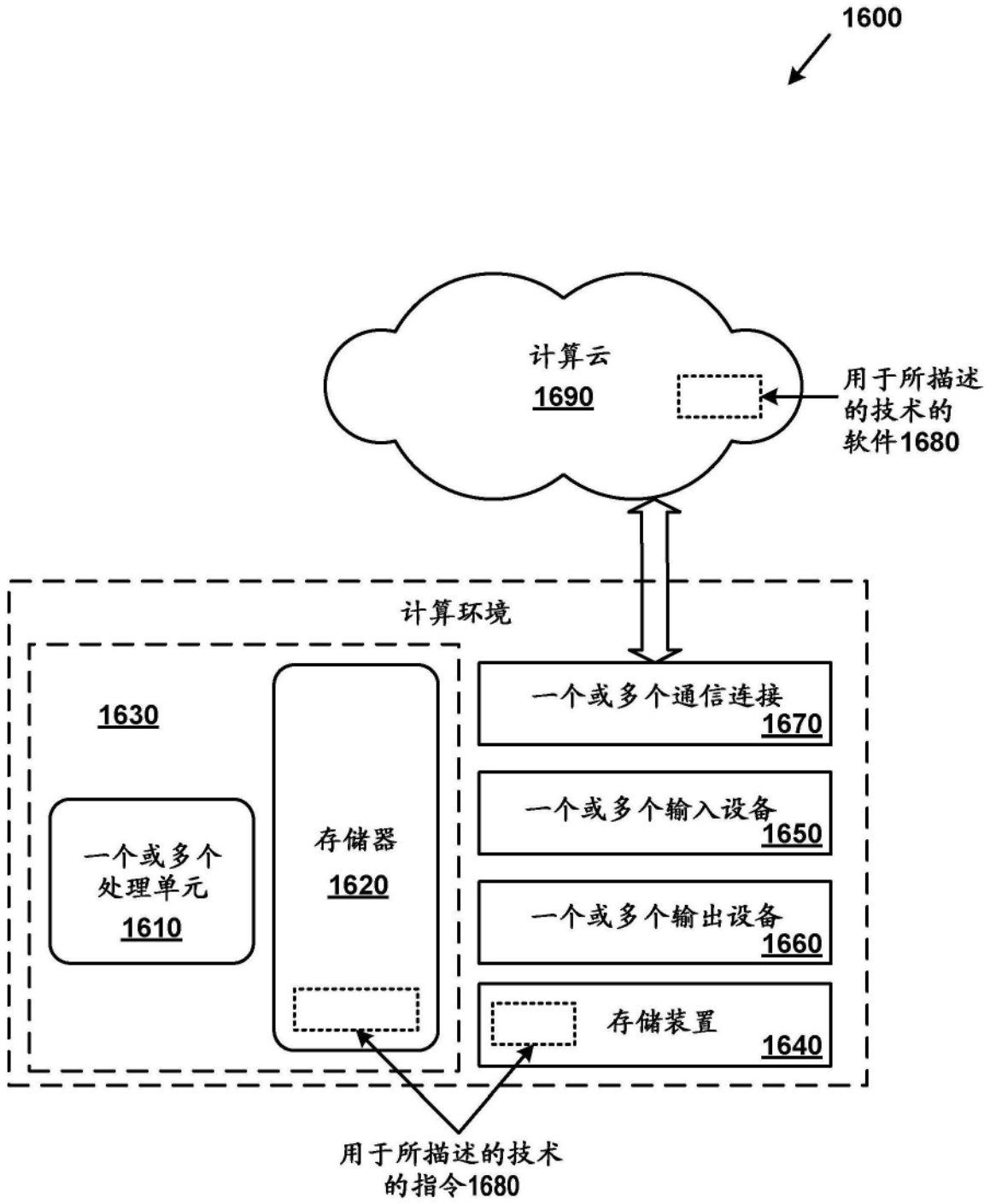

1600
计算云
1690
用于所描述的技术的软件1680
计算环境
1630
一个或多个处理单元
1610
存储器
1620
一个或多个通信连接
1670
一个或多个输入设备
1650
一个或多个输出设备
1660
存储装置
1640
用于所描述的技术的指令1680

图16