(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 106547518 B
(45)授权公告日 2019.02.01

(21)申请号 201611088838.6
(22)申请日 2013.06.20
(65)同一申请的已公布的文献号
申请公布号 CN 106547518 A
(43)申请公布日 2017.03.29
(30)优先权数据
13/729,915 2012.12.28 US
(62)分案原申请数据
201380059921.9 2013.06.20
(73)专利权人 英特尔公司
地址 美国加利福尼亚州
(72)发明人 O•本-琪琪 I•帕多 R•凡伦天 E•威斯曼 D•马可维奇 Y•优素福
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 黄嵩泉
(51)Int.Cl.
G06F 9/30(2006.01)
G06F 9/38(2006.01)
(56)对比文件
US 5430841 A,1995.07.04,
CN 102741826 A,2012.10.17,
CN 102314671 A,2012.01.11,
US 7598958 B1,2009.10.06,
US 7746350 B1,2010.06.29,
CN 102741806 A,2012.10.17,
审查员 刘畅

权利要求书5页 说明书27页 附图23页

(54)发明名称

用于加速器的低等待时间调用的装置和方法

(57)摘要

描述了用于提供加速器的低等待时间调用的装置和方法。例如，根据一个实施例的处理器包括：命令寄存器，用于存储标识将被执行的命令的命令数据；结果寄存器，用于存储命令的结果或指示该命令为何不能被执行的原因的数据；执行逻辑，用于执行多条指令，这些指令包括用于调用一个或多个加速器命令的加速器调用指令；以及一个或多个加速器，用于从命令寄存器中读取命令数据，并且响应性地尝试执行由命令数据标识的命令。

CN 106547518 B

1.一种处理器,包括:

多个同时多线程SMT核,所述SMT核中的每一个都用于执行对多个线程的乱序指令执行;

至少一个共享高速缓存电路,用于在所述SMT核中的两个或更多个之间被共享;

所述SMT核中的至少一个SMT核包括:

指令取出电路,用于取出所述线程中的一个或多个线程的指令;

指令解码电路,用于解码所述指令;

寄存器重命名电路,用于重命名寄存器组的寄存器;

指令高速缓存电路,用于存储待执行的指令;以及

数据高速缓存电路,用于存储数据;

至少一个第二级L2高速缓存电路,用于存储指令和数据两者且通信地耦合至所述指令高速缓存电路和所述数据高速缓存电路;

通信互连电路,用于将所述SMT核中的一个或多个通信地耦合至加速器设备;

所述通信互连电路用于提供对包括所述至少一个共享高速缓存电路的所述处理器的资源的加速器设备访问;以及

存储器访问电路,用于在存储来自应用的执行的数据的存储器空间中标识所述加速器设备的加速器上下文保存/恢复区域,所述应用在应用的执行期间调用所述加速器设备,所述加速器上下文保存/恢复区域由上下文保存/恢复指针来指示,且所述加速器上下文保存/恢复区域用于存储加速器上下文状态。

2.如权利要求1所述的处理器,其中,所述加速器设备用于从所述加速器上下文保存/恢复区域恢复所述加速器的上下文状态。

3.如权利要求1所述的处理器,其中,所述上下文保存/恢复指针标识存储器地址。

4.如权利要求1所述的处理器,其特征在于,进一步包括:

寄存器,用于存储所述上下文保存/恢复指针。

5.如权利要求1所述的处理器,其中所述通信互连电路包括外围组件互连快速PCIe电路。

6.一种方法,包括:

执行对多个同时多线程SMT核上的多个线程的乱序指令执行;

在所述SMT核中的两个或更多个之间共享至少一个共享高速缓存;

取出所述线程中的一个或多个线程的指令;

解码所述指令;

重命名寄存器组的寄存器;

将待执行的指令存储在指令高速缓存电路中;

将数据存储在数据高速缓存电路中;

将指令和数据两者存储在至少一个第二级高速缓存电路中,所述至少一个第二级高速缓存电路通信地耦合至所述指令高速缓存电路和所述数据高速缓存电路;

将所述SMT核中的一个或多个通信地耦合至加速器设备;

提供对包括所述至少一个共享高速缓存电路的处理器的资源的加速器设备访问;以及

在存储来自应用的执行的数据的存储器空间中标识所述加速器设备的加速器上下文

保存/恢复区域，所述应用在应用的执行期间调用所述加速器设备，所述加速器上下文保存/恢复区域由上下文保存/恢复指针来指示，且所述加速器上下文保存/恢复区域用于存储加速器上下文状态。

7.如权利要求6所述的方法，其中，所述加速器设备用于从所述加速器上下文保存/恢复区域恢复所述加速器的上下文状态。

8.如权利要求6所述的方法，其中，所述上下文保存/恢复指针标识存储器地址。

9.如权利要求6所述的方法，进一步包括：

将所述上下文保存/恢复指针存储在寄存器中。

10.如权利要求6所述的方法，其中，所述一个或多个SMT核通过外围组件互连快速PCIe电路而通信地耦合至所述加速器设备。

11.一种设备，包括：

用于执行对多个同时多线程SMT核上的多个线程的乱序指令执行的装置；

用于在所述SMT核中的两个或更多个之间共享至少一个共享高速缓存的装置；

用于取出所述线程中的一个或多个线程的指令的装置；

用于解码所述指令的装置；

用于重命名寄存器组的寄存器的装置；

用于将待执行的指令存储在指令高速缓存电路中的装置；

用于将数据存储在数据高速缓存电路中的装置；

用于将指令和数据两者存储在至少一个第二级高速缓存电路中的装置，所述至少一个第二级高速缓存电路通信地耦合至所述指令高速缓存电路和所述数据高速缓存电路；

用于将所述SMT核中的一个或多个通信地耦合至加速器设备的装置；

用于提供对包括所述至少一个共享高速缓存电路的所述设备的资源的加速器设备访问的装置；以及

用于在存储来自应用的执行的数据的存储器空间中标识所述加速器设备的加速器上下文保存/恢复区域的装置，所述应用在应用的执行期间调用所述加速器设备，所述加速器上下文保存/恢复区域由上下文保存/恢复指针来指示，且所述加速器上下文保存/恢复区域用于存储加速器上下文状态。

12.如权利要求11所述的设备，其中，所述加速器设备用于从所述加速器上下文保存/恢复区域恢复所述加速器的上下文状态。

13.如权利要求11所述的设备，其中，所述上下文保存/恢复指针标识存储器地址。

14.如权利要求11所述的设备，进一步包括：

寄存器，用于存储所述上下文保存/恢复指针。

15.如权利要求11所述的设备，其中所述用于将所述SMT核中的一个或多个通信地耦合至加速器设备的装置包括外围组件互连快速PCIe电路。

16.一种系统，包括：

多个处理器；

第一互连，用于通信地耦合所述多个处理器中的两个或更多个；

第二互连，用于将所述多个处理器中的一个或多个通信地耦合至一个或多个其他系统组件；以及

系统存储器,通信地耦合至所述处理器中的一个或多个;

至少一个处理器,包括:

多个同时多线程SMT核,所述SMT核中的每一个都用于执行对多个线程的乱序指令执行;

至少一个共享高速缓存电路,用于在所述SMT核中的两个或更多个之间被共享;

所述SMT核中的至少一个SMT核包括:

指令取出电路,用于取出所述线程中的一个或多个线程的指令;

指令解码电路,用于解码所述指令;

寄存器重命名电路,用于重命名寄存器组的寄存器;

指令高速缓存电路,用于存储待执行的指令

数据高速缓存电路,用于存储数据;

至少一个第二级L2高速缓存电路,用于存储指令和数据两者且通信地耦合至所述指令高速缓存电路和所述数据高速缓存电路;

通信互连电路,用于将所述SMT核中的一个或多个通信地耦合至加速器设备;

所述通信互连电路用于提供对包括所述至少一个共享高速缓存电路的所述处理器的资源的加速器设备访问;以及

存储器访问电路,用于在存储来自应用的执行的数据的存储器空间中标识所述加速器设备的加速器上下文保存/恢复区域,所述应用在应用的执行期间调用所述加速器设备,所述加速器上下文保存/恢复区域由上下文保存/恢复指针来指示,且所述加速器上下文保存/恢复区域用于存储加速器上下文状态。

17.如权利要求16所述的系统,其中,所述加速器设备用于从所述加速器上下文保存/恢复区域恢复所述加速器的上下文状态。

18.如权利要求16所述的系统,其中,所述上下文保存/恢复指针标识存储器地址。

19.如权利要求16所述的系统,进一步包括:

寄存器,用于存储所述上下文保存/恢复指针。

20.如权利要求16所述的系统,其中所述通信互连电路包括外围组件互连快速PCIe电路。

21.如权利要求16所述的系统,进一步包括:

至少一个存储设备,通信地耦合至所述处理器中的一个或多个。

22.如权利要求16所述的系统,进一步包括:

至少一个通信设备,通信地耦合至所述处理器中的一个或多个。

23.如权利要求16所述的系统,其中所述系统存储器包括动态随机存取存储器。

24.一种方法,包括:

通信地耦合多个处理器中的两个或更多个;

将所述多个处理器中的一个或多个通信地耦合至一个或多个其他系统组件;以及

将系统存储器通信地耦合至所述处理器中的一个或多个;

执行对多个同时多线程SMT核上的多个线程的乱序指令执行;

在所述SMT核中的两个或更多个之间共享至少一个共享高速缓存;

取出所述线程中的一个或多个线程的指令;

解码所述指令；

重命名寄存器组的寄存器；

将待执行的指令存储在指令高速缓存电路中；

将数据存储在数据高速缓存电路中；

将指令和数据两者存储在至少一个第二级高速缓存电路中，所述至少一个第二级高速缓存电路通信地耦合至所述指令高速缓存电路和所述数据高速缓存电路；

将所述SMT核中的一个或多个通信地耦合至加速器设备；

提供对包括所述至少一个共享高速缓存电路的所述多个处理器中的一个或多个的资源的加速器设备访问；以及

在存储来自应用的执行的数据的存储器空间中标识所述加速器设备的加速器上下文保存/恢复区域，所述应用在应用的执行期间调用所述加速器设备，所述加速器上下文保存/恢复区域由上下文保存/恢复指针来指示，且所述加速器上下文保存/恢复区域用于存储加速器上下文状态。

25.如权利要求24所述的方法，其中，所述加速器设备用于从所述加速器上下文保存/恢复区域恢复所述加速器的上下文状态。

26.如权利要求24所述的方法，其中，所述上下文保存/恢复指针标识存储器地址。

27.如权利要求24所述的方法，进一步包括：

将所述上下文保存/恢复指针存储在寄存器中。

28.如权利要求24所述的方法，其中，所述一个或多个SMT核通过外围组件互连快速PCIe电路而通信地耦合至所述加速器设备。

29.如权利要求24所述的方法，进一步包括：

将至少一个存储设备通信地耦合至所述处理器中的一个或多个。

30.如权利要求24所述的方法，进一步包括：

将至少一个通信设备通信地耦合至所述处理器中的一个或多个。

31.如权利要求24所述的方法，其中所述系统存储器包括动态随机存取存储器。

32.一种设备，包括：

用于通信地耦合多个处理器中的两个或更多个的装置；

用于将所述多个处理器中的一个或多个通信地耦合至一个或多个其他系统组件的装置；以及

用于将系统存储器通信地耦合至所述处理器中的一个或多个的装置；

用于执行对多个同时多线程SMT核上的多个线程的乱序指令执行的装置；

用于在所述SMT核中的两个或更多个之间共享至少一个共享高速缓存的装置；

用于取出所述线程中的一个或多个线程的指令的装置；

用于解码所述指令的装置；

用于重命名寄存器组的寄存器的装置；

用于将待执行的指令存储在指令高速缓存电路中的装置；

用于将数据存储在数据高速缓存电路中的装置；

用于将指令和数据两者存储在至少一个第二级高速缓存电路中的装置，所述至少一个第二级高速缓存电路通信地耦合至所述指令高速缓存电路和所述数据高速缓存电路；

用于将所述SMT核中的一个或多个通信地耦合至加速器设备的装置；

用于提供对包括所述至少一个共享高速缓存电路的设备的资源的加速器设备访问的装置;以及

用于在存储来自应用的执行的数据的存储器空间中标识所述加速器设备的加速器上下文保存/恢复区域的装置,所述应用在应用的执行期间调用所述加速器设备,所述加速器上下文保存/恢复区域由上下文保存/恢复指针来指示,且所述加速器上下文保存/恢复区域用于存储加速器上下文状态。

33.如权利要求32所述的设备,其中,所述加速器设备用于从所述加速器上下文保存/恢复区域恢复所述加速器的上下文状态。

34.如权利要求32所述的设备,其中,所述上下文保存/恢复指针标识存储器地址。

35.如权利要求32所述的设备,进一步包括：

用于将所述上下文保存/恢复指针存储在寄存器中的装置。

36.如权利要求32所述的设备,其中,所述一个或多个SMT核通过外围组件互连快速PCIe电路而通信地耦合至所述加速器设备。

37.如权利要求32所述的设备,进一步包括：

用于将至少一个存储设备通信地耦合至所述处理器中的一个或多个的装置。

38.如权利要求32所述的设备,进一步包括：

用于将至少一个通信设备通信地耦合至所述处理器中的一个或多个的装置。

39.如权利要求32所述的设备,其中所述系统存储器包括动态随机存取存储器。

40.一种具有指令的计算机可读存储介质,所述指令在被执行时使机器执行如权利要求6－10以及24－31的任一项所述的方法。

用于加速器的低等待时间调用的装置和方法

[0001] 本申请是PCT国际申请号为PCT/US2013/046863、国际申请日为2013年 6月20日、进入中国国家阶段的申请号为201380059921.9,题为“用于加速器的低等待时间调用的装置和方法”的申请的分案申请。

背景技术

技术领域

[0002] 本发明总体涉及计算机处理器领域。更具体地说,本发明涉及用于加速器的低等待时间调用的通用的可扩展指令。

背景技术

[0003] 如今,调用加速器需要通过驱动器接口。在其中使用层次结构保护域的系统中,这意味着切换到环0,并且将数据复制到不同的地址空间,从而消耗显著的时间和处理资源。由于高等待时间,此类加速器接口固有地也是异步的。可编程加速器要求被加速的代码以其自身的指令集架构(ISA)被实现。

[0004] 一些当前的处理器架构尝试解决这些顾虑中的一些,但是仅提供具有在被加速的任务请求及其执行之间的高等待时间的、粗粒度的异步机制。此外,当前的架构使用非X86ISA,这需要单独的工具链来生成被加速的任务,并将该被加速任务与主x86程序集成。

[0005] 此外,当前的异步硬件加速器(例如,GPU)允许被加速的任务执行与触发该被加速的任务的应用线程不相关的任务。这允许该应用线程处理异常和/ 或中断而不影响被加速的任务,并且甚至允许该应用线程在多个核之间迁移而不影响系统上被加速的任务的位置。

[0006] 当前的同步硬件加速器需要确保中断、异常、上下文切换和核迁移仍然是功能正确的,并且确保向前进展。这是通过下述之一完成的:(1)确保加速器足够短并且不导致任何异常,使得任何中断被推迟到完成该加速器为止;(2) 在现有的架构寄存器(例如,REPMOV)中保持加速器的向前进展;或(3) 定义保存加速器状态的新架构寄存器,并且将它们添加到XSAVE/XRESTORE。

附图说明

[0007] 结合以下附图,从以下具体实施方式中可获得对本发明更好的理解,其中:

[0008] 图1A是示出根据本发明的多个实施例的示例性有序流水线和示例性的寄存器重命名的乱序发布/执行流水线的框图;

[0009] 图1B是示出根据本发明的各实施例的要包括在处理器中的有序架构核的示例性实施例和示例性的寄存器重命名的乱序发布/执行架构核的框图;

[0010] 图2是根据本发明的多个实施例的具有集成的存储器控制器和图形器件的单核处理器和多核处理器的框图。

[0011] 图3示出根据本发明的一个实施例的系统的框图；

[0012] 图4示出根据本发明的实施例的第二系统的框图；

[0013] 图5示出根据本发明的实施例的第三系统的框图；

[0014] 图6示出根据本发明的实施例的芯片上系统(SoC)的框图；

[0015] 图7示出根据本发明的多个实施例的、对照使用软件指令转换器将源指令集中的二进制指令转换成目标指令集中的二进制指令的框图；

[0016] 图8A示出可在其中实现本发明的多个实施例的处理器架构；

[0017] 图8B-C示出存储用于调用加速器并回顾结果的数据的寄存器；

[0018] 图9A-C示出根据本发明的一个实施例的用于调用加速器的方法；

[0019] 图10示出用于处理经常失败的复杂指令的方法；

[0020] 图11示出使用用于存储加速器状态信息的栈的本发明的一个实施例。

[0021] 图12A和12B是示出根据本发明的多个实施例的通用向量友好指令格式及其指令模板的框图；

[0022] 图13A-D是示出根据本发明的多个实施例的示例性专用向量友好指令格式的框图；以及

[0023] 图14是根据本发明的一个实施例的寄存器架构的框图。

[0024] 图15示出根据本发明的某些实施例的计算机系统。

具体实施方式

[0025] 在下面的描述中，为了进行解释，阐述了众多具体细节以便提供对下述本发明的多个实施例的透彻理解。然而，对本领域的技术人员显而易见的是，可以在没有这些具体细节中的一些细节的情况下实施本发明的多个实施例。在其他实例中，公知的结构和设备以框图形式示出，以避免使本发明的多个实施例的基本原理模糊。

[0026] 示例性处理器架构和数据类型

[0027] 图1A是示出根据本发明的多个实施例的示例性有序流水线和示例性的寄存器重命名的乱序发布/执行流水线的框图。图1B是示出根据本发明的多个实施例的要包括在处理器中的有序架构核的示例性实施例和示例性的寄存器重命名的乱序发布/执行架构核的框图。图1A-B中的实线框示出了有序流水线和有序核，而可选增加的虚线框示出了寄存器重命名的、乱序发布/执行流水线和核。考虑到有序方面是乱序方面的子集，将描述乱序方面。

[0028] 在图1A中，处理器流水线100包括取出级102、长度解码级104、解码级106、分配级108、重命名级110、调度(也被称为分派或发布)级112、寄存器读取/存储器读取级114、执行级116、写回/存储器写入级118、异常处理级122和提交级124。

[0029] 图1B示出处理器核190，其包括耦合到执行引擎单元150的前端单元130，且执行引擎单元和前端单元两者都耦合到存储器单元170。核190可以是精简指令集计算(RISC)核、复杂指令集计算(CISC)核、超长指令字(VLIW) 核或混合或替代核类型。作为又一选项，核190可以是专用核，诸如例如，网络或通信核、压缩引擎、协处理器核、通用计算图形处理单元(GPGPU)核、图形核等。

[0030] 前端单元130包括耦合到指令高速缓存单元134的分支预测单元132，该指令高速

缓存单元耦合到指令转换后备缓冲器(TLB)136,该指令转换后备缓冲器耦合到指令取出单元138,指令取出单元耦合到解码单元140。解码单元140(或解码器)可解码指令,并生成从原始指令解码出的、或以其他方式反映原始指令的、或从原始指令导出的一个或多个微操作、微代码进入点、微指令、其他指令或其他控制信号作为输出。解码单元140可使用各种不同的机制来实现。合适机制的示例包括但不仅限于,查找表、硬件实现、可编程逻辑阵列(PLA)、微代码只读存储器(ROM)等。在一个实施例中,核190包括微代码ROM或存储用于某些宏指令的微代码的其他介质(例如,在解码单元 140中或以其他方式在前端单元130内)。解码单元140耦合至执行引擎单元 150中的重命名/分配器单元152。

[0031] 执行引擎单元150包括耦合到引退单元154和一个或多个调度器单元的集合156的重命名/分配器单元152。调度器单元156表示任意数量的不同调度器,包括预留站、中央指令窗等。调度器单元156耦合到物理寄存器组单元158。物理寄存器组单元158中的每一个表示一个或多个物理寄存器组,其中不同的物理寄存器组存储一个或多个不同的数据类型,例如,标量整数、标量浮点、紧缩整数、紧缩浮点、向量整数、向量浮点,状态(例如,作为要被执行的下一条指令的地址的指令指针)等。在一个实施例中,物理寄存器组单元158包括向量寄存器单元、写掩码寄存器单元和标量寄存器单元。这些寄存器单元可以提供架构向量寄存器、向量掩码寄存器、和通用寄存器。物理寄存器组单元 158被引退单元154覆盖,以示出可实现寄存器重命名和乱序执行的各种方式 (例如,使用重排序缓冲器和引退寄存器组;使用未来文件(future file)、历史缓冲器、引退寄存器组;使用寄存器映射和寄存器池等)。引退单元154和物理寄存器组单元158耦合至执行群集160。执行群集160包括一个或多个执行单元的集合162以及一个或多个存储器访问单元的集合164。执行单元162 可执行多种操作(例如,移位、加法、减法、乘法),并且可对多种数据类型 (例如,标量浮点、紧缩整数、紧缩浮点、向量整数、向量浮点)执行操作。尽管一些实施例可以包括专用于特定功能或功能集的多个执行单元,但其他实施例可包括全部执行所有功能的仅一个执行单元或多个执行单元。调度器单元 156、物理寄存器组单元158、执行群集160被示出为可能是复数个,因为某些实施例为某些数据/操作类型创建了诸个单独流水线(例如,均具有各自调度器单元、物理寄存器组单元和/或执行群集的标量整数流水线、标量浮点/紧缩整数/紧缩浮点/向量整数/向量浮点流水线、和/或存储器访问流水线,以及在单独的存储器访问流水线的情况下特定实施例被实现为仅仅该流水线的执行群集具有存储器访问单元164)。还应当理解,在使用分开的流水线的情况下,这些流水线中的一个或多个可以是乱序发布/执行的,并且其余流水线可以是有序发布/执行的。

[0032] 存储器访问单元的集合164耦合到存储器单元170,该存储器单元包括耦合到数据高速缓存单元174的数据TLB单元172,其中,数据高速缓存单元耦合到第二级(L2)高速缓存单元176。在一个示例性实施例中,存储器访问单元164可包括加载单元、存储地址单元和存储数据单元,其中的每一个均耦合至存储器单元170中的数据TLB单元172。指令高速缓存单元134还耦合到存储器单元170中的第二级(L2)高速缓存单元176。L2高速缓存单元176耦合到一个或多个其他级的高速缓存,并最终耦合到主存储器。

[0033] 作为示例,示例性的寄存器重命名的、乱序发布/执行核架构可按如下方式实现流水线100:1)指令取出138执行取出和长度解码级102和104;2)解码单元140执行解码级106;3)重命名/分配器单元152执行分配级108和重命名级110;4)调度器单元156执行调度级

112;5)物理寄存器组单元158和存储器单元170执行寄存器读取/存储器读取级114;执行群集160执行执行级 116;6)存储器单元170和物理寄存器组单元158执行写回/存储器写入级118; 7)各单元可牵涉到异常处理级122;以及8)引退单元154和物理寄存器组单元 158执行提交级124。

[0034] 核190可支持一个或多个指令集(例如,x86指令集(在更新的版本中加入了一些扩展);加利福尼亚州桑尼维尔市的MIPS技术公司的MIPS指令集;加利福尼州桑尼维尔市的ARM控股公司的ARM指令集(具有诸如NEON之类的可选附加扩展)),其中包括本文中描述的各指令。在一个实施例中,核190 包括用于支持紧缩数据指令集扩展(例如,AVX1、AVX2和/或先前描述的一些形式的通用向量友好指令格式(U=0和/或U=1))的逻辑,从而允许由许多多媒体应用使用的操作能够使用紧缩数据来执行。

[0035] 应当理解,核可支持多线程操作(执行两个或更多个并行的操作或线程的集合),并且可以按各种方式来完成该多线程操作,各种方式包括时分多线程操作、同步多线程操作(其中,单个物理核为物理核正在同步进行多线程操作的多个线程中的每一个线程提供逻辑核)或其组合(例如,时分取出和解码以及此后诸如利用Intel®超线程技术的同步多线程操作)。

[0036] 尽管在乱序执行的上下文中描述了寄存器重命名,但应当理解,可在有序架构中使用寄存器重命名。尽管所示出的处理器的实施例还包括分开的指令和数据高速缓存单元134/174以及共享L2高速缓存单元176,但替代实施例可以具有用于指令和数据两者的单个内部高速缓存,诸如例如一级(L1)内部高速缓存或多个级别的内部高速缓存。在一些实施例中,该系统可包括内部高速缓存和在核和/或处理器外部的外部的高速缓存的组合。或者,所有高速缓存都可在核和/或处理器的外部。

[0037] 图2是根据本发明的多个实施例的、可能具有多于一个的核、可能具有集成存储器控制器、并且可能具有集成图形器件的处理器200的框图。图2中的实线框示出具有单个核202A、系统代理210、一个或多个总线控制器单元216 的集合的处理器200,而虚线框的可选附加示出具有多个核202A-N、系统代理单元210中的一个或多个集成存储器控制器单元214的集合以及专用逻辑208 的替代处理器200。

[0038] 因此,处理器200的不同实现可包括:1)CPU,其中专用逻辑208是集成图形和/或科学(吞吐量)逻辑(其可包括一个或多个核),并且核202A-N是一个或多个通用核(例如,通用有序核、通用乱序核、这两者的组合);2)协处理器,其中核202A-N是旨在主要用于图形和/或科学(吞吐量)的大量专用核;以及3)协处理器,其中核202A-N是大量通用有序核。因此,处理器200 可以是通用处理器、协处理器或专用处理器,该专用处理器诸如例如,网络或通信处理器、压缩引擎、图形处理器、GPGPU(通用图形处理单元)、高吞吐量的集成众核(MIC)协处理器(包括30个或更多核)、嵌入式处理器等。该处理器可以被实现在一个或多个芯片上。处理器200可以是一个或多个基板的一部分,并且/或者可使用多种工艺技术(诸如,BiCMOS、CMOS、或NMOS) 中的任意技术被实现在一个或多个基板上。

[0039] 存储器层次结构包括核内的一个或多个层级的高速缓存240A-N、一组或一个或多个共享高速缓存单元206以及耦合至集成存储器控制器单元的集合 214的外部存储器(未示出)。共享高速缓存单元的集合206可包括一个或多个中级高速缓存,诸如,第二级(L2)、第三级(L3)、第四级(L4)或其他层级的高速缓存、末级高速缓存(LLC)和/或以上的组合。尽

管在一个实施例中，基于环的互连单元212将集成图形逻辑208、共享高速缓存单元的集合206以及系统代理单元210/集成存储器控制器单元214互连，但替代实施例可使用任何数量的公知技术来将此类单元互连。在一个实施例中，可维护一个或多个高速缓存单元206和核202A-N之间的一致性(coherency)。

[0040] 在一些实施例中，一个或多个核202A-N能够进行多线程操作。系统代理 210包括协调和操作核202A-N的那些组件。系统代理单元210可包括例如功率控制单元(PCU)和显示单元。PCU可以是或可包括调节核202A-N和集成图形逻辑208的功率状态所需的逻辑和组件。显示单元用于驱动一个或多个外部连接的显示器。

[0041] 核202A-N在架构指令集方面可以是同构的或异构的；也就是说，这些核 202A-N中的两个或更多个核可能能够执行相同的指令集，而其他核可能能够执行该指令集的仅仅子集或不同的指令集。

[0042] 图3-6是示例性计算机架构的框图。本领域已知的对膝上型计算机、台式机、手持PC、个人数字助理、工程工作站、服务器、网络设备、网络集线器、交换机、嵌入式处理器、数字信号处理器(DSP)、图形设备、视频游戏设备、机顶盒、微控制器、蜂窝电话、便携式媒体播放器、手持设备以及各种其他电子设备的其他系统设计和配置也是合适的。一般地，能够包含本文中所公开的处理器和/或其他执行逻辑的多个系统和电子设备通常都是合适的。

[0043] 现在参考图3，所示出的是根据本发明一个实施例的系统300的框图。系统300可包括一个或多个处理器310、315，这些处理器耦合到控制器中枢320。在一个实施例中，控制器中枢320包括图形存储器控制器中枢(GMCH)390 和输入/输出中枢(IOH)350(其可在分开的芯片上)；GMCH 390包括存储器和图形控制器，存储器340和协处理器345耦合到该存储器和图形控制器；IOH 350将输入/输出(I/O)设备360耦合到GMCH 390。或者，存储器和图形控制器中的一个或两者可以被集成在处理器内(如本文中所描述的)，存储器340 和协处理器345直接耦合到处理器310以及控制器中枢320，该控制器中枢与 IOH 350处于单个芯片中。

[0044] 附加的处理器315的可选性质在图3中通过虚线来表示。每个处理器310、315可包括本文中描述的处理核中的一个或多个，并且可以是处理器200的某一版本。

[0045] 存储器340可以是例如动态随机存取存储器(DRAM)、相变存储器(PCM) 或这两者的组合。对于至少一个实施例，控制器中枢320经由诸如前端总线 (FSB)之类的多分支总线、诸如快速通道互连(QPI)之类的点对点接口、或者类似的连接395与处理器310、315进行通信。

[0046] 在一个实施例中，协处理器345是专用处理器，诸如例如，高吞吐量MIC 处理器、网络或通信处理器、压缩引擎、图形处理器、GPGPU、嵌入式处理器等。在一个实施例中，控制器中枢320可以包括集成图形加速器。

[0047] 在物理资源310、315之间可以存在包括架构、微架构、热、和功耗特征等的一系列品质度量方面的各种差异。

[0048] 在一个实施例中，处理器310执行控制一般类型的数据处理操作的指令。协处理器指令可嵌入在这些指令中。处理器310将这些协处理器指令识别为应当由附连的协处理器345执行的类型。因此，处理器310在协处理器总线或者其他互连上将这些协处理器指令(或者表示协处理器指令的控制信号)发布到协处理器345。协处理器345接受并执行所接收的协处理器指令。

[0049] 现在参考图4，所示为根据本发明的实施例的更具体的第一示例性系统 400的框图。如图4所示，多处理器系统400是点对点互连系统，并且包括经由点对点互连450耦合的第一处理器470和第二处理器480。处理器470和480 中的每一个都可以是处理器200的某一版本。在本发明的一个实施例中，处理器470和480分别是处理器310和315，而协处理器438是协处理器345。在另一实施例中，处理器470和480分别是处理器310和协处理器345。

[0050] 处理器470和480被示出为分别包括集成存储器控制器(IMC)单元472 和482。处理器470也包括作为其总线控制器单元的部分的点对点(P-P)接口 476和478；类似地，第二处理器480包括P-P接口486和488。处理器470、 480可以经由使用点对点(P-P)接口电路450、478的P-P接口450来交换信息。如图4所示，IMC 472和482将处理器耦合到各自的存储器，即存储器432和存储器434，这些存储器可以是本地附连到各自处理器的主存储器的部分。

[0051] 处理器470、480可各自经由使用点对点接口电路476、494、486、498 的各个P-P接口452、454与芯片组490交换信息。芯片组490可以可选地经由高性能接口439与协处理器438交换信息。在一个实施例中，协处理器438 是专用处理器，诸如例如，高吞吐量MIC处理器、网络或通信处理器、压缩引擎、图形处理器、GPGPU、嵌入式处理器等。

[0052] 共享高速缓存(未示出)可以被包括在任一处理器之内，或被包括在两个处理器外部但仍经由P-P互连与这些处理器连接，从而如果将某处理器置于低功率模式时，可将任一处理器或两个处理器的本地高速缓存信息存储在该共享高速缓存中。

[0053] 芯片组490可以经由接口496耦合至第一总线416。在一个实施例中，第一总线416可以是外围组件互连(PCI)总线或诸如PCI高速总线或另一第三代I/O互连总线之类的总线，但是本发明的范围不限于此。

[0054] 如图4所示，各种I/O设备414可连同总线桥418一起耦合到第一总线416，总线桥418将第一总线416耦合到第二总线420。在一个实施例中，诸如协处理器、高吞吐量MIC处理器、GPGPU的处理器、加速器(诸如例如图形加速器或数字信号处理(DSP)单元)、现场可编程门阵列或任何其他处理器的一个或多个附加处理器415耦合到第一总线416。在一个实施例中，第二总线420 可以是低引脚计数(LPC)总线。各种设备可以被耦合至第二总线420，在一个实施例中，这些设备包括例如，键盘/鼠标422、通信设备427以及诸如可包括指令/代码和数据428的盘驱动器或其他大容量存储设备之类的存储单元430。此外，音频I/O 424可以被耦合至第二总线420。注意，其他架构是可能的。例如，代替图4的点对点架构，系统可以实现多分支总线或其他这类架构。

[0055] 现在参考图5，所示为根据本发明的实施例的更具体的第二示例性系统 500的框图。图4和图5中的相同部件用相同附图标记表示，并从图5中省去了图4中的某些方面，以避免使图5的其他方面变得模糊。

[0056] 图5示出处理器470、480可分别包括集成存储器和I/O控制逻辑(“CL”) 472和482。因此，CL 472、482包括集成存储器控制器单元并包括I/O控制逻辑。图5示出不仅存储器432、434耦合至CL 472、482，而且I/O设备514也耦合至控制逻辑472、482。传统I/O设备515被耦合至芯片组490。

[0057] 现在参考图6，所示出的是根据本发明的实施例的SoC 600的框图。图2 中的相似组件具有相同的标号。另外，虚线框是更先进的SoC上的可选特征。在图6中，互连单元602被耦合至：应用处理器610，其包括一个或多个核的集合202A-N以及共享高速缓存单元206；系

统代理单元210;总线控制器单元 216;集成存储器控制器单元214;一组或一个或多个协处理器620,其可包括集成图形逻辑、图像处理器、音频处理器和视频处理器;静态随机存取存储器 (SRAM)单元630;直接存储器存取(DMA)单元632;显示单元640,其以及用于耦合至一个或多个外部显示器。在一个实施例中,协处理器620包括专用处理器,诸如例如,网络或通信处理器、压缩引擎、GPGPU、高吞吐量MIC处理器、嵌入式处理器等。

[0058] 本文公开的机制的各实施例可以被实现在硬件、软件、固件或此类实现方式的组合中。可将本发明的多个实施例实现为在可编程系统上执行的计算机程序或程序代码,该可编程系统包括至少一个处理器、存储系统(包括易失性和非易失性存储器和/或存储元件)、至少一个输入设备以及至少一个输出设备。

[0059] 可将程序代码(诸如图4中示出的代码430)应用于输入指令,以执行本文描述的多个功能并生成输出信息。可以按已知方式将输出信息应用于一个或多个输出设备。为了本申请的目的,处理系统包括具有诸如例如数字信号处理器(DSP)、微控制器、专用集成电路(ASIC)或微处理器之类的处理器的任何系统。

[0060] 程序代码可以用高级程序化语言或面向对象的编程语言来实现,以便与处理系统通信。在需要时,也可用汇编语言或机器语言来实现程序代码。事实上,本文中描述的机制不限于任何特定编程语言的范围。在任何情况下,该语言可以是编译语言或解释语言。

[0061] 至少一个实施例的一个或多个方面可由存储在表示处理器中的各种逻辑的机器可读介质上的表示性指令来实现,当由机器读取这些表示性指令时,这些指令使该机器制作用于执行本文所述的技术的逻辑。可将被称为“IP核”的此类表示存储在有形的机器可读介质上,并将其提供给各种客户或生产设施,以便加载到实际制造该逻辑或处理器的制造机器中。

[0062] 此类机器可读存储介质可以包括但不限于通过机器或设备制造或形成的物品的非瞬态的有形安排,其包括存储介质,诸如:硬盘;任何其他类型的盘,包括软盘、光盘、紧致盘只读存储器(CD-ROM)、紧致盘可重写(CD-RW)以及磁光盘;半导体器件,例如只读存储器(ROM)、诸如动态随机存取存储器(DRAM) 和静态随机存取存储器(SRAM)之类的随机存取存储器(RAM)、可擦除可编程只读存储器(EPROM)、闪存、电可擦除可编程只读存储器(EEPROM);相变存储器(PCM);磁卡或光卡;或适于存储电子指令的任何其他类型的介质。

[0063] 相应地,本发明的多个实施例也包括非瞬态的有形机器可读介质,该介质包含指令或包含定义本文中描述的结构、电路、装置、处理器和/或系统特征的设计数据(例如,硬件描述语言(HDL))。也将此类实施例称为程序产品。

[0064] 在一些情况下,指令转换器可用来将指令从源指令集转换至目标指令集。例如,指令转换器可变换(例如,使用静态二进制变换、包括动态编译的动态二进制变换)、变形、仿真指令或以其他方式将指令转换成将由核来处理的一条或多条其他指令。可在软件、硬件、固件或其组合中实现该指令转换器。指令转换器可在处理器上、在处理器外、或者部分在处理器上且部分在处理器外。

[0065] 图7是根据本发明的多个实施例的对照使用软件指令转换器将源指令集中的二进制指令转换成目标指令集中的二进制指令的框图。在所示的实施例中,指令转换器是软件指令转换器,但也可替代地在软件、固件、硬件或其各种组合中实现该指令转换器。图7示出可使用x86编译器704来编译利用高级语言706的程序,以生成可由具有至少一个x86指令集

核的处理器716原生地执行的x86二进制代码706。具有至少一个x86指令集核的处理器716表示能通过兼容地执行或以其他方式处理以下内容来执行与具有至少一个x86指令集核的英特尔处理器基本相同功能的任何处理器:1)英特尔x86指令集核的指令集的本质部分,或2)目标为在具有至少一个x86指令集核的英特尔处理器上运行以取得与具有至少一个x86指令集核的英特尔处理器基本相同的结果的应用或其他软件的目标代码版本。x86编译器704表示用于生成x86二进制代码706 (例如,目标代码)的编译器,该二进制代码可通过或不通过附加的链接处理在具有至少一个x86指令集核的处理器716上被执行。类似地,图7示出可使用替代的指令集编译器708来编译利用高级语言702的程序,以生成可以由不具有至少一个x86指令集核的处理器714(例如,具有执行加利福尼亚州桑尼维尔市的MIPS技术公司的MIPS指令集和/或执行加利福尼亚州桑尼维尔市的ARM控股公司的ARM指令集的核的处理器)原生地执行的替代指令集二进制代码710。指令转换器712被用来将x86二进制代码706转换成可以由不具有x86指令集核的处理器714原生地执行的代码。该被转换的代码不大可能与替代的指令集二进制代码710相同,因为能够这样做的指令转换器难以制造;然而,被转换的代码将完成一般操作,并且由来自替代指令集中的指令构成。因此,指令转换器712通过仿真、模拟或任何其他过程来表示允许不具有x86 指令集处理器或核的处理器或其他电子设备执行x86二进制代码706的软件、固件、硬件或其组合。

[0066] 用于高效地调用加速器的装置和方法

[0067] 本发明的一个实施例提供用于同步的(例如,固定功能或可编程的)加速器(例如,协处理器、功能单元)的低等待时间调用的通用的可扩展指令(在本文中被称为“XCALL”指令)。在一个实施例中,该指令是x86指令。然而,本发明的基本原理不限于任何指令集架构(ISA)。

[0068] 根据一个实施例,指令格式分别是:XCALL结果寄存器,命令寄存器,参数寄存器,它们分别标识:结果寄存器,其用于存储执行指令之后的结果;命令寄存器,其用于存储将由加速器响应于该指令执行的特定命令和相关联的信息;以及参数寄存器,其用于存储与被调用指令相关联的参数)。下文陈述根据本发明的一个实施例的、被存储在每一个寄存器中的特定信息。

[0069] 图8A示出高层级流,其中,一个或多个处理器群集804执行通用处理操作,而一个或多个加速器群集801执行加速器专用操作。作为示例,通用处理器群集804可包括用于执行指令的处理器核之内的执行逻辑,这些指令(例如,诸如x86指令之类的通用指令)包括调用对加速器群集801的命令的指令。在一个实施例中,加速器群集801的加速器是用于执行专用数据处理操作(例如,向量/SIMD操作、图形操作、排序操作和循环操作等)的协处理器或功能单元。然而,本发明的基本原理不限于任何特定类型的通用核或加速器核。

[0070] 处理器群集804和加速器群集801可以是相同的处理器芯片或核中的逻辑单元。或者,处理器群集804可在一个芯片上,而加速器群集801可在不同的芯片上(在相同的半导体封装中或在不同的封装上)并且经由通信总线(例如,PCI快速总线、直接媒体接口(DMS)总线或其他类型通信总线)被连接。在又一实施例中,加速器群集801中的一些可位于与处理器群集804相同的芯片或核上,而其他加速器群集801可在不同的芯片或核上。本文所描述的本发明的多个实施例不限于任何特定的芯片/封装配置,并且支持具有多个不同类型的加速器群集的实现。

[0071] 如图8A中所示，提供寄存器的集合830以使本文中所述的通用处理器群集804和加速器群集801之间的命令、参数和结果的通信成为可能。具体而言，在一个实施例中，寄存器集合830包括由XCALL指令指定的命令寄存器、结果寄存器和参数寄存器。寄存器集合830可以是可用于下文所指定目的(例如，响应于执行XCALL指令，存储命令、参数数据和结果数据)的通用寄存器 (GPR)。在替代实施例中，这些是专用的应用专用寄存器。

[0072] 在一个实施例中，这些群集执行包括如808处所示的、可使一个或多个加速器被调用的XCALL指令的程序代码806-807、809-810。作为响应，经由寄存器集合830中的命令寄存器(下文中参照图8B进行描述)和/或参数寄存器，向加速器801提供指定要被执行的操作的控制信息。作为响应，加速器可使用一个或多个固定功能单元802和/或可编程功能单元803来执行命令。或者，加速器群集801可用忙碌指示、异常或违背来响应。然后，经由寄存器集合830 之内的结果寄存器(下文中参照图8C进行描述)，向处理器群集804提供这些结果。如果成功地执行了该命令，则可将得到的数据存储在结果寄存器中。相比之下，如果没有成功地执行该命令，则可将指示失败的原因的数据存储在该结果寄存器中(并且可将该数据用于例如确定是否重新尝试执行该命令)。

[0073] 如图8A所示，可在处理器群集上执行一个或多个处理程序805、806。在一个实施例中，如图所示，由处理程序生成的中断可导致对加速器群集的调用。

[0074] 图8B示出命令寄存器结构的一个实施例。如图所示，命令寄存器的顶部 16位(被标识为字段811-815)包含用指定数量的位进行编码的下列数据字段：

[0075] 保留的811:2位

[0076] 继续812:1位

[0077] 反馈(tickle)813:1位

[0078] 私有的814:1位

[0079] Id 815:11位

[0080] 在一个实施例中，该id唯一地标识将调用的加速器。例如，如上所述，多个加速器可被包括在加速器群集801中，并且可通过加速器id码来唯一地标识这些加速器中的每一个。

[0081] 在一个实施例中，该“私有”位指示加速器是否属于特定的已知加速器组。例如，如果该私有位被设置为0，则该id可标识加速器的通用集合(如本专利申请的受让人所定义的那样)中的一个，使得相同的id是指横跨所有计算机系统/处理器的相同加速器。如果该私有位被设置为1，则该id标识专有或库存单元(SKU)专用加速器。因此，该私有位被设置为1时，相同的id可以指不同系统中的不同加速器。

[0082] 在一个实施例中，该命令寄存器的低48位(被标识为8B中的字段816) 以及所有的参数寄存器(没有示出)包含由特定的被调用的加速器定义的应用专用数据。

[0083] 在一个实施例中，当被引退时，XCALL指令集按如下方式设置EFLAGS 中的Z位。如本领域技术人员所理解的那样，EFLAGS是x86实现中包含处理器的当前状态的状态寄存器。如果XCALL完成对所请求的加速器的执行，该 Z位被设置为1。在这种情况下，如果反馈位被设置为1，则结果寄存器不被修改，并且没有实际的工作被完成。如果该反馈位被设置为0，则将该结果寄存器设置为加速器专用值。如果XCALL没有做任何工作，则该Z位被设置为 0。虽然在该实施例中，该Z位用于指示XCALL指令是否成功，但是，可设置不同的位而仍然符合

本发明的基本原理。

[0084] 在一个实施例中，如图8C所示，结果寄存器包含下列数据字段：

[0085] 保留的817:2位(在一个实施例中，总是被设置为零)

[0086] 永久的818:1位

[0087] 私有的819:1位

[0088] 失败细节820:60位

[0089] 在一个实施例中，该永久位818用于指示对相同的XCALL的后续调用是否将成功。例如，该永久位被设置为0指示对相同的XCALL的未来调用可能成功 (例如，如果加速器正忙于服务另一硬件线程)。相比之下，如果重新尝试相同的XCALL没有意义(例如，如果所指定的加速器不存在于当前的SKU中，或者如果所请求的特定的命令和/或参数组合不被该SKU中的加速器支持)，则该永久位被设置为1。

[0090] 在一个实施例中，设置该结果寄存器的低60位以提供关于XCALL失败的原因的附加数据。在一个实施例中，加速器群集801提供更新上述结果寄存器所需的信息。

[0091] 在一个实施例中，如果结果寄存器819的私有位被设置为1，则这些细节具有加速器专用格式。如果该私有位被设置为0，则按预先确定的通用格式(例如，由本专利申请的受让人指定的格式)来提供这些细节。在本发明的一个实施例中采用的示例性失败结果码包括：

[0092] 命令寄存器中的保留位不是0

[0093] 加速器不存在

[0094] 加速器正忙于服务其他线程

[0095] 图9A-C中呈现的流程图示出由本发明的一个实施例执行的操作。在901 处，解码XCALL指令。结果，在902处，将与要由加速器执行的命令有关的数据发送到命令寄存器，并且将任何必要的参数发送到参数寄存器。在903处，取决于加速器是否属于已知的加速器组或专有加速器(如上所述)，在命令寄存器中设置私有位。此外，在903处，在命令寄存器中更新ID码以标识将执行该命令的特定的加速器。

[0096] 在904处，被标识的加速器接收由XCALL指令指定的命令，并且确定是否可执行该命令。例如，该加速器可能当前正忙于服务另一硬件线程，并且因此不能够执行当前的命令。此外，如果所请求的当前命令和/或参数组合不被该加速器支持，则该加速器将不能够成功地执行该命令。或者，该加速器可在904 处成功地执行该命令。

[0097] 如果成功地执行了该命令，则该过程移向图9B，其中，在906处，EFLAGS 的Z位被设置为等于0以指示对该命令的成功执行(如上文中所讨论的那样)。如果在907处确定了该命令寄存器的反馈位先前已被设置为1(例如，在图9A 中的操作902处)，则在908处，结果寄存器保持不被修改。如果该反馈位先前已被设置为0，则在909处，该反馈位被设置为加速器专用值。

[0098] 如果未由加速器成功地执行XCALL指令所指定的命令(在图9A中的905 处进行确定)，则在图9C中的910处，EFLAGS的Z位被设置为等于1(以指示未能执行该命令)。如果在911处确定了预期执行该XCALL指令的未来尝试将是成功的，则在913处，则结果寄存器(图8C中的818)的永久位被设置为0。也可在结果寄存器的失败细节字段820中设置指定失败原因的附加数据。

[0099] 如果在911处预期执行该XCALL指令的未来尝试将是不成功的，则在912 处，该永久位被设置为等于1，(以指示结果的永久性)，并且在结果寄存器的细节字段820中设置与未能执行该XCALL指令有关的附加数据。在上述任意一种情况下，可分析细节字段820中的数据以确定失败的根本原因，并且/或者采取措施来修改指令的执行。

[0100] 如上所述，控制寄存器和/或参数寄存器可能被XCALL指令修改。此外，就像和普通调用一样，XCALL可消耗处理器内的栈区。在使用x86架构的一个实施例中，在XCALL期间(例如，当由异常处理程序检查时)，更新64位栈指针寄存器(RSP)以反映栈的使用。在引退时，该RSP寄存器被恢复为其原始值以反映释放了所使用的栈区。所使用的栈量取决于使用中的特定加速器。

[0101] 在本文所描述的操作序列期间，被调用的加速器可检查和/或修改附加的寄存器和/或存储器位置的值。虽然对于不同的加速器，具体的语义可能不同，但是本发明的基本原理保持相同。

[0102] 在一个实施例中，加速器配置成用于遵循下列一组规则：

[0103] (1)如果在XCALL期间允许中断和/或异常，则继续位被设置为1，并且一旦处理程序完成就重新发布该XCALL，并且执行继续。

[0104] (2)在中断和/或异常存在时，该加速器必须确保向前进展。

[0105] (3)可在所记录的加速器专用位置中更新加速器在中断和/或异常存在的情况下实现向前进展所需要的任何状态，所记录的加速器专用位置可以是下述中的一个或多个：(a)命令和/或参数寄存器；(b)其他架构寄存器；(c)栈区； (d)附加的存储器位置。在上述情况的全部情况下，此类状态必须幸免于诸如来自上下文切换的保存和恢复操作(例如，XSAVE/上下文切换 /XRESTORE)。

[0106] (4)如果加速器被给予“无效的”命令和/或参数寄存器(例如，不被支持的特征、超过硬件限制的值等)，则该加速器可选择永久地拒绝调用。然而，如果加速器已接受调用，则其负责完成该请求并提供结果。

[0107] (5)可编程加速器调用用户码，该用户码可按加速器专用方式(由图8A中的可编程功能单元803表示)受限。例如，“排序”加速器可调用比较功能，而“循环”加速器可调用循环体。如果该用户码不遵循所期望的限制(例如，当使用基于环的分层保护域时，该用户码试图进入环0)，则该加速器将在照例保存其状态之后触发异常(具体而言，UD)。

[0108] (6)异常处理程序可选择：(a)基于所保存的状态，在非加速软件中完成部分地被评价的加速器；(b)仿真不被支持的指令，并且重新发布XCALL (需要微调被保存的状态，因此不重新尝试该不被支持的操作)；或(c)终止执行。仅仅试图重新发布该XCALL而不作任何修改将仅重新触发异常(如针对UD所预期的那样)。

[0109] 本文所描述的本发明的多个实施例提供可结合进诸如x86ISA之类的指令集架构(ISA)以用于调用加速器的标准机制。与本专利申请的背景技术中所描述的技术相比，本文中所描述的加速器调用技术允许自然地共享尽量多(或尽量少)的核资源(例如，存储器转换、寄存器、高速缓存等)的细粒度、低等待时间的同步加速器。可编程XCALL加速器允许用户加速普通的x86代码 (例如，循环和排序)，该代码是主x86程序的集成部分，并且不需要单独的工具链。

[0110] 此外，针对特定的加速器设计当前的加速器接口，而本文中所述的本发明的多个

实施例是可扩展的，进而允许针对特定的市场细分流水线化地供应专用加速器，并且跨越所有的市场细分流水线化地供应“通用”加速器。可在低等待时间和没有数据复制开销的情况下完成加速器调用，从而允许此类加速器的生态系统能够涵盖以前无法实际提供的功能。利用用于特定市场(例如，嵌入式系统、图像处理、HPC服务器等)的加速器来定制SKU，从而保持与诸如 x86之类的现有ISA的紧密集成也变得可能。

[0111] 本文中所描述的XCALL接口也开放无需跳出CPU ISA和工具链(针对本专利申请的受让人所指定的处理器的x86ISA)就可扩展CPU以涵盖先前不可被访问的功能的能力。例如，使用本文中所述的多种技术，可提供可编程加速器803(例如，可编程循环加速器(SKMD)和排序加速器)和固定功能加速器802(例如，执行快速傅里叶变换(FFT)、纹理采样和各种其他功能的那些加速器)。

[0112] 复杂ISA指令的快速失败处理

[0113] 当前，失败指令除了通过通常用于异常处理程序中的专用标记位和/或专用寄存器之外，不具有用于提供关于失败的附加细节的方法。下文所述的本发明的多个实施例提供用于指令的新的“快速失败”行为。按照该新行为，指令可返回成功/失败指示(例如，在诸如EFLAGS或一些其他寄存器之类的标记寄存器内)。此外，在一个实施例中，在检测到失败后，该指令在普通的目的地寄存器中写入附加的失败细节。这允许应用代码能够测试该指令的成功/失败，并且不需要浪费处理资源和时间(这种浪费源自调用异常处理程序或切换到采用分层保护域的系统上的低层级域(例如，环0))就可响应于某些失败模式。

[0114] 针对既易于失败又具有复杂失败模式的某种类型的指令(例如，上文所描述的XCALL指令)选择所提出的针对指令失败处理的新平衡(trade-off)点。然而，这不适用于不容易失败的其他类型的操作(例如，除以零(DIV))，或不适用于具有简单失败模式的容易失败的操作(例如，锁定)。

[0115] 本发明的一个实施例将指令分类为下列多个组中的一个：

[0116] (1)总是成功。例如，预期将两个寄存器中的值相加的指令的每一个实例都将成功。在本发明的一个实施例中，针对本类别中的指令，没有失败处理被提供。

[0117] (2)预期多数时候成功。例如，将存储在两个寄存器中的值相除的指令一般将会成功。它将仅会由于除以零错误而失败。在本发明的一个实施例中，这类指令将触发对失败的异常处理程序。然后，该异常处理程序可检查诸如x86 控制寄存器(CR)之类的包含附加的失败信息的专用寄存器以确定正确的动作过程(例如，用于页错误的CR2)。该异常处理程序和普通应用代码分开，从而保持应用代码干净，并且不受失败处理逻辑污染。

[0118] (3)具有简单失败模式的预期“经常”失败。在一个实施例中，对于这些类型的指令，设置标记和/或目的地寄存器中的位以指示失败，但是没有细节被提供。一个示例是尝试设置锁定数据的指令。对于这些简单的失败模式，应用代码自身显式地处理恢复(不需要异常处理程序)。

[0119] (4)具有复杂失败模式的预期“经常”失败。对于此类指令，处理系统当前需要求助于异常处理程序来访问专用寄存器，以便检查失败细节。对于“经常”失败并且具有复杂失败模式的指令，本发明的多个实施例允许在标记和/ 或目的地寄存器中设置位以指示失败，也在目的地寄存器中设置附加的位以指定该失败的细节，从而允许应用代码不需要求助于异常处理程序就能采取正确的动作。

[0120] 这将失败的代价降到了最低(以必须测试每条指令的结果为代价)。它也允许该应用简单地将其失败处理逻辑定制为当前的上下文,而不是使用难以改变的通用异常处理程序(以必须在任何调用点显式地调用该逻辑为代价)。

[0121] 作为示例,上文中针对XCALL指令描述了该行为。在图9A-C中提供的示例中,XCALL指令指定将由特定的加速器执行的命令。作为响应,该加速器可执行该命令,并且可在结果寄存器(如讨论的那样,其可以是通用寄存器) 中提供结果。或者,该加速器可能出于各种原因无法执行该命令,并且用失败原因来更新该结果寄存器。例如,该加速器可能当前正忙于服务另一硬件线程,并且因此不能够执行当前的命令。在这种情况下,在稍后该加速器不再忙碌的时候,可成功地执行该XCALL指令。由此,响应于该失败指示,在结果寄存器中永久位818被设置为0以指示可进行执行该XCALL指令的第二尝试。

[0122] 相比之下,如果所请求的当前命令和/或参数组合不被该加速器支持,则该加速器将永远不能够成功地执行该命令。由此,响应于该失败指示,在结果寄存器中该永久位818被设置为1以指示第二尝试将不会导致该XCALL指令的成功执行。

[0123] 随后,后续的程序代码可读取该结果寄存器以确定如何继续。例如,如果永久位被设置为0,则后续的程序代码可再次尝试执行该XCALL指令,而如果该永久位被设置为1,则加速器可不尝试执行该XCALL指令。

[0124] 图10是示出用于实现这种模式的操作的本发明的一个实施例的流程图。可由执行单元内的逻辑实现该流程图中指定的操作。在1001处,尝试执行第一指令,并且在1002处,尝试执行第二指令。如果在1003处确定成功地执行了该第一指令,则在1004处,该第二指令也被成功地执行。例如,该第二指令可依赖于被写入到寄存器(例如,上述的结果寄存器)中的第一指令的结果。

[0125] 如果没有成功地执行该第一指令,则在1005处,第二指令也不能执行。与现有实现相比,在1006处检查复杂的失败细节而不调用异常处理程序,从而可由应用程序代码执行失败评估。具体而言,可执行后续的指令以从结果寄存器中读取结果,并且确定是否应当进行对执行该第一指令的新尝试。如果该失败的结果指示第二尝试将不起作用,则可阻止该第二尝试,从而节省时间和处理器资源。如果这些结果指示第二尝试可能成功,则可进行执行该第一指令的第二尝试。尽管为了易于解释,提供了这些具体示例,但是应当注意,本发明的基本原理不限于这些具体细节。

[0126] 因此,在本文所描述的本发明的多个实施例中,指令的普通目的地寄存器用于双重作用;在正常执行的情况下,它们保存结果,并且如果指令失败,则它们保存失败细节。这与当前的实现不同,在当前的实现中,存在用于计算结果和用于失败结果的专用寄存器,并且/或者其中必须调用异常处理程序。这些技术可应用于可编程处理器(GPU、DSP、GPU、…)的所有提供者。

[0127] 使用复杂指令的快速失败处理开放了实现以其他方式将难以定义为高效指令的、诸如XCALL之类的指令的可能性。使用此类高效指令的处理器将实现改善的性能和减少的开发成本。

[0128] 任务可切换同步硬件加速器

[0129] 同步硬件加速器需要确保在异常情况下向前进展;为此,它们需要将其状态保存在幸免于保存和恢复操作(例如,x86架构中的XSAVE/XRESTORE) 的位置中。本发明的一个

实施例通过扩展该保存/恢复区来启用该操作，以便支持新的硬件加速器(例如，上述的和图8A中所示的那些加速器)。

[0130] 本发明的一个实施例将存储器中的栈区用于存储同步硬件加速器的中间状态，以便允许强健的异常模型(包括处理任务切换和核迁移，而没有操作系统(OS)启用)。具体而言，本发明的多个实施例允许诸如同步硬件加速器之类的加速器能够将其状态保存在存储器栈中，并且在各种类型的处理器事件 (例如，下述由异常处理程序管理的异常)之后能够安全地恢复其状态。

[0131] 在一个实施例中，硬件加速器调用被视为CALL指令，在该CALL指令中，该加速器可消耗用户栈上的区域以保存其状态。当异常和/或中断迫使该加速器暂停时，该状态是自动地持久的，并且当在异常处理程序、上下文切换和 /或核迁移之后恢复该加速器时，该状态是可用的。在后一种情况下，恢复计算的硬件加速器可以是不同的加速器(与新核关联的加速器)。在这种情况下，该新核可访问栈中(例如，从存储器或共享高速缓存中)被保存的状态。

[0132] 在一个实施例中，同步加速器被当作被调用的库函数，该同步加速器在调用之后使用栈，随后，当被完成时，释放栈的这部分(表现得像函数调用)。在一个实施例中，当加速器被调用时，栈指针被移动，以便对被调用的加速器的本地变量起作用。当该调用完成时，该栈指针被返回到其最初所在的地方，使得调用程序能够在调用发生时该栈指针停止之处开始。在一个实施例中，假如异常处理程序被调用，则程序的栈指针被调节以反映加速器的栈使用，进而确保该异常处理程序不修改该加速器的保存区。

[0133] 图11中示出本发明的一个实施例，图11示出存储器中的硬件栈1450、应用硬件线程1451和加速器线程1452。图11中所示的特定栈1450包括：调用程序栈1420，用于存储与应用硬件线程1451的执行相关联的数据；加速器保存区1430，用于存储与加速器线程1452的执行相关联的数据；以及异常处理程序栈1440 ，用于存储与异常处理程序1405的执行相关联的数据。

[0134] 在一个实施例中，在应用硬件线程的执行期间，加速器功能被调用。作为响应，栈指针被调节为指向加速器保存区1430的顶部，并且在1401处，锁定转换后备缓冲器(TLB)中与该加速器保存区1430相关联的的条目。这样做的一个原因在于，如果异常发生并且加速器保存其状态(无论是在栈上还是在另一被指定的存储器区中)，期望避免将会将原始异常转换为双倍异常的附加的页错误。避免这样的一种方法是，当加速器开始工作时，锁定用于加速器保存区1430的一个(或多个)TLB页条目，进而确保没有此类页错误将被生成。OS可仍然将该页标记为不可用，但是，但该OS被迫推迟物理地驱逐该页，直到下一个上下文切换(当线程完全不运行，并且加速器状态被安全地保存时) 为止。在从上下文切换返回时，该加速器重新获取这些TLB页条目(其可指向不同的物理位置)，加载状态，并且继续。大型加速器保存区可跨越多个TLB 页(在极端情况下，跨越几十个4k页)。通过使用大型页(例如，64k页)，可减少需要被锁定的TLB条目的数量。

[0135] 在1402处，该加速器基于其正在执行的命令执行操作，并且在1403处，该加速器将其当前的状态保存到栈1450内的加速器保存区1430中。然后，如 1404处所示，该加速器解锁TLB 1404(其在1401处已被锁定以避免上述附加的页错误)。如所示出的那样，检测到异常事件，该异常事件被传送到在应用软件线程1451内被执行的异常处理程序1405中。在执

行期间，该执行处理程序可使用栈1450的部分1440 进行读取/写入(即，它在处理该异常条件期间，将异常处理程序栈1440 用于存储中间状态信息)。一旦该异常处理程序已完成其操作，则它允许加速器线程1452恢复。

[0136] 在1406处，该加速器再次锁定TLB(出于与上述相同的原因)，并且在 1407处，该加速器加载先前已被存储到加速器保存区1430中的状态。注意，在该阶段，加速器线程1452实际上可在与该加速器线程的第一部分(操作 1401-1404)不同的核或处理器上被执行。在此类情况下，该加速器可简单地从加速器保存区1430(其在物理上可位于共享存储器或高速缓冲中)中加载被保存的加速器状态。然后，该加速器在1408处完成其执行线程，在1409处解锁TLB，并且在1410处完成。然后，控制被往回传送到应用硬件线程1451，该应用硬件线程将栈指针重置到加速器保存区1430的顶部(即，当加速器开始执行加速器线程1452时该栈指针停止的地方)。

[0137] 将会理解，可实现对以上提供的具体细节的各种修改而仍然符合本发明的基本原理。例如，在一个实施例中，可为加速器指定特定的存储器区以在其中保存其状态(而不是使用栈)。在这种情况下，没有必要为异常处理程序修改程序的栈指针。

[0138] 在任一实施例中，本文中描述的多种技术允许加速器在调用线程在(对称的)核之间被迁移时透明地工作；一个核上的加速器将其状态保存到存储器中，并且当该线程在另一核上被调度时，那里的加速器从存储器中(例如，为了高效，经由共享公共高速缓存)加载该数据。因此，本文中描述的本发明的多个实施例在存在异常、上下文切换和/或核迁移并且不启用OS(例如，不修改 XSAVE/XRESTORE和/或添加架构寄存器)的情况下，允许加速器透明地保存其状态并确保向前进展。这转而允许使用先前需要经由被修改的XSAVE来添加新架构寄存器并启用OS的加速器形式。使用此类加速器的的处理器实现改善的性能和减少的开发成本。

[0139] 示例性指令格式

[0140] 能以不同的格式使本文所述的指令的多个实施例具体化。另外，在下文中详述示例性系统、架构和流水线。指令的实施例可在此类系统、架构和及流水线上执行，但是不限于详述的系统、架构和流水线。

[0141] 向量友好指令格式是适于向量指令(例如，存在专用于向量操作的某些字段)的指令格式。尽管描述了其中通过向量友好指令格式支持向量和标量操作两者的实施例，但是替代实施例仅使用通过向量友好指令格式的向量操作。

[0142] 图12A-12B是示出根据本发明的实施例的通用向量友好指令格式及其指令模板的框图。图12A是示出根据本发明的多个实施例的通用向量友好指令格式及其A类指令模板的框图；而图12B是示出根据本发明的多个实施例的通用向量友好指令格式及其B类指令模板的框图。具体而言，针对通用向量友好指令格式1100定义A类和B类指令模板，两者都包括无存储器访问1105的指令模板和存储器访问1120的指令模板。在向量友好指令格式的上下文中的术语“通用”是指不束缚于任何特定指令集的指令格式。

[0143] 尽管将描述其中向量友好指令格式支持以下情况的本发明的实施例，但是替代实施例可支持更大、更小、和/或不同的向量操作数尺寸(例如，256字节向量操作数)与更大、更小或不同的数据元素宽度(例如，128位(16字节) 数据元素宽度)：64字节向量操作数长度(或尺寸)与32位(4字节)或64 位(8字节)数据元素宽度(或尺寸)(并且由此，64字节向量

由16个双字尺寸的元素或者替代地8个四字尺寸的元素组成)、64字节向量操作数长度(或尺寸)与16位(2字节)或8位(1字节)数据元素宽度(或尺寸)、32字节向量操作数长度(或尺寸)与32位(4字节)、64位(8字节)、16位(2字节)、或8位(1字节)数据元素宽度(或尺寸)、以及16字节向量操作数长度(或尺寸)与32位(4字节)、64位(8字节)、16位(2字节)、或8位(1字节) 数据元素宽度(或尺寸),但是替代实施例可支持更大、更小、和/或不同的向量操作数尺寸(例如,256字节向量操作数)与更大、更小或不同的数据元素宽度(例如,128位(16字节)数据元素宽度)。

[0144] 图12A中的A类指令模板包括:1)在无存储器访问1105的指令模板内,示出无存储器访问的完全舍入控制型操作1110的指令模板以及无存储器访问的数据变换型操作1115的指令模板;以及2)在存储器访问1120的指令模板内,示出存储器访问的时效性1125的指令模板和存储器访问的非时效性1130的指令模板。图11B中的B类指令模板包括:1)在无存储器访问1105的指令模板内,示出无存储器访问的写掩码控制的部分舍入控制型操作1112的指令模板以及无存储器访问的写掩码控制的vsize型操作1117的指令模板;以及2)在存储器访问1120的指令模板内,示出存储器访问的写掩码控制1127的指令模板。

[0145] 通用向量友好指令格式1100包括以下列出的按照在图12A-12B中示出的顺序的如下字段。

[0146] 格式字段1140－该字段中的特定值(指令格式标识符值)唯一地标识向量友好指令格式,并且由此标识指令在指令流中以向量友好指令格式出现。由此,该字段对于仅具有通用向量友好指令格式的指令集是不需要的,在这个意义上该字段是可选的。

[0147] 基础操作字段1142－其内容区分不同的基础操作。

[0148] 寄存器索引字段1144-其内容直接或者通过地址生成来指定源或目的地操作数在寄存器中或者在存储器中的位置。这些字段包括从PxQ(例如,32x512、16x128、32x1024、64x1024)寄存器组中选择N个寄存器的足够数量的位。尽管在一个实施例中N可多至三个源和一个目的地寄存器,但是替代实施例可支持更多或更少的源和目的地寄存器(例如,可支持多至两个源(其中,这些源中的一个源还用作目的地),可支持多至三个源(其中,这些源中的一个源还用作目的地),可支持多至两个源和一个目的地)。

[0149] 修饰符(modifier)字段1146－其内容将指定存储器访问的以通用向量指令格式出现的指令与不指定存储器访问的以通用向量指令格式出现的指令区分开;即在无存储器访问1105的指令模板与存储器访问1120的指令模板之间进行区分。存储器访问操作读取和/或写入到存储器层次结构(在一些情况下,使用寄存器中的值来指定源和/或目的地地址),而非存储器访问操作不这样(例如,源和/或目的地是寄存器)。尽管在一个实施例中,该字段还在三种不同的方式之间选择以执行存储器地址计算,但是替代实施例可支持更多、更少或不同的方式来执行存储器地址计算。

[0150] 扩充操作字段1150－其内容区分除基础操作以外还要执行各种不同操作中的哪一个操作。该字段是针对上下文的。在本发明的一个实施例中,该字段被划分成类字段1168、α字段1152、以及β字段1154。扩充操作字段1150允许在单条指令而非2、3或4条指令中执行多组共同的操作。

[0151] 比例字段1160－其内容允许用于存储器地址生成(例如,用于使用$2^{比例}$* 索引+基址的地址生成)的索引字段的内容的按比例缩放。

[0152] 位移字段1162A－其内容用作存储器地址生成的一部分(例如,用于使用 $2^{比例}$*索引+基址+位移的地址生成)。

[0153] 位移因数字段1162B(注意,位移字段1162A直接在位移因数字段1162B 上的并置指示使用一个或另一个)－其内容用作地址生成的一部分,它指定通过存储器访问的尺寸(N)按比例缩放的位移因数,其中N是存储器访问中的字节数量(例如,用于使用$2^{比例}$*索引+基址+按比例缩放的位移的地址生成)。忽略冗余的低阶位,并且因此将位移因数字段的内容乘以存储器操作数总尺寸 (N)以生成在计算有效地址中使用的最终位移。N的值由处理器硬件在运行时基于完整操作码字段1174(在本文中描述的)和数据操纵字段1154C确定。位移字段1162A和位移因数字段1162B可以不用于无存储器访问1105的指令模板,并且/或者不同的实施例可实现两者中的仅一个或不实现两者中的任一个,在这个意义上,位移字段1162A和位移因数字段1162B是可选的。

[0154] 数据元素宽度字段1164－其内容区分将使用多个数据元素宽度中的哪一个(在一些实施例中用于所有指令,在其他实施例中仅用于指令中的一些)。如果支持仅一个数据元素宽度,并且/或者使用操作码的某一方面来支持数据元素宽度,则该字段是不需要的,在这个意义上该字段是可选的。

[0155] 写掩码字段1170－其内容在每一数据元素位置的基础上控制目的地向量操作数中的数据元素位置是否反映基础操作和扩充操作的结果。A类指令模板支持合并-写掩码操作,而B类指令模板支持合并写掩码操作和归零写掩码操作两者。当合并时,向量掩码允许在执行(由基础操作和扩充操作指定的)任何操作期间保护目的地中的任何元素集免于更新;在其他实施例中,保持其中对应掩码位具有0的目的地的每一元素的旧值。相反,当归零时,向量掩码允许在执行(由基础操作和扩充操作指定的)任何操作期间,使目的地中的任何元素集归零;在一个实施例中,当对应掩码位具有0值时,将目的地的元素设置为0。该功能的子集是控制正在被执行的操作的向量长度的能力(即,从第一个到最后一个被修改的元素的跨度),然而,被修改的元素不一定要是连续的。由此,写掩码字段1170允许部分向量操作,这包括加载、存储、算术、逻辑等。尽管描述了其中写掩码字段1170的内容选择了多个写掩码寄存器中的包含要使用的写掩码的一个写掩码寄存器(并且由此写掩码字段1170的内容间接地标识了要执行的掩码操作)的本发明的实施例,但是替代实施例相反或另外允许掩码写字段1170的内容直接地指定要执行的掩码操作。

[0156] 立即数字段1172－其内容允许对立即数的指定。该字段在不支持立即数的通用向量友好格式的实现中不存在,并且在不使用立即数的指令中不存在,在这个意义上该字段是可选的。

[0157] 类字段1168－其内容在不同类的指令之间进行区分。参考图12A-B,该字段的内容在A类和B类指令之间进行选择。在图12A-B中,圆角方形用于指示专用值存在于字段中(例如,在图12A-B中分别用于类字段1168的A类 1168A和B类1168B)。

[0158] A类指令模板

[0159] 在A类非存储器访问1105的指令模板的情况下,α字段1152被解释为其内容区分要执行不同扩充操作类型中的哪一种(例如,针对无存储器访问的舍入型操作1110和无存储器访问的数据变换型操作1115的指令模板,分别指定舍入1152A.1和数据变换1152A.2)的RS字段1152A,而β字段1154区分要执行指定类型的操作中的哪一种。在无存储器访问1105

指令模板中，比例字段1160、位移字段1162A以及位移比例字段1162B不存在。

[0160] 无存储器访问的指令模板－完全舍入控制型操作

[0161] 在无存储器访问的完全舍入控制型操作1110的指令模板中，β字段1154 被解释为其内容提供静态舍入的舍入控制字段1154A。尽管在本发明的所述实施例中，舍入控制字段1154A包括抑制所有浮点异常(SAE)字段1156和舍入操作控制字段1158，但是替代实施例可支持这两个概念，并且可将这两个概念都编码成相同的字段，或者仅具有这些概念/字段中的一个或另一个(例如，可仅具有舍入操作控制字段1158)。

[0162] SAE字段1156－其内容区分是否禁用异常事件报告；当SAE字段1156 的内容指示启用抑制时，给定的指令不报告任何种类的浮点异常标志，并且不唤起任何浮点异常处理程序。

[0163] 舍入操作控制字段1158－其内容区分执行一组舍入操作中的哪一个(例如，向上舍入、向下舍入、向零舍入、以及就近舍入)。由此，舍入操作控制字段1158允许逐指令地改变舍入模式。在其中处理器包括用于指定舍入模式的控制寄存器的本发明的一个实施例中，舍入操作控制字段1158的内容覆盖该寄存器值。

[0164] 无存储器访问的指令模板－数据变换型操作

[0165] 在无存储器访问的数据变换型操作1115的指令模板中，β字段1154被解释为数据变换字段1154B，其内容区分要执行多个数据变换中的哪一个(例如，无数据变换、混合、广播)。

[0166] 在A类存储器访问1120的指令模板的情况下，α字段1152被解释为驱逐提示字段1152B，其内容区分要使用驱逐提示中的哪一个(在图12A中，对于存储器访问时效性1125的指令模板和存储器访问非时效性1130的指令模板分别指定时效性的1152B.1和非时效性的1152B.2)，而β字段1154被解释为数据操纵字段1154C，其内容区分要执行多个数据操纵操作(也称为基元 (primitive))中的哪一个(例如，无操纵、广播、源的向上转换以及目的地的向下转换)。存储器访问1120的指令模板包括比例字段1160，并可选地包括位移字段1162A或位移比例字段1162B。

[0167] 向量存储器指令使用转换支持来执行来自存储器的向量加载和去往存储器的向量存储。如同寻常的向量指令，向量存储器指令以数据元素式的方式往返于存储器传输数据，其中，实际传输的元素由被选为写掩码的向量掩码的内容规定。

[0168] 存储器访问的指令模板－时效性的

[0169] 时效性的数据是可能足够快地被重新使用以从高速缓存操作中受益的数据。然而，这是提示，且不同的处理器能以不同的方式实现它，包括完全忽略该提示。

[0170] 存储器访问的指令模板－非时效性的

[0171] 非时效性的数据是不可能被足够快地重新使用以从第一级高速缓存中的高速缓存操作中受益且应当被给予驱逐优先级的数据。然而，这是提示，且不同的处理器可以不同的方式实现它，包括完全忽略该提示。

[0172] B类指令模板

[0173] 在B类指令模板的情况下，α字段1152被解释为写掩码控制(Z)字段 1152C，其内容区分由写掩码字段1170控制的写掩码操作应当是合并还是归零。

[0174] 在B类非存储器访问1105的指令模板的情况下，β字段1154的部分被解释为RL字段

1157A,其内容区分要执行不同扩充操作类型中的哪一种(例如,针对无存储器访问的写掩码控制部分舍入控制类型操作1112的指令模板和无存储器访问的写掩码控制VSIZE型操作1117的指令模板,分别指定舍入 1157A.1和向量长度(VSIZE)1157A.2),而β字段1154的其余部分区分要执行指定类型的操作中的哪一种。在无存储器访问1105指令模板中,比例字段 1160、位移字段1162A以及位移比例字段1162B不存在。

[0175] 在无存储器访问的写掩码控制的部分舍入控制型操作1110的指令模板中,β字段1154的其余部分被解释为舍入操作字段1159A,并且禁用异常事件报告(给定的指令不报告任何种类的浮点异常标志,并且不唤起任何浮点异常处理程序)。

[0176] 舍入操作控制字段1159A－就如同舍入操作控制字段1158,其内容区分一组舍入操作中的哪一个(例如,向上舍入、向下舍入、向零舍入、以及就近舍入)要执行。由此,舍入操作控制字段1159A允许逐指令地改变舍入模式。在其中处理器包括用于指定舍入模式的控制寄存器的本发明的一个实施例中,舍入操作控制字段1159A的内容覆盖该寄存器值。

[0177] 在无存储器访问的写掩码控制VSIZE型操作1117的指令模板中,β字段 1154的其余部分被解释为向量长度字段1159B,其内容区分要执行多个数据向量长度中的哪一个(例如,128字节、256字节或512字节)。

[0178] 在B类存储器访问1120的指令模板的情况下,β字段1154的部分被解释为广播字段1157B,其内容区分是否要执行广播型数据操纵操作,而β字段1154 的其余部分被解释为向量长度字段1159B。存储器访问1120的指令模板包括比例字段1160,并可选地包括位移字段1162A或位移比例字段1162B。

[0179] 针对通用向量友好指令格式1100,示出完整操作码字段1174包括格式字段1140、基础操作字段1142以及数据元素宽度字段1164。尽管示出了其中完整操作码字段1174包括所有这些字段的一个实施例,但是在不支持所有这些字段的实施例中,完整操作码字段1174包括少于所有这些字段的字段。完整操作码字段1174提供操作码(opcode)。

[0180] 扩充操作字段1150、数据元素宽度字段1164以及写掩码字段1170允许以通用向量友好指令格式逐指令地指定这些特征。

[0181] 写掩码字段和数据元素宽度字段的组合创建类型化的指令,因为它们允许基于不同的数据元素宽度应用该掩码。

[0182] 在A类和B类内出现的各种指令模板在不同的情形下是有益的。在本发明的一些实施例中,不同处理器或者处理器内的不同核可支持仅A类、仅B 类或者可支持两类。举例而言,旨在用于通用计算的高性能通用乱序核可仅支持B类,旨在主要用于图形和/或科学(吞吐量)计算的核可仅支持A类,并且旨在用于两者的核可支持两者(当然,具有来自两类的模板和指令的一些混合、但是并非来自两类的所有模板和指令的核在本发明的范围内)。同样,单一处理器可包括多个核,所有核支持相同的类或者其中不同的核支持不同的类。举例而言,在具有单独的图形和通用核的处理器中,旨在主要用于图形和 /或科学计算的图形核中的一个核可仅支持A类,而通用核中的一个或多个可以是具有旨在用于通用计算的、仅支持B类的乱序执行和寄存器重命名的高性能通用核。不具有单独的图形核的另一处理器可包括既支持A类又支持B类的一个或多个通用有序或乱序核。当然,在本发明的不同实施例中,来自一类的特征也可在其他类中实现。可使以高级语言撰写的程序成为(例如,恰被及时编译或静态编译)各种不同的可执行形式,包括:1)仅具有由用于执行的目标处理器支

持的类的指令的形式;或者2)具有使用所有类的指令的不同组合而编写的替代例程且具有选择这些例程以基于由当前正在执行代码的处理器支持的指令而执行的控制流代码的形式。

[0183] 图13A是示出根据本发明的多个实施例的示例性专用向量友好指令格式的框图。图13A示出专用向量友好指令格式1200,其指定位置、尺寸、解释和字段的次序以及那些字段中的一些字段的值,在这个意义上向量友好指令格式1200是专用的。专用向量友好指令格式1200可用于扩展x86指令集,并且由此这些字段中的一些与现有x86指令集及其扩展(例如,AVX)中使用的那些字段类似或相同。该格式保持与具有扩展的现有x86指令集的前缀编码字段、实操作码字节字段、MOD R/M字段、SIB字段、位移字段、以及立即数字段一致。示出来自图12的、将来自图13的字段映射到其的字段。

[0184] 应当理解,虽然出于说明的目的,在通用向量友好指令格式1100的上下文中参考专用向量友好指令格式1200描述了本发明的多个实施例,但是本发明不限于专用向量友好指令格式1200,除非另有声明。例如,通用向量友好指令格式1100构想各种字段的各种可能的尺寸,而专用向量友好指令格式1200 被示出为具有特定尺寸的字段。作为具体示例,尽管在专用向量友好指令格式 1200中,数据元素宽度字段1164被示出为一位的字段,但是本发明不限于此 (也就是说,通用向量友好指令格式1100构想数据元素宽度字段1164的其他尺寸)。

[0185] 通用向量友好指令格式1200包括以下按照图13A中示出的顺序列出的下列字段。

[0186] EVEX前缀(字节0-3)1202－以四字节形式进行编码。

[0187] 格式字段1140(EVEX字节0,位[7:0])－第一字节(EVEX字节0)是格式字段1140,并且它包含0x62(在本发明的一个实施例中用于区分向量友好指令格式的唯一值)。

[0188] 第二－第四字节(EVEX字节1-3)包括提供专用能力的多个位字段。

[0189] REX字段1205(EVEX字节1,位[7-5])－由EVEX.R位字段(EVEX 字节1,位[7]-R)、EVEX.X位字段(EVEX字节1,位[6]-X)以及(BEX字节1,位[5]-B)组成。EVEX.R、EVEX.X和EVEX.B位字段提供与对应VEX 位字段相同的功能,并且使用1补码的形式进行编码,即ZMM0被编码为 1111B,ZMM15被编码为0000B。这些指令的其他字段对如在本领域中已知的寄存器索引的较低三个位(rrr、xxx以及bbb)进行编码,由此可通过增加 EVEX.R、EVEX.X以及EVEX.B来形成Rrrr、Xxxx以及Bbbb。

[0190] REX'字段1210－这是REX'字段1210的第一部分,并且是用于对扩展的 32个寄存器集合的较高16个或较低16个寄存器进行编码的EVEX.R'位字段 (EVEX字节1,位[4]-R')。在本发明的一个实施例中,该位与以下指示的其他位一起以位反转的格式被存储以(在公知x86的32位模式下)与实操作码字节是62的BOUND指令进行区分,但是在MOD R/M字段(在下文中描述) 中不接受MOD字段中的值11;本发明的替代实施例不以反转的格式存储该以下其他被指示的位。值1用于对较低16个寄存器进行编码。换句话说,通过组合EVEX.R'、EVEX.R、以及来自其他字段的其他RRR来形成R'Rrrr。

[0191] 操作码映射字段1215(EVEX字节1,位[3:0]-mmmm)-其内容对隐含的前导操作码字节(0F、0F 38、或0F 3)进行编码。

[0192] 数据元素宽度字段1164(EVEX字节2,位[7]-W)－由记号EVEX.W表示。EVEX.W用于定义数据类型(32位数据元素或64位数据元素)的粒度(尺寸)。

[0193] EVEX.vvvv 1220(EVEX字节2,位[6:3]-vvvv)－EVEX.vvvv的作用可包括如下:1)以反转(1补码)形式被指定并且对具有2个或更多源操作数的指令有效VEX.vvvv对第一源寄存器操作数进行编码;2)针对某些向量偏移以 1补码形式被指定的VEX.vvvv对目的地寄存器操作数进行编码;或者3) VEX.vvvv不对任何操作数进行编码,保留该字段,并且该字段应当包含1111b。由此,EVEX.vvvv字段1220对以反转(1补码)的形式存储的第一源寄存器指定符的4个低阶位进行编码。取决于该指令,附加的不同的EVEX位字段用于将指定符尺寸扩展到32个寄存器。

[0194] EVEX.U 1168类字段(EVEX字节2,位[2]-U)－如果EVEX.U＝0,则它指示A类或EVEX.U0;如果EVEX.U＝1,则它指示B类或EVEX.U1。

[0195] 前缀编码字段1225(EVEX字节2,位[1:0]-pp)－提供了用于基础操作字段的附加位。除了对以EVEX前缀格式的传统SSE指令提供支持以外,这也具有压缩SIMD前缀的益处(EVEX前缀只需要2位,而不是需要字节来表达SIMD前缀)。在一个实施例中,为了支持使用以传统格式和以EVEX前缀格式两者的SIMD前缀(66H、F2H、F3H)的传统SSE指令,将这些传统SIMD 前缀编码为SIMD前缀编码字段;在提供给解码器的PLA之前,在运行时可被扩展为传统SIMD前缀(因此,PLA可执行传统和EVEX格式的这些传统指令,而无需修改)。虽然较新的指令可将EVEX前缀编码字段的内容直接用作操作码扩展,但是为了一致性,某些实施例以类似的方式扩展,但允许由这些传统SIMD前缀指定不同的含义。替代实施例可重新设计PLA以支持2位 SIMD前缀编码,并且因此不需要扩展。

[0196] α字段1152(EVEX字节3,位[7]-EH,也称为EVEX.EH、EVEX.rs、 EVEX.RL、EVEX.写掩码控制、以及EVEX.N;也以α示出)－如先前所述,该字段是针对上下文的。

[0197] β字段1154(EVEX字节3,位[6:4]-SSS,也称为EVEX.s_{2-0}、EVEX.r_{2-0}、 EVEX.rr1、EVEX.LL0、EVEX.LLB;也以βββ示出)－如先前所述,该字段是针对上下文的。

[0198] REX'字段1210－这是REX'字段的其余部分,并且是可用于对扩展的32 个寄存器集合的较高16个或较低16个寄存器进行编码的EVEX.V'位字段 (EVEX字节3,位[3]-V')。该位以位反转的格式存储。值1用于对较低16 个寄存器进行编码。换句话说,通过组合EVEX.V'、EVEX.vvvv来形成 V'VVVV。

[0199] 写掩码字段1170(EVEX字节3,位[2:0]-kkk)－其内容指定写掩码寄存器中的寄存器索引,如先前所述。在本发明的一个实施例中,特定值EVEX.kkk＝000具有暗示没有写掩码用于特定指令的特殊行为(这能以各种方式实现,包括使用硬连线到所有的写掩码或者绕过掩码硬件的硬件来实现)。

[0200] 实操作码字段1230(字节4)也被称为操作码字节。在该字段中指定操作码的部分。

[0201] MOD R/M字段1240(字节5)包括MOD字段1242、Reg字段1244以及 R/M字段1246。如先前所述的,MOD字段1242的内容将存储器访问和非存储器访问操作区分开。Reg字段1244的作用可被归结为两种情形:对目的地寄存器操作数或源寄存器操作数进行编码;或者被视为操作码扩展且不用于对任何指令操作数进行编码。R/M字段1246的作用可包括如下:对引用存储器地址的指令操作数进行编码;或者对目的地寄存器操作数或源寄存器操作数进行编码。

[0202] 比例、索引、基址(SIB)字节(字节6)－如先前所述的,比例字段1250 的内容用于存储器地址生成。SIB.xxx 1254和SIB.bbb 1256－先前已经针对寄存器索引Xxxx和Bbbb提

及了这些字段的内容。

[0203] 位移字段1162A(字节7-10)－当MOD字段1242包含10时，字节7-10 是位移字段1162A，并且它以与传统32位位移(disp32)相同的方式工作，以字节粒度工作。

[0204] 位移因数字段1162B(字节7)－当MOD字段1242包含01时，字节7 是位移因数字段1162B。该字段的位置与以字节粒度工作的传统x86指令集8 位位移(disp8)的位置相同。由于disp8是符号扩展的，因此它仅能在-128和 127字节偏移量之间寻址；在64字节高速缓存行的方面，disp8使用可被设为仅四个真正有用的值-128、-64、0和64的8位；由于常常需要更大的范围，所以使用disp32；然而，disp32需要4个字节。与disp8和disp32对比，位移因数字段1162B是对disp8的重新解释；当使用位移因数字段1162B时，通过将位移因数字段的内容乘以存储器操作数访问的尺寸(N)来确定实际位移。该类型的位移被称为disp8*N。这减小了平均指令长度(单个字节用于位移，但具有大得多的范围)。此类压缩位移基于有效位移是存储器访问的粒度的倍数的假设，并且由此，不需要对地址偏移量的冗余低阶位进行编码。换句话说，位移因数字段1162B替代传统x86指令集的8位位移。由此，以与x86指令集的8位位移相同的方式对位移因数字段1162B进行编码(因此，在ModRM/SIB 编码规则中没有变化)，唯一的例外在于，将disp8超载至disp8*N。换句话说，编码规则或编码长度中不存在变化，而仅在通过硬件对位移值的解释中存在变化(这需要通过存储器操作数的尺寸按比例缩放位移量以获得字节式地址偏移量)。

[0205] 立即数字段1172按先前所述进行操作。

[0206] 完整操作码字段

[0207] 图13B是示出根据本发明的一个实施例的、专用向量友好指令格式1200 中构成完整操作码字段1174字段的框图。具体而言，完整操作码字段1174包括格式字段1140、基础操作字段1142以及数据元素宽度(W)字段1164。基础操作字段1142包括前缀编码字段1225、操作码映射字段1215以及实操作码字段1230。

[0208] 寄存器索引字段

[0209] 图13C是示出根据本发明的一个实施例的、专用向量友好指令格式1200 中构成寄存器索引字段1144的字段的框图。具体而言，寄存器索引字段1144 包括REX字段1205、REX'字段1210、MODR/M.reg字段1244、MODR/M.r/m 字段1246、VVVV字段1220、xxx字段1254以及bbb字段1256。

[0210] 扩充操作字段

[0211] 图13D是示出根据本发明的一个实施例、专用向量友好指令格式1200中构成扩充操作字段1150的字段的框图。当类(U)字段1168包含0时，它表明EVEX.U0(A类1168A)；当它包含1时，它表明EVEX.U1(B类1168B)。当U＝0且MOD字段1242包含11(表明无存储器访问操作)时，α字段1152 (EVEX字节3，位[7]-EH)被解释为rs字段1152A。当rs字段1152A包含1(舍入1152A.1)时，β字段1154(EVEX字节3，位[6:4]-SSS)被解释为舍入控制字段1154A。舍入控制字段1154A包括一位的SAE字段1156和两位的舍入操作字段1158。当rs字段1152A包含0(数据变换1152A.2)时，β字段1154(EVEX字节3，位[6:4]-SSS)被解释为三位的数据变换字段1154B。当 U＝0且MOD字段1242包含00、01或10(表明存储器访问操作)时，α字段 1152(EVEX字节3，位[7]-EH)被解释为驱逐提示(EH)字段1152B且β字段1154(EVEX字节3，位[6:4]-SSS)被解释为三位数据操纵字段1154C。

[0212] 当U＝1时，α字段1152(EVEX字节3，位[7]-EH)被解释为写掩码控制 (Z)字段1152C。当U＝1且MOD字段1242包含11(表明无存储器访问操作)时，β字段1154的一部分(EVEX字节3，位[4]-S_0)被解释为RL字段 1157A；当它包含1(舍入1157A.1)时，β字段1154的其余部分(EVEX字节 3，位[6-5]-S_{2-1})被解释为舍入操作字段1159A，而当RL字段1157A包含0 (VSIZE 1157.A2)时，β字段1154的其余部分(EVEX字节3，位[6-5]-S_{2-1}) 被解释为向量长度字段1159B(EVEX字节3，位[6-5]-L_{1-0})。当U＝1且MOD 字段1242包含00、01或10(表明存储器访问操作)时，β字段1154(EVEX 字节3，位[6:4]-SSS)被解释为向量长度字段1159B(EVEX字节3，位[6-5]-L_{1-0}) 和广播字段1157B(EVEX字节3，位[4]-B)。

[0213] 图14是根据本发明的一个实施例的寄存器架构1300的框图。在所示出的实施例中，有32个512位宽的向量寄存器1310；这些寄存器被引用为zmm0 到zmm31。较低的16个zmm寄存器的较低阶256个位覆盖在寄存器ymm0-15 上。较低的16个zmm寄存器的较低阶128个位(ymm寄存器的较低阶128 个位)覆盖在寄存器xmm0-15上。专用向量友好指令格式1200按下表所示，对这些覆盖的寄存器组进行操作。

[0214]

可调节向量长度	类	操作	寄存器
不包括向量长度字段 1159B 的指令模板	A(图 12A；U=0)	1110、1115、1125、1130	zmm 寄存器（向量长度是 64 字节）
	B(图 12B；U=1)	1112	zmm 寄存器（向量长度是 64 字节）
包括向量长度字段 1159B 的指令模板	B(图 12B；U=1)	1117、1127	zmm、ymm、或 xmm 寄存器（向量长度是 64 字节、32 字节、或 16 字节），取决于向量长度字段 1159B

[0215] 换句话说，向量长度字段1159B在最大长度与一个或多个其他较短长度 (其中，此类较短长度的长度是前一个长度的一半)之间进行选择；不具有向量长度字段1159B的指令模板对最大向量长度进行操作。此外，在一个实施例中，专用向量友好指令格式1200的B类指令模板对紧缩或标量单/双精度浮点数据以及紧缩或标量整数数据进行操作。标量操作是对zmm/ymm/xmm寄存器中的最低阶数据元素位置执行的操作；取决于本实施例，较高阶数据元素位置保持与在指令之前相同或者归零。

[0216] 写掩码寄存器1315－在所示的实施例中，存在8个写掩码寄存器(k0至k7)，每一写掩码寄存器的尺寸是64位。在替代实施例中，写掩码寄存器1315 的尺寸为16位。如先前所述的，在本发明的一个实施例中，向量掩码寄存器 k0不能用作写掩码；当正常指示k0的编码用作写掩码时，它选择硬连线的写掩码0xFFFF，从而有效地禁用该指令的写掩码操作。

[0217] 通用寄存器1325——在所示出的实施例中，有十六个64位通用寄存器，这些寄存器结合现有的x86寻址模式，用于寻址存储器操作数。这些寄存器通过名称RAX、RBX、RCX、RDX、RBP、RSI、RDI、RSP以及R8到R15来引用。

[0218] 标量浮点栈寄存器组(x87栈)1345，在其上面重叠了MMX紧缩整数平坦寄存器组1350——在所示出的实施例中，x87栈是用于使用x87指令集扩展来对32/64/80位浮点数据执行标量浮点操作的八元素栈；而将MMX寄存器用于64位紧缩整数数据执行操作，以及用于为在MMX和XMM寄存器之间执行的一些操作保存操作数。

[0219] 本发明的替代实施例可以使用更宽的或更窄的寄存器。

[0220] 另外，本发明的替代实施例可使用更多、更少或不同的寄存器组和寄存器。

[0221] 示例性计算机系统

[0222] 图15是说明可用在本发明的一些实施例中的示例性客户机和服务器的框图。应当注意，尽管图15示出数据处理系统的各种组件，但是，其并不旨在表示互连这些组件的任何特定架构或方式，因为此类细节不是与本发明的多个实施例有密切关系的。应当理解，具有更少或更多组件的其他计算机系统也可与本发明的多个实施例一起被使用。

[0223] 如图15所示，数据处理系统形式的计算机系统1500包括互联/总线1501，该互联/总线1501通信地将处理器群集804耦合至各种其他系统组件。该互联/ 总线可包括可通过本领域中公知的各种桥、控制器和/或适配器被彼此连接的各个层级的互联。作为示例，互联1501可包括快速路径互连(QPI)组件、外围组件互联快速(“PCI快速”)组件或用于将各种组件互联到处理器群集804的其他技术。本发明的基本原理不限于任何特定的互联或总线。

[0224] 尽管在图15中被示出为单独的组件，但是加速器801也可被集成在处理器群集804中。或者，一些加速器可被集成在处理器群集中，而一些经由互连 /总线被连接到计算机系统。如上文中详述的那样，这些加速器适用于高效地执行某些类型的程序代码(例如，向量/SIMD操作、图形操作、排序和循环操作等)。作为示例，通用处理器群集804可包括用于执行通用指令的处理器核之内的执行逻辑，这些通用指令(例如，x86指令)调用对加速器群集801的命令。然而，本发明的基本原理不限于任何特定类型的通用群集或加速器群集。

[0225] 图15中所示的实施例也包括用于将存储器模块1525耦合至计算机系统的存储器接口1520。在一个实施例中，存储器模块1525是诸如随机存取存储器 (RAM)之类的双列直插存储器模块(DIMM)，并且存储器接口可生成访问该存储器模块1525所需的电信令(例如、列地址选通(CAS)、行地址选通 (RAS)、允许写入(WE)和允许输出(OE)信号)。

[0226] 在一个实施例中，存储器接口1520包括用于与各种类型的存储器模块对接的逻辑和电路，各种类型的存储器模块包括：易失性存储器模块，例如，RAM；和非易失性存储器模块，例如，相变存储器(有时也被称为相变随机存取存储器(PRAM或PCRAM)、PCME、双向统一存储器或硫族RAM(C-RAM))。例如，计算机系统1500的一个实施例实现两层级(2L)的存储器层次结构，其包括可以是诸如RAM之类的易失性存储器的“近存储器”部分和可以被实现为相变存储器(PCM)的“远存储器”部分。在这种情况下，存储器接口可包括访问这两种存储器

类型所需的逻辑和电路。

[0227] 所示出的实施例1500也包括用于与诸如硬驱动器或其他非易失性存储设备对接的一个或多个存储接口1518。在一个实施例中，存储接口1518包括串行ATA存储接口，而硬驱动器包括固态驱动器(SSD)或磁存储设备。在使用 2LM存储器(如上文中所讨论)的本发明的实施例中，存储设备1519上的存储设备的部分可用于“远存储器”(或“远存储器”的部分)。

[0228] 所示出的实施例1500也包括用于与一个或多个图形处理单元1503对接的图形接口1502。GPU可被嵌入在计算机系统的主板上，或位于被插入在该主板中的单独的卡上(例如，经由PCI快速图形接口或其他高速图形接口)。诸如数字视频接口(DVI)、高清多媒体接口(HDMI)或显示端口视频输出接口之类的视频输出接口1504将视频流输出到为终端用户渲染视频的监视器1505。如所指出的那样，可使用本文中所描述的实施例中的任何实施例，将 GPU实现为用于执行图形程序代码的加速器组件。

[0229] 所示出的实施例1500也包括用于接收多个数字和模拟音频输入的音频数据接口1516。例如，话筒可被耦合至音频输入接口中的一个接口以捕捉用户的语音(例如，在网上聊天、打电话或记录音频期间)。此外，数字语音输入可被用作诸如Toslink接口。

[0230] 所示出的实施例1500也包括用于从各种不同的系统传感器1509中收集数据的传感器中枢1515。作为示例而非限制，传感器1509可包括用于检测计算机系统1500的位置和定向的机械传感器、运动传感器和位置传感器。例如，在一个实施例中，传感器可包括用于检测沿X、Y和Z轴的加速度值并将该数据报告给传感器中枢的多轴加速度计。然后，传感器中枢可执行计算一确定计算机系统1500的当前定向。例如，如果计算机系统是笔记本计算机，则传感器中枢可检测该计算机监视器的当前位置。传感器1509也可以是用于检测距离参考位置的位移的惯性传感器和/或用于检测与用户或其他设备的接近度的接近度传感器。在一个实施例中，传感器1509包括全球定位系统(GPS)传感器或用于确定计算机系统的当前全球位置的其他传感器。传感器1509也可包括用于检测地球电场取向(即，用于确定计算系统相对于北的当前位置)的磁力计。传感器1509也可包括用于检测取向变化的陀螺仪和用于检测当前照明条件的环境光传感器(例如，从而使得传感器中枢或其他系统组件可响应性地调节监视器1505的亮度)。

[0231] 从各种传感器1509中收集到的全部数据可用于确定操作的当前模式，并响应性地调节计算设备1500的操作。例如，响应于来自传感器1509的信号，计算设备可进入：第一操作模式，其中，本文中所描述的加速器调用被启用；以及第二操作模式，其中，本文中所描述的加速器调用被禁用。

[0232] 所示出的实施例1500也包括用于耦合至用于捕捉运动视频和静止图片的视频相机的相机接口1514。例如，在一个实施例中，相机接口1514收集用于视频会议应用(可在其中使用本文中所描述的加速器调用技术)的运动视频。例如，一个加速器可配置成用于高效地将视频流编码为H.264/MPEG-4AVC格式。然而，应当注意，本发明的基本原理不限于任何特定的视频压缩格式。

[0233] 所示出的实施例1500也包括用于建立与所连接设备之间(例如，移动电话、平板、打印机、外部相机、MIDI设备等)的串行数据通信的串行总线接口1513。该实施例进一步包括用于在以太网网络上建立网络连接的以太网接口 1512和用于使用蜂窝通信协议在蜂窝

网络上建立语音和数据连接的蜂窝接口 1511。可采用各种蜂窝技术，包括但不限于：第三代合作伙伴计划(例如，3GPP2) 码分多址技术(例如，使用1xRTT/EVDO/eHRPD的CDMA2000技术)；长期演进(LTE)技术和/或LTE-高级型(LTE-A)技术；以及诸如WCDMA/TDSCDMA 之类的通用移动通信系统(UMTS)技术。此外，所示出的实施例也包括用于分别在WiFi信道(例如，802.11信道)和/或蓝牙信道上建立通信的WiFi和/ 或蓝牙接口1510。以太网通信接口、蜂窝通信接口和WiFi通信接口中的每一个包括收发机和用于使用合适的技术生成模拟传输信号的其他电路。在一个实施例中，也可调用加速器以支持网络通信过程(例如，用于执行诸如数据编码之类的网络基带功能)。

[0234] 所示出的实施例1500也包括功率管理接口1517，其用于检测计算机系统内的当前条件(例如，热、功率使用、电池寿命等)，并响应性地调节用于不同系统组件中的每一个的功率。例如，在某些条件下，功率管理接口1517可关闭本文中所描述的加速器功能，以便节省功率(例如，当电量降到低于阈值时)。

[0235] 所示出的实施例1500也包括功率管理接口1517，该功率管理接口1517 也可包括用于接收用户输入的各种类型的输入/输出设备，例如，光标控制装置 (例如，鼠标、触屏、触板等)、键盘灯。

[0236] 将会理解，在本发明的某些实施例中，没有在图15中示出的附加组件也可以是数据处理系统1500的部分，而在本发明的某些实施例中，可使用比图 15中所示的更少的组件。另外，应当理解，没有在图15中示出的一个或多个总线和/或互连可用于如本领域中公知的那样互连各种组件。

[0237] 本发明的多个实施例可包括上述各个步骤。可在可被用于使通用或专用处理器执行这些步骤的机器可执行指令中具体化这些步骤。或者，可由包含用于执行这些步骤的硬连线逻辑的专用硬件组件，或可由被编程的计算机组件和自定义硬件组件的任何组合来执行这些步骤。

[0238] 如本文中所述，指令可以指硬件的具体配置，例如，配置成用于执行某些操作或具有预定功能的专用集成电路(ASIC)，或者被存储在被具体化在非瞬态计算机可读介质中的存储器中的软件指令。因此，可使用被存储在一个或多个电子设备(例如，终端站、网络元件等)上并在其上被执行的代码和数据来执行附图中所示的技术。此类电子设备使用诸如非瞬态计算机机器可读存储介质(例如，磁盘；光盘；随机存取存储器；只读存储器；闪存设备；相变存储器)之类的计算机机器可读介质和瞬态计算机机器可读通信介质(例如，电、光、声或其他形式的传播信号——诸如载波、红外信号、数字信号等)来(内部地和/或在网络上与其他电子设备之间进行)存储和传递代码和数据。另外，此类电子设备一般包括耦合至一个或多个其他组件的一个或多个处理器的集合，所述一个或多个其他组件例如是一个或多个存储设备(非瞬态机器可读存储介质)、用户输入/输出设备(例如，键盘、触摸屏和/或显示器)以及网络连接。该组处理器和其他组件的耦合一般是通过一个或多个总线和桥(也称为总线控制器)实现的。存储设备和携带网络话务的信号分别表示一个或多个机器可读存储介质和机器可读通信介质。因此，给定电子设备的存储设备通常存储用于在该电子设备的一个或多个处理器的集合上执行的代码和/或数据。当然，本发明的实施例的一个或多个部分可使用软件、固件和/或硬件的不同组合来实现。贯穿此具体实施方式，为了进行解释，陈述了众多具体细节以提供对本发明的透彻理解。然而，对本领域技术人员显而易见的是，没

有这些具体细节中的一些细节也可实施本发明。在某些实例中，并不详细描述公知的结构和功能，以免使本发明的主题模糊。因此，本发明的范围和精神应根据所附权利要求书来判断。


100 流水线
取出 102
长度解码 104
解码 106
分配 108
重命名 110
调度 112
寄存器读取/存储器读取 114
执行级 116
写回/存储器写入 118
异常处理 122
提交 124

图1A

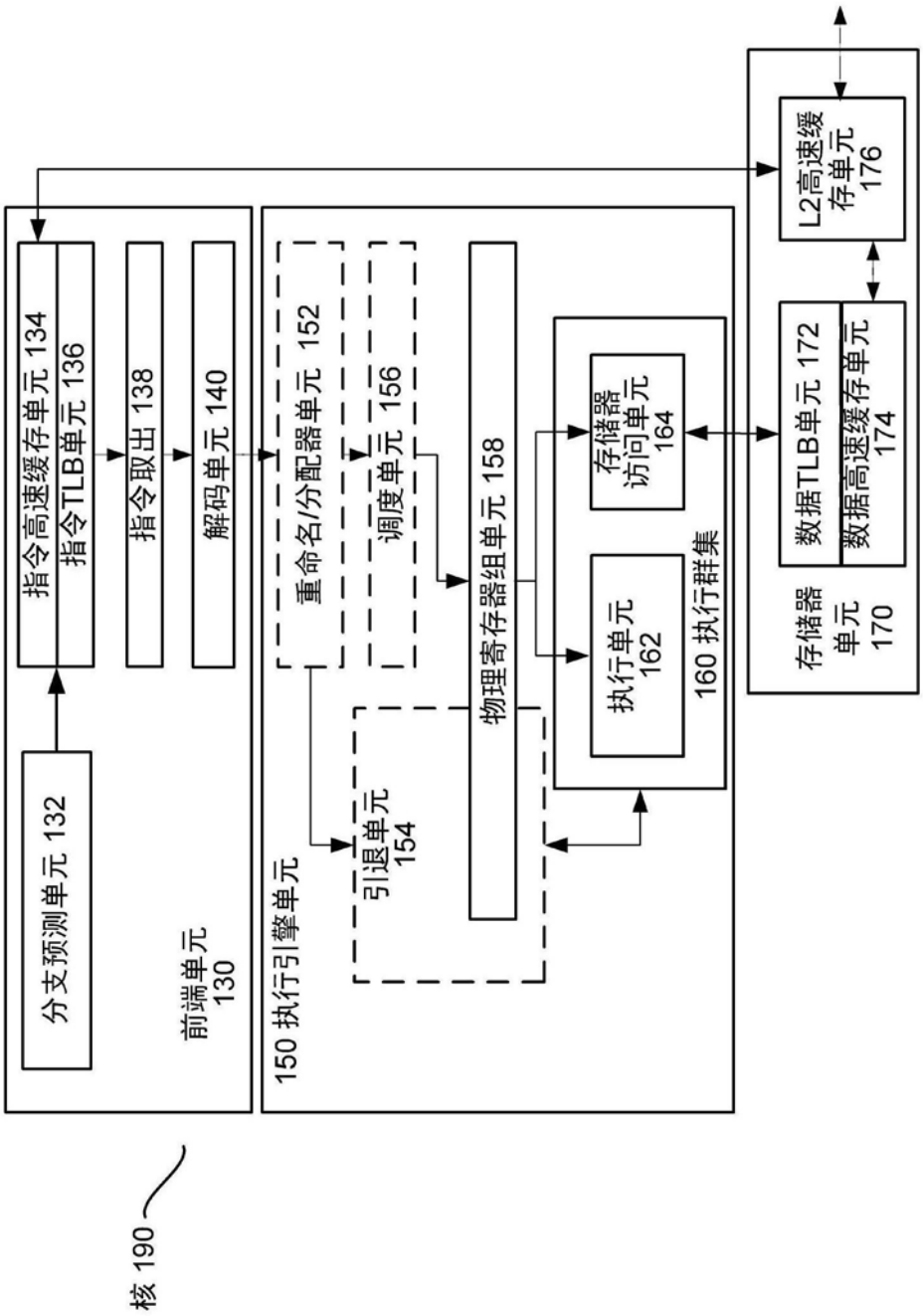
核 190
前端单元 130
分支预测单元 132
指令高速缓存单元 134
指令TLB单元 136
指令取出 138
解码单元 140
150 执行引擎单元
重命名/分配器单元 152
引退单元 154
调度单元 156
物理寄存器组单元 158
160 执行群集
执行单元 162
存储器访问单元 164
存储器单元 170
数据TLB单元 172
数据高速缓存单元 174
L2高速缓存单元 176

图1B

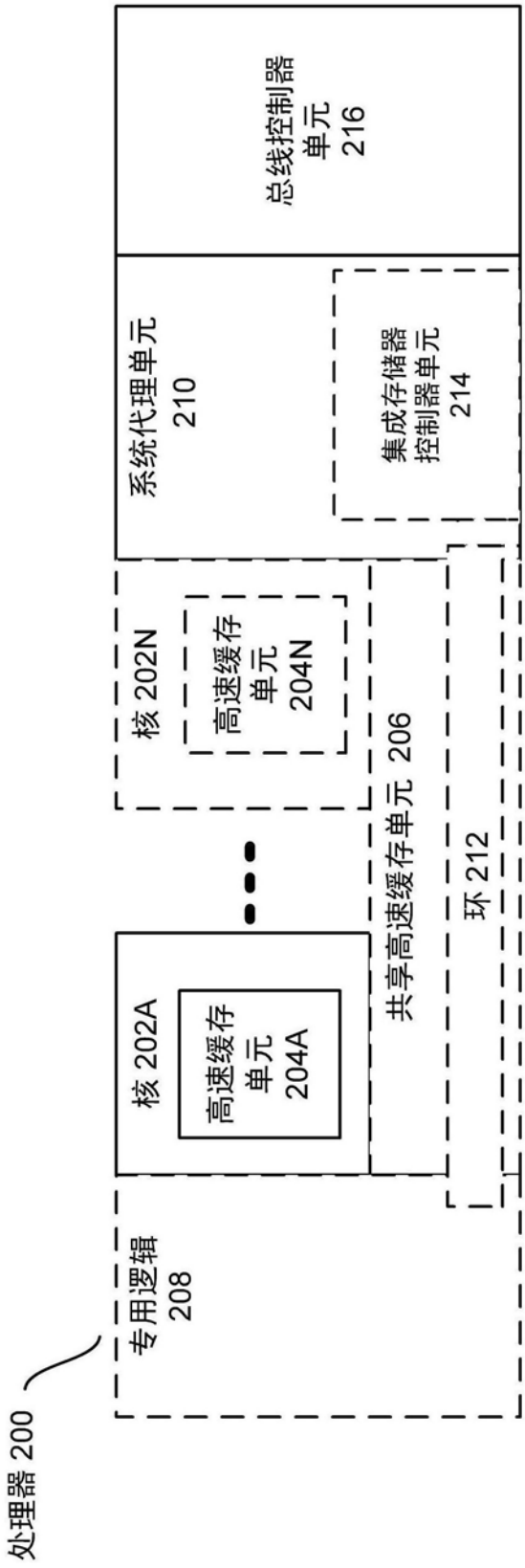
处理器 200
专用逻辑 208
核 202A
高速缓存单元 204A
核 202N
高速缓存单元 204N
共享高速缓存单元 206
环 212
系统代理单元 210
集成存储器控制器单元 214
总线控制器单元 216

图2

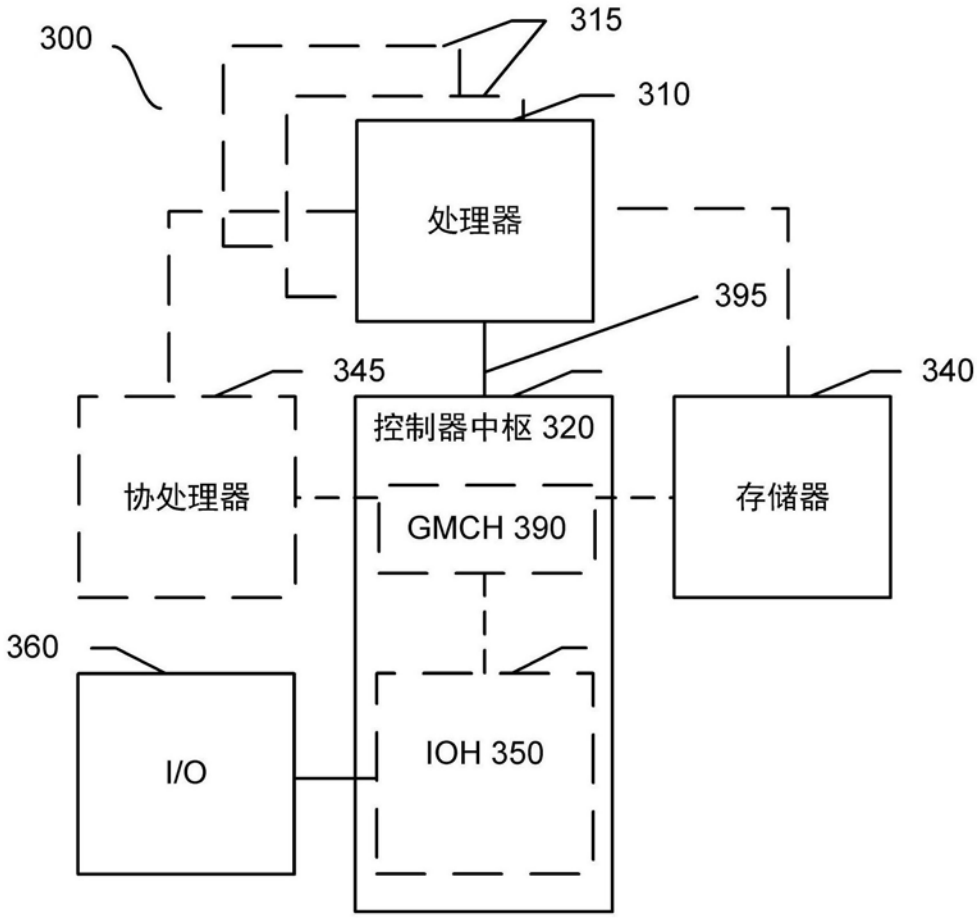
300
315
310
处理器
395
345
340
控制器中枢 320
协处理器
GMCH 390
存储器
360
I/O
IOH 350

图3

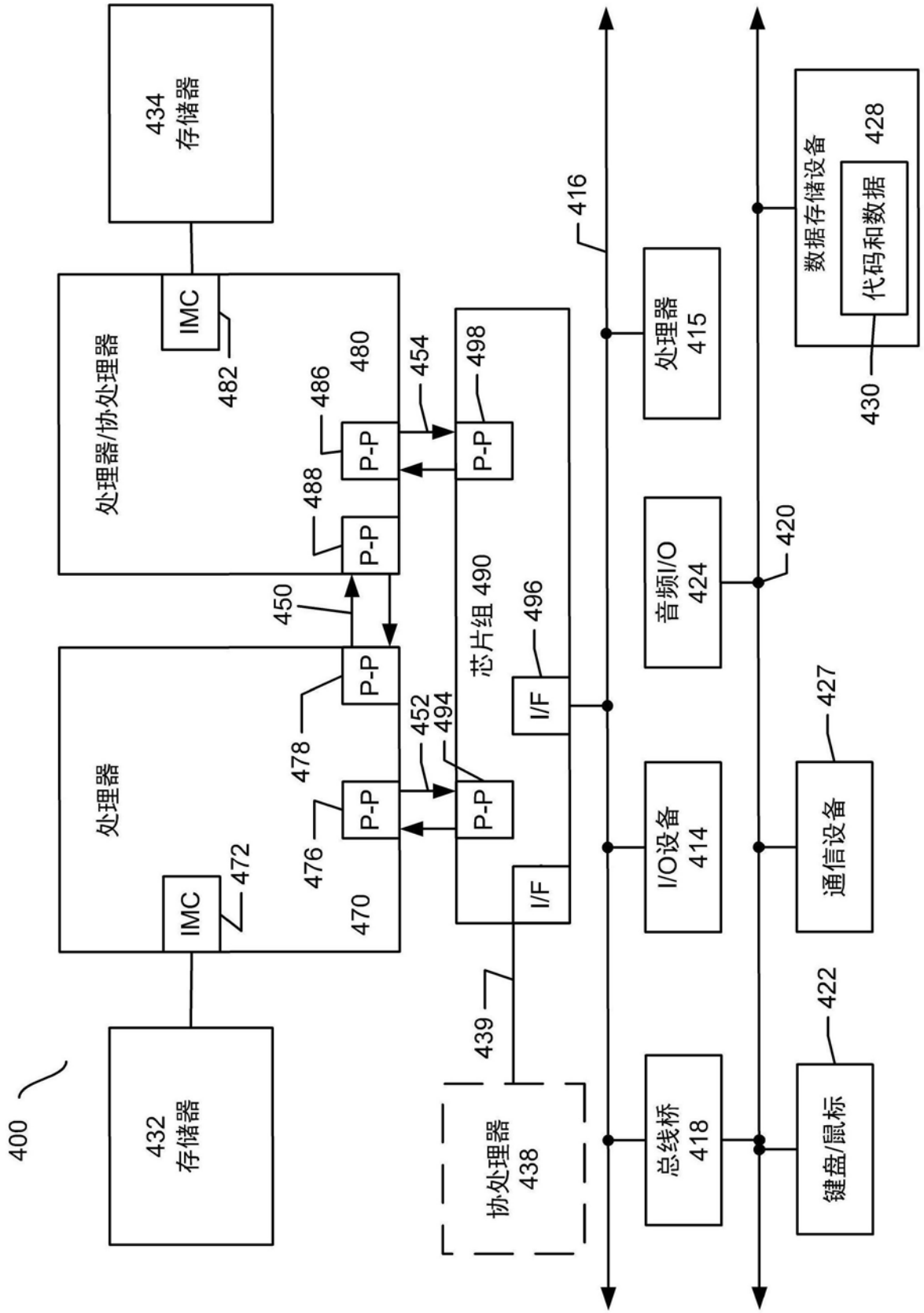
400
434
存储器
IMC
482
处理器/协处理器
480
486
P-P
488
P-P
454
498
P-P
450
处理器
IMC
472
470
478
P-P
476
P-P
452
494
芯片组 490
496
I/F
I/F
439
协处理器
438
432
存储器
416
处理器
415
音频I/O
424
420
I/O设备
414
427
通信设备
总线桥
418
422
键盘/鼠标
数据存储设备
428
代码和数据
430

图4

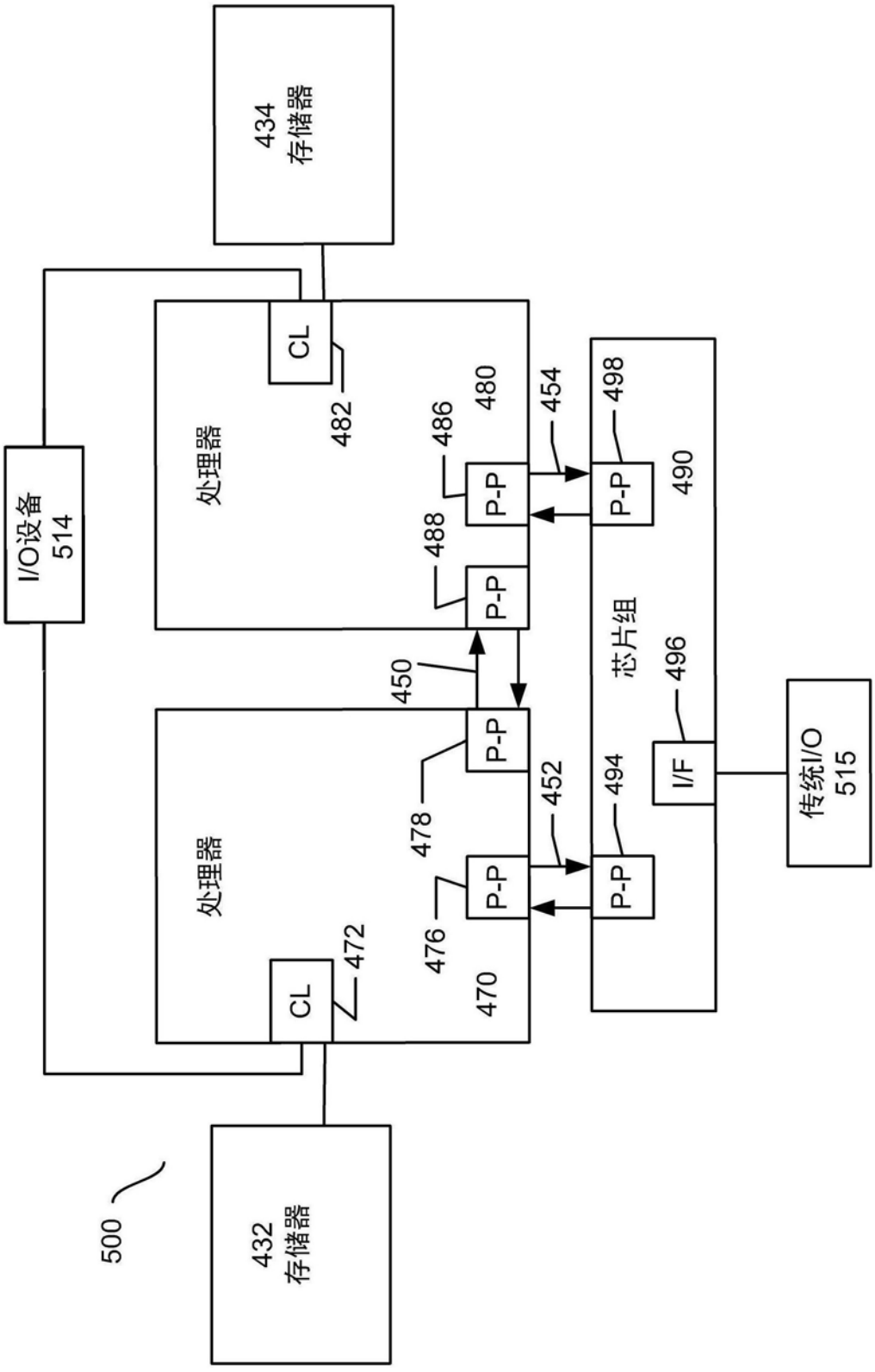
434
存储器
CL
482
处理器
480
486
P-P
488
P-P
454
498
P-P
490
芯片组
I/O设备
514
450
处理器
478
P-P
452
494
P-P
476
P-P
472
CL
470
496
I/F
传统I/O
515
500
432
存储器

图5

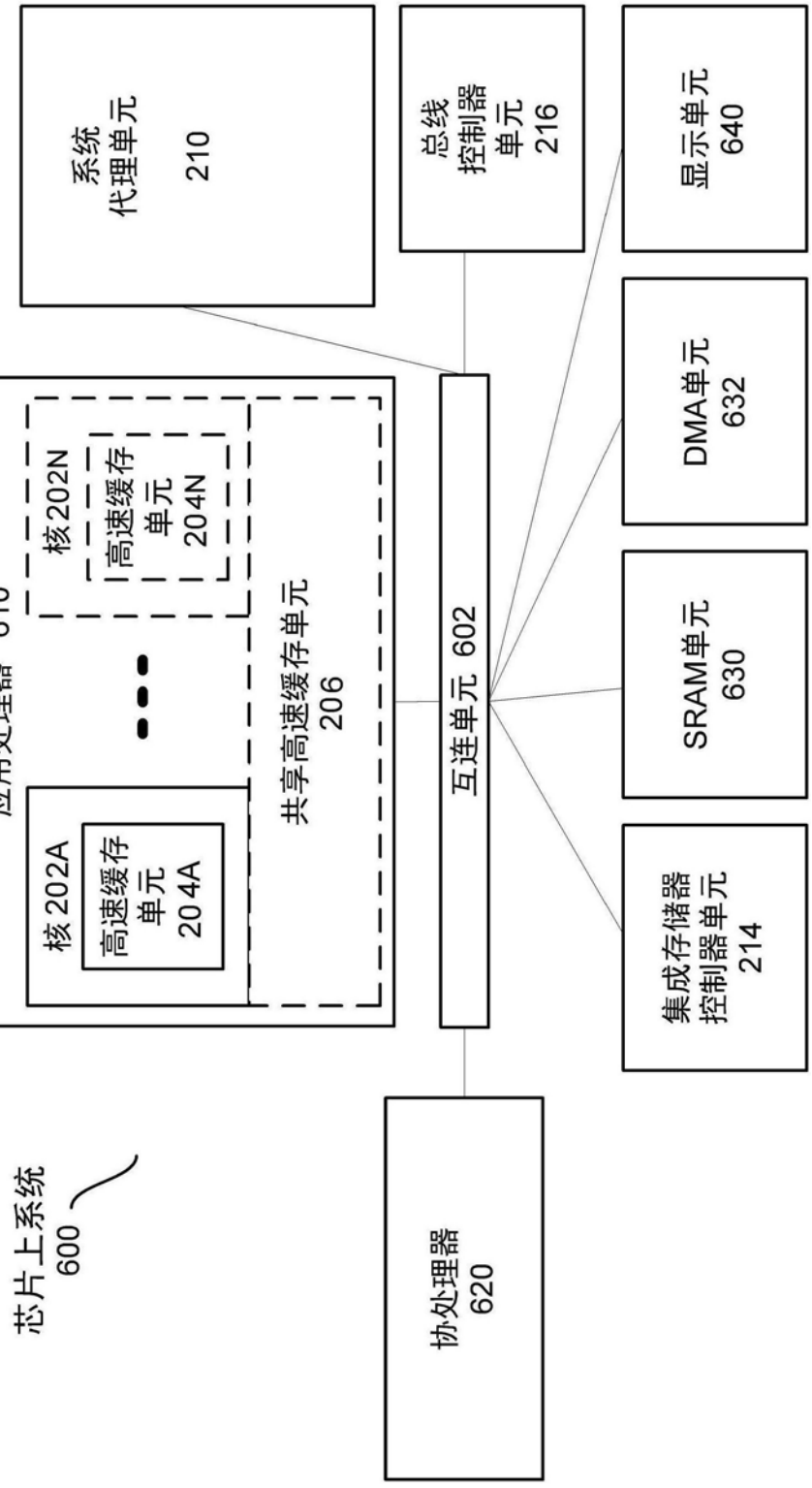
系统
代理单元
210
总线
控制器
单元
216
显示单元
640
应用处理器 610
核 202N
高速缓存
单元
204N
核 202A
高速缓存
单元
204A
共享高速缓存单元
206
互连单元 602
DMA单元
632
SRAM单元
630
集成存储器
控制器单元
214
芯片上系统
600
协处理器
620

图6

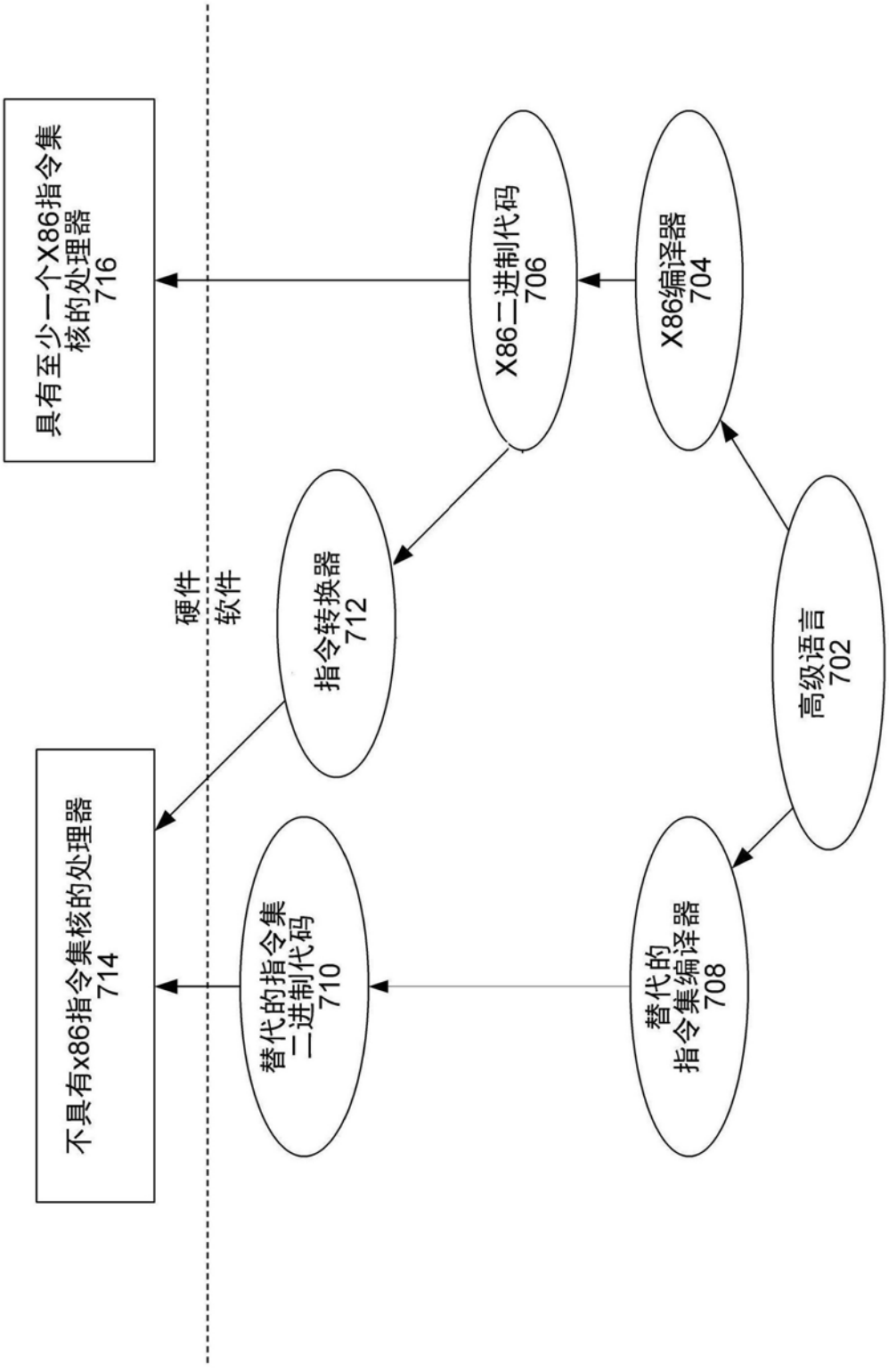
具有至少一个X86指令集核的处理器
716
X86二进制代码
706
X86编译器
704
硬件
软件
指令转换器
712
高级语言
702
不具有x86指令集核的处理器
714
替代的指令集二进制代码
710
替代的指令集编译器
708

图7

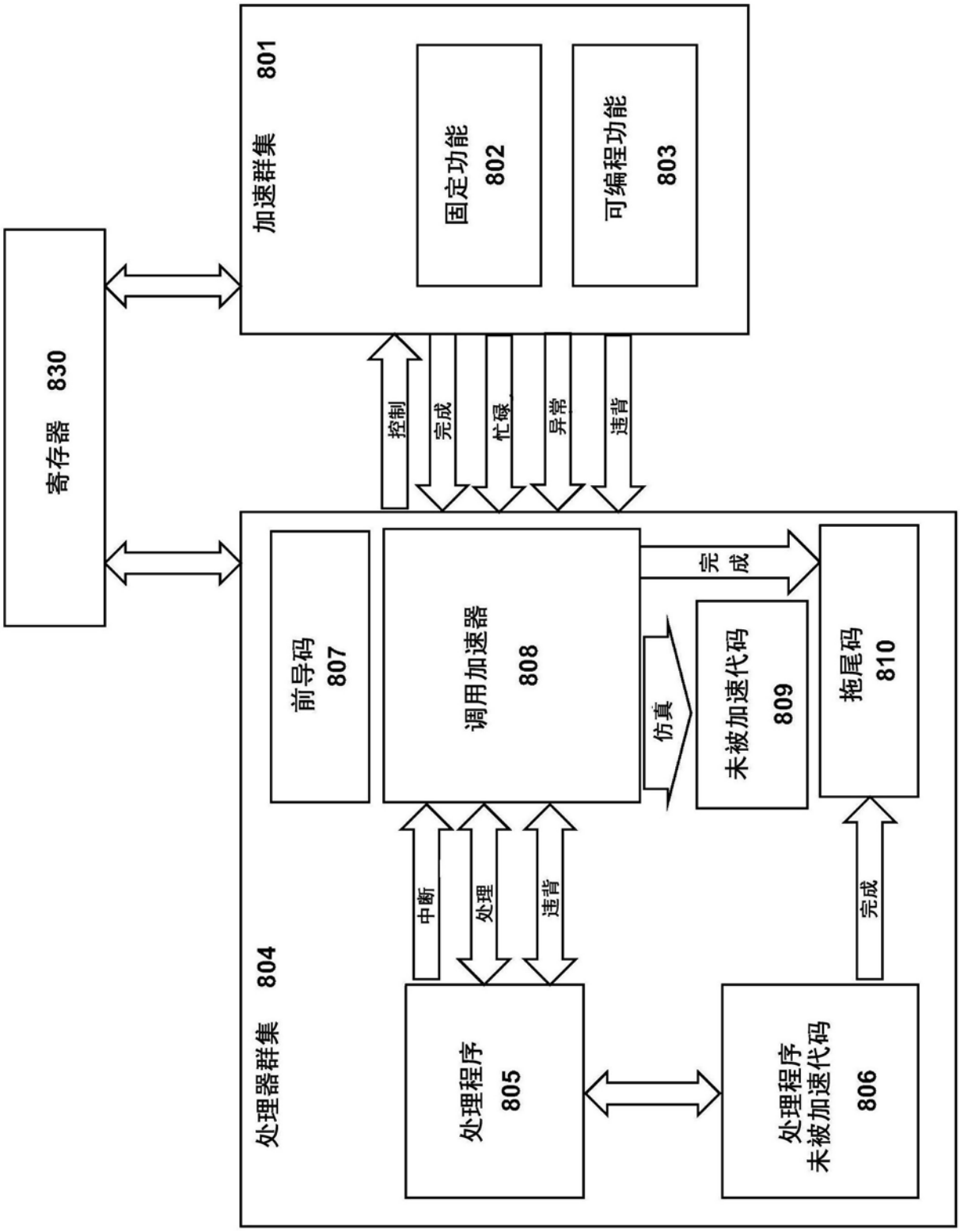
寄存器 830
加速群集 801
固定功能 802
可编程功能 803
控制
完成
忙碌
异常
违背
处理器群集 804
前导码 807
调用加速器 808
完成
仿真
未被加速代码 809
拖尾码 810
中断
处理
违背
完成
处理程序 805
处理程序 未被加速代码 806

图8A

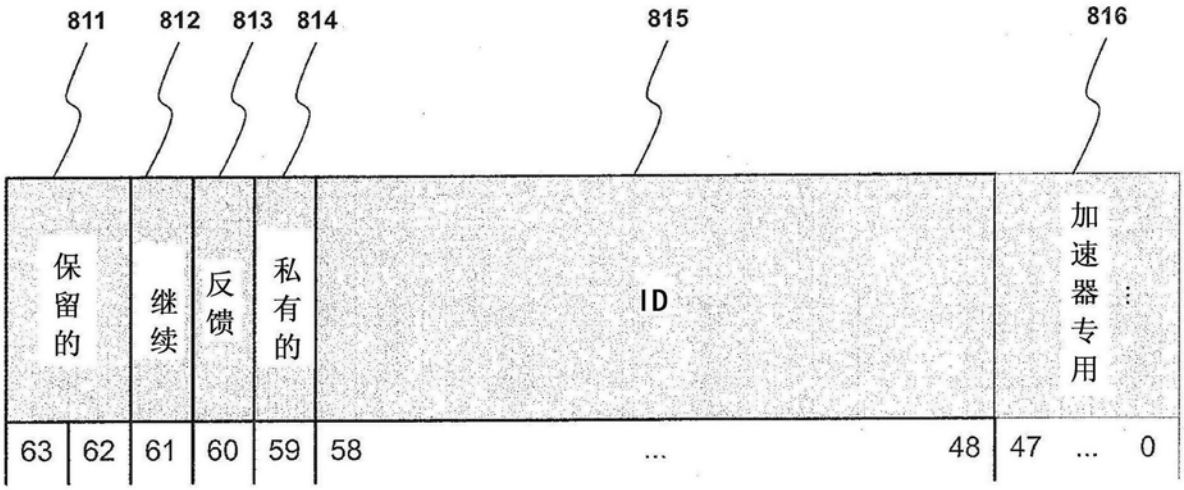
811
812
813
814
815
816
保留的
继续
反馈
私有的
ID
加速器专用
63
62
61
60
59
58
...
48
47
...
0

图8B

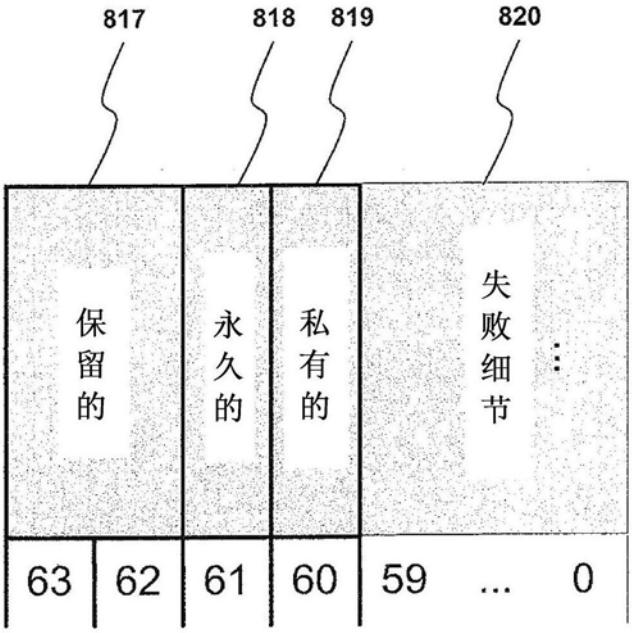
817
818
819
820
保留的
永久的
私有的
失败细节
63
62
61
60
59
...
0

图8C

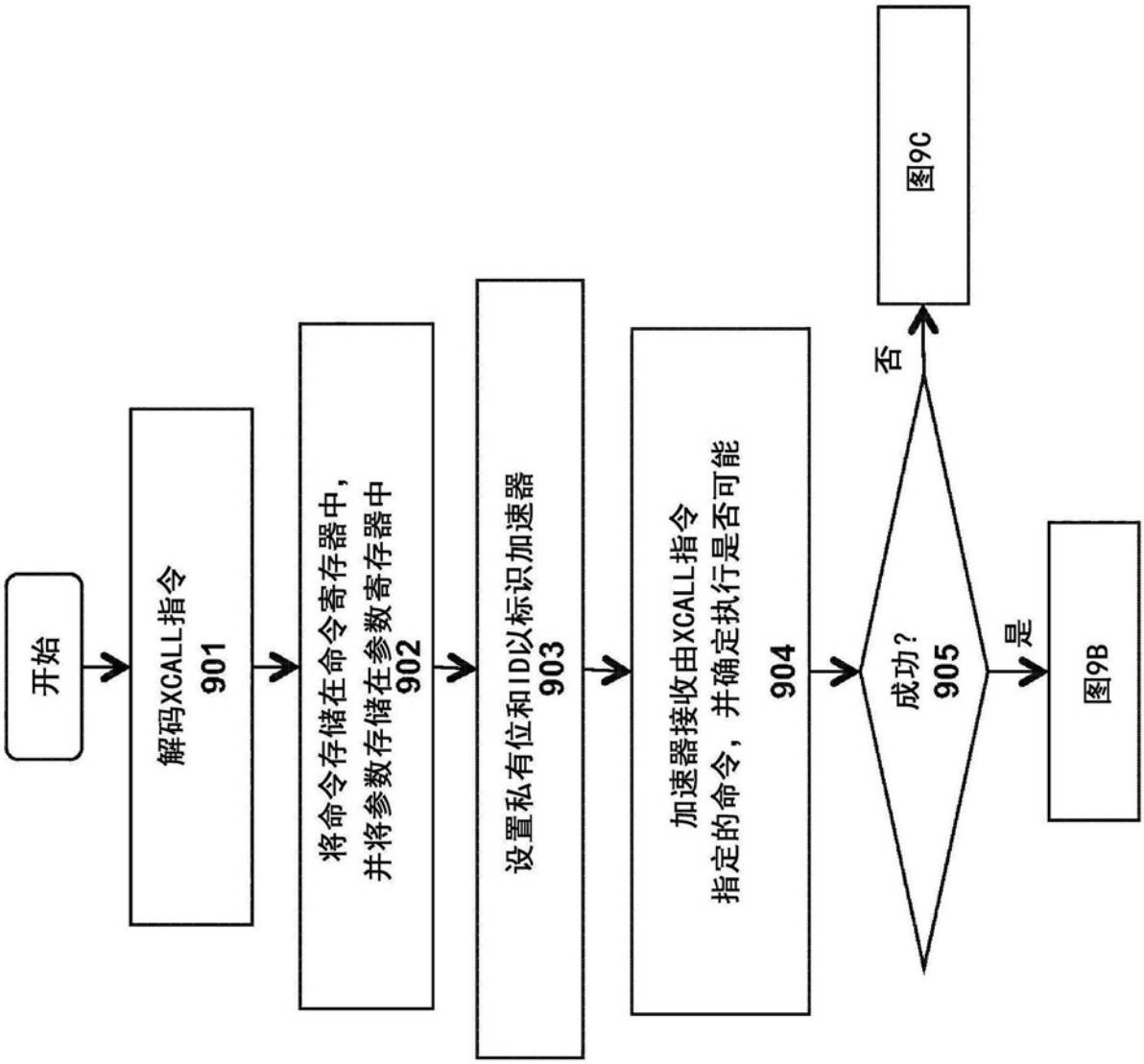
开始
解码XCALL指令
901
将命令存储在命令寄存器中，
并将参数存储在参数寄存器中
902
设置私有位和ID以标识加速器
903
加速器接收由XCALL指令
指定的命令，并确定执行是否可能
904
成功？
905
否
图9C
是
图9B

图9A

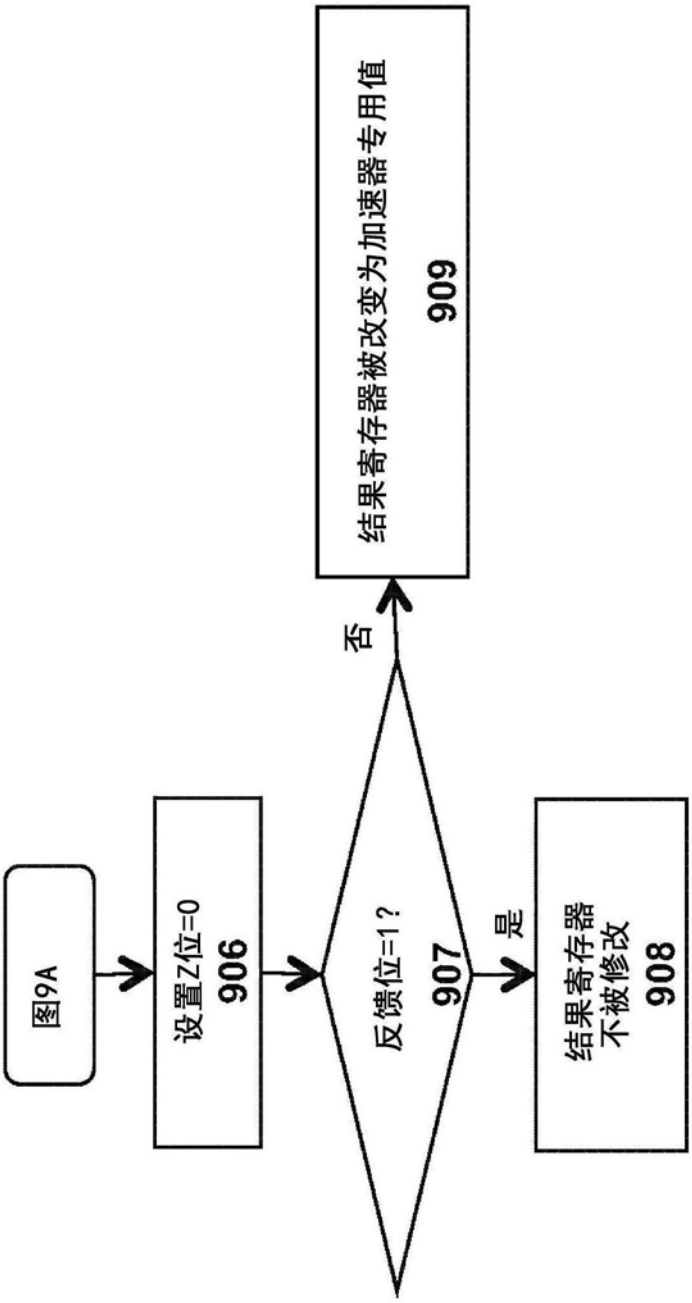
图9A
设置Z位=0
906
反馈位=1？
907
否
是
结果寄存器被改变为加速器专用值
909
结果寄存器
不被修改
908

图9B

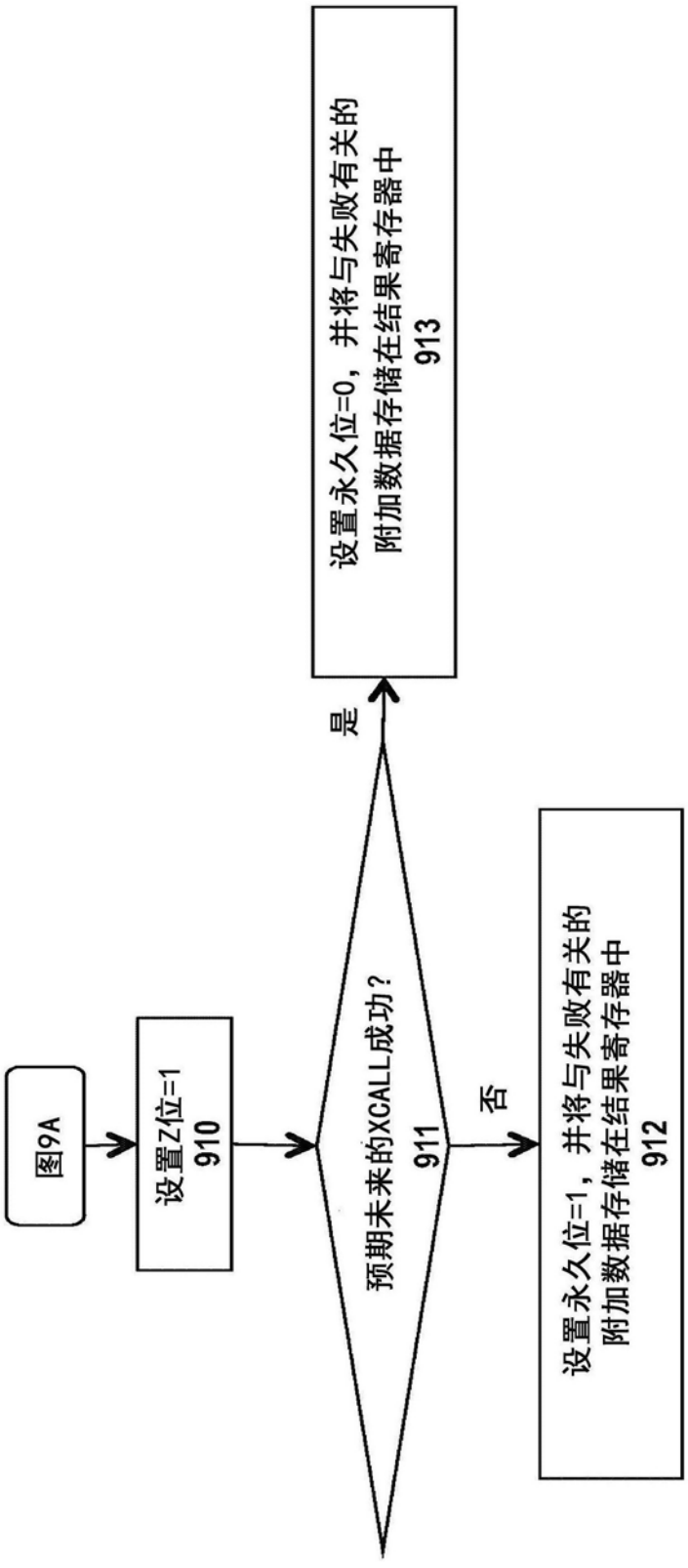
图9A
设置Z位=1
910
预期未来的XCALL成功？
911
是
设置永久位=0，并将与失败有关的
附加数据存储在结果寄存器中
913
否
设置永久位=1，并将与失败有关的
附加数据存储在结果寄存器中
912

图9C

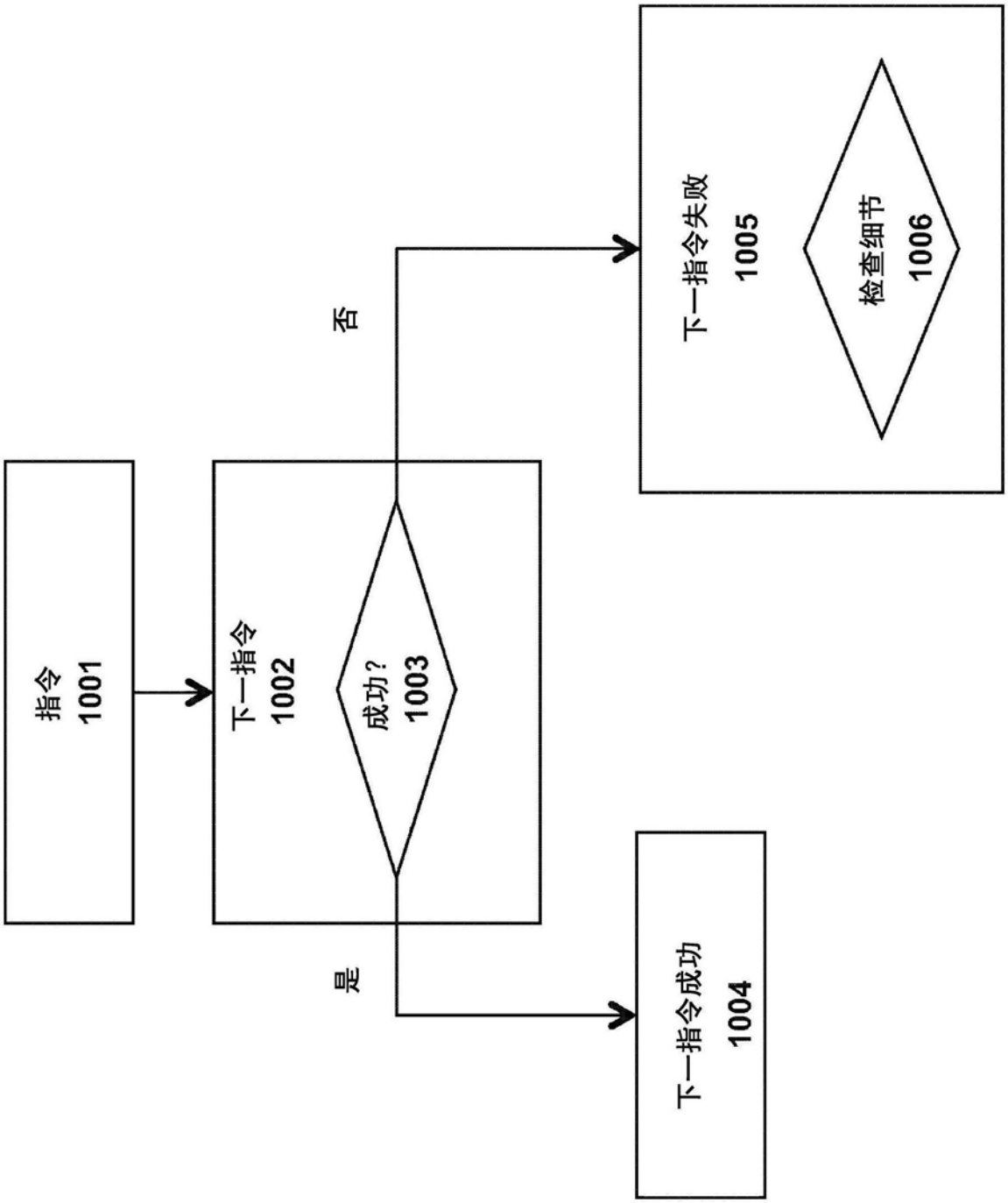
指令
1001
下一指令
1002
成功?
1003
否
是
下一指令失败
1005
检查细节
1006
下一指令成功
1004

图10

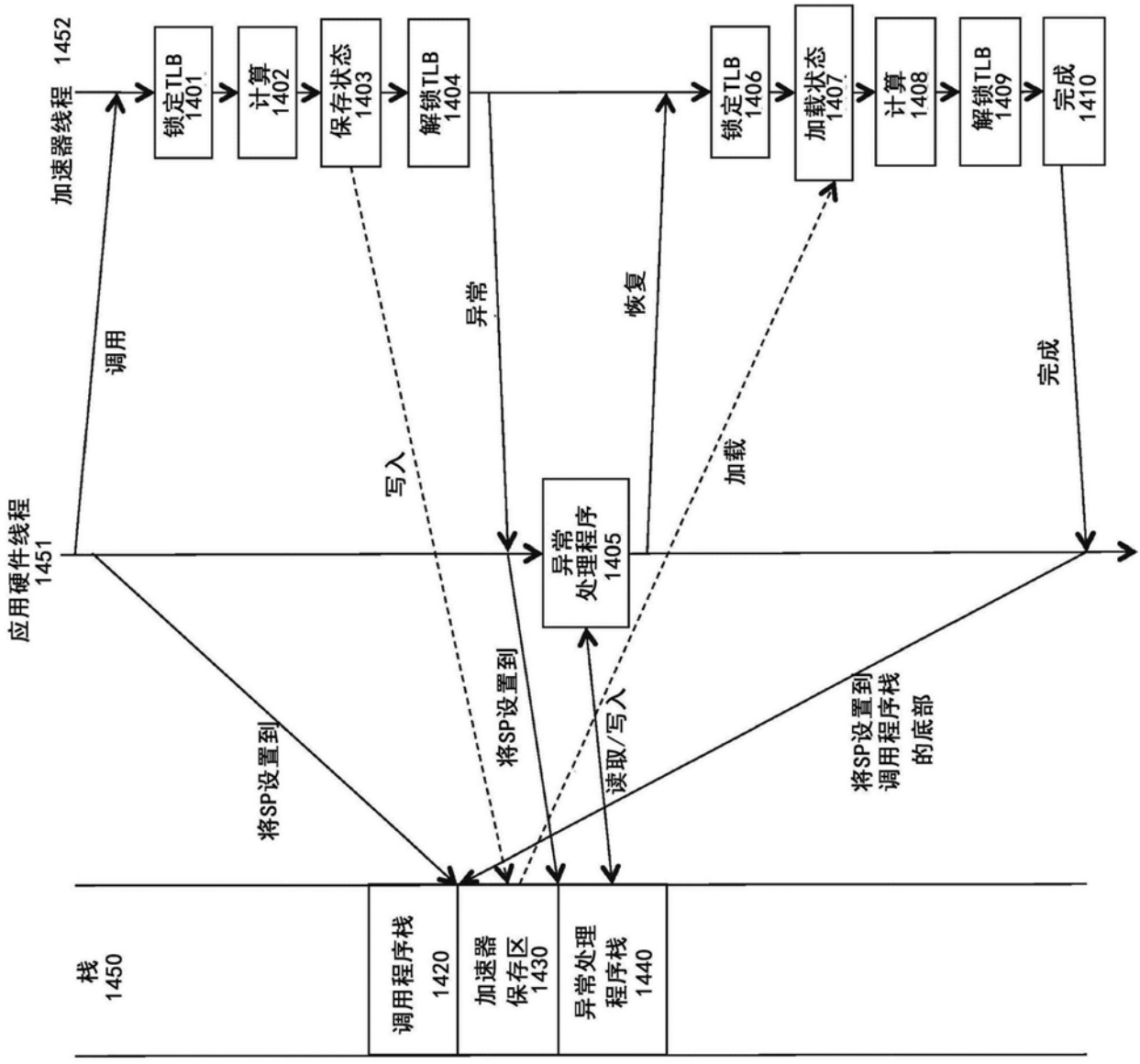
加速器线程 1452
应用硬件线程 1451
锁定TLB 1401
计算 1402
保存状态 1403
解锁TLB 1404
锁定TLB 1406
加载状态 1407
计算 1408
解锁TLB 1409
完成 1410
异常处理程序 1405
调用
异常
恢复
完成
写入
加载
将SP设置到
将SP设置到
读取/写入
将SP设置到调用程序栈的底部
栈 1450
调用程序栈 1420
加速器保存区 1430
异常处理程序栈 1440

图11

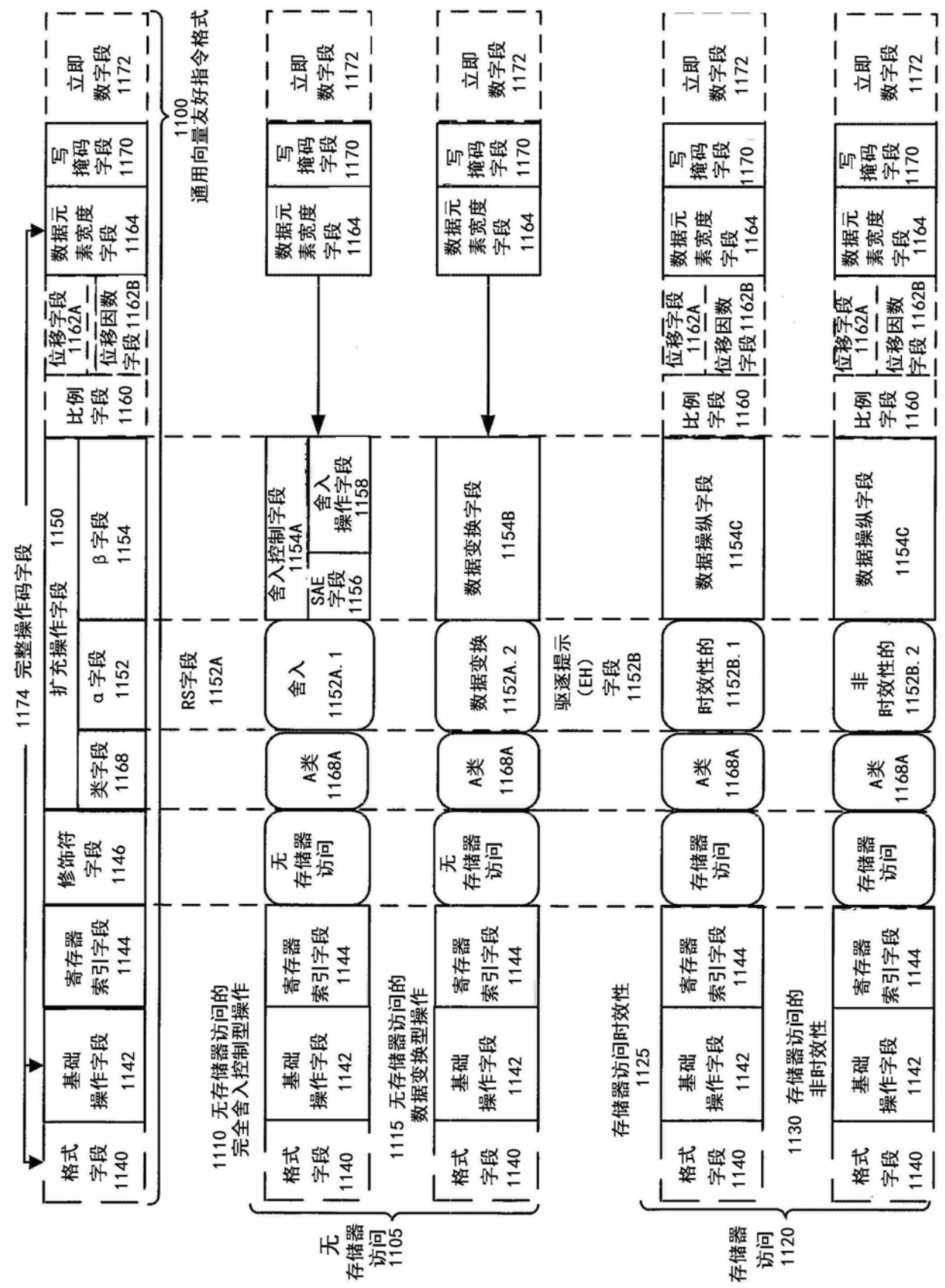
1174 完整操作码字段
1100 通用向量友好指令格式
格式字段 1140
基础操作字段 1142
寄存器索引字段 1144
修饰符字段 1146
扩充操作字段 1150
类字段 1168
α 字段 1152
β 字段 1154
比例字段 1160
位移字段 1162A
位移因数字段 1162B
数据元素宽度字段 1164
写掩码字段 1170
立即数字段 1172
RS字段 1152A
A类 1168A
无存储器访问
舍入 1152A.1
舍入控制字段 1154A
SAE 字段 1156
舍入操作字段 1158
数据变换 1152A.2
数据变换字段 1154B
驱逐提示 (EH) 字段 1152B
时效性的 1152B.1
非时效性的 1152B.2
数据操纵字段 1154C
1110 无存储器访问的完全舍入控制型操作
1115 无存储器访问的数据变换型操作
存储器访问时效性 1125
1130 存储器访问的非时效性
无存储器访问 1105
存储器访问 1120

图12A

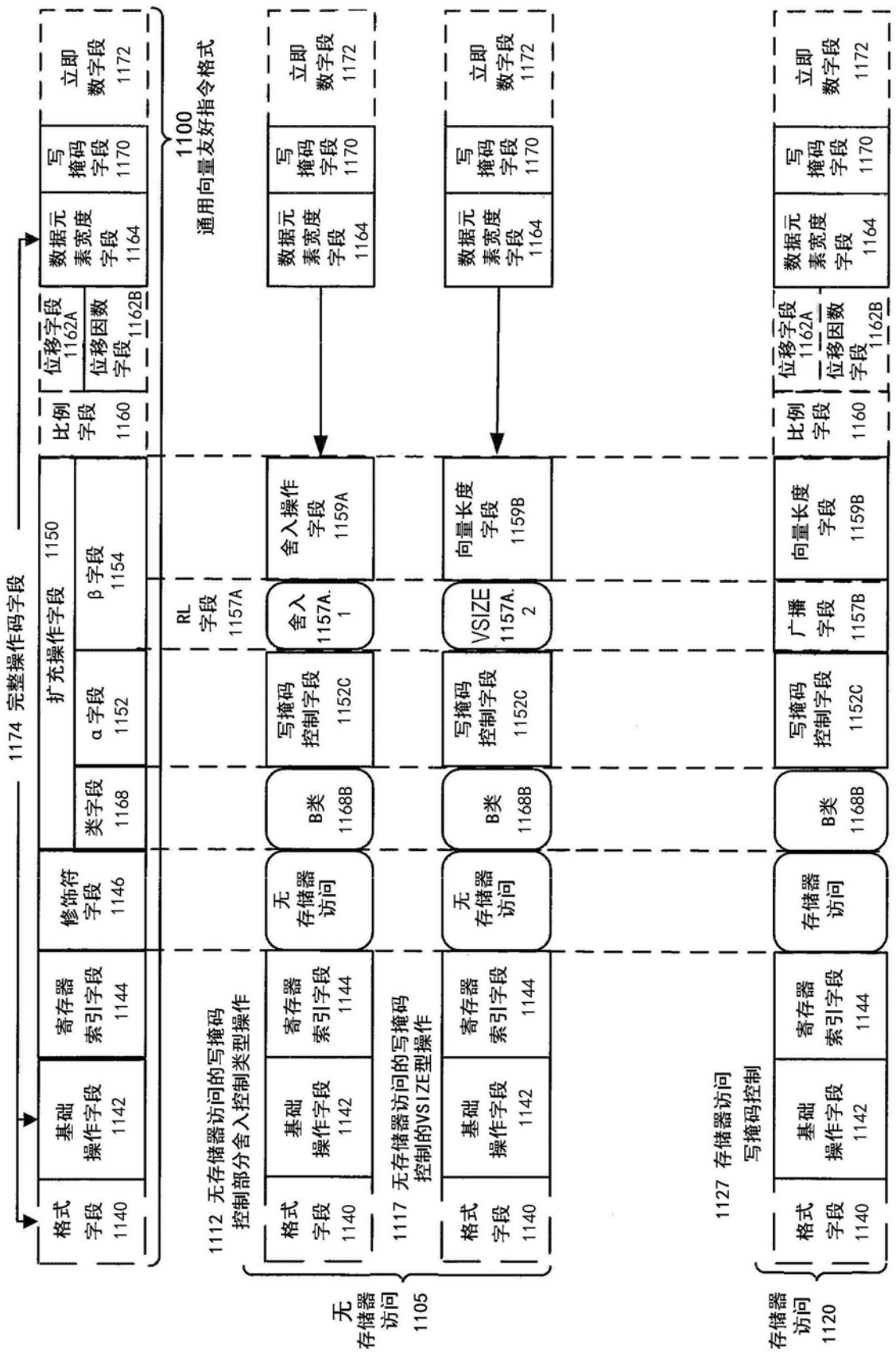
1100
通用向量友好指令格式
1174 完整操作码字段
格式字段 1140
基础操作字段 1142
寄存器索引字段 1144
修饰符字段 1146
类字段 1168
扩充操作字段 1150
α 字段 1152
β 字段 1154
比例字段 1160
位移字段 1162A
位移因数字段 1162B
数据元素宽度字段 1164
写掩码字段 1170
立即数字段 1172
无存储器访问 1105
1112 无存储器访问的写掩码控制部分舍入控制类型操作
无存储器访问
B类 1168B
写掩码控制字段 1152C
RL 字段 1157A
舍入 1157A.1
舍入操作字段 1159A
1117 无存储器访问的写掩码控制的VSIZE型操作
VSIZE 1157A.2
向量长度字段 1159B
1127 存储器访问写掩码控制
存储器访问
存储器访问 1120
广播字段 1157B

图12B

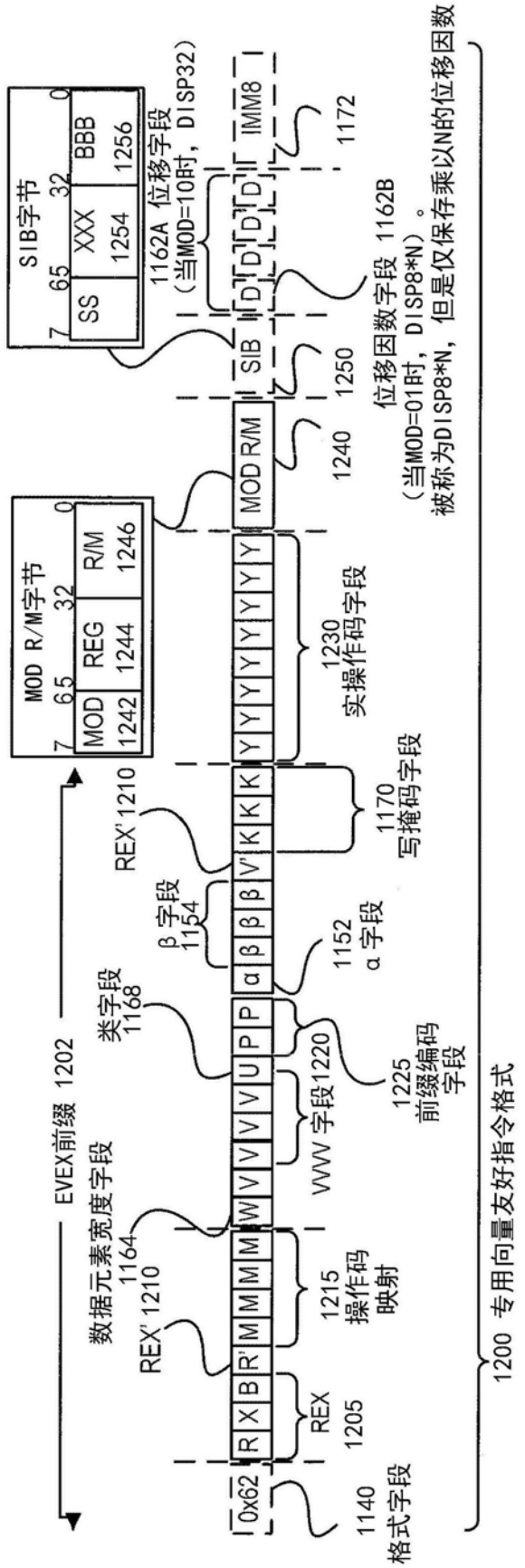
SIB字节
SS
XXX
BBB
1254
1256
1162A 位移字段
（当MOD=10时，DISP32）
MOD R/M字节
MOD
REG
R/M
1242
1244
1246
SIB
MOD R/M
1240
1250
位移因数字段 1162B
（当MOD=01时，DISP8*N）。
被称为DISP8*N，但是仅保存乘以N的位移因数
1172
EVEX前缀 1202
数据元素宽度字段
1164
类字段
1168
REX' 1210
β 字段
1154
1152
α 字段
1230
实操作码字段
1170
写掩码字段
1225
前缀编码
字段
VVVV 字段1220
REX' 1210
1215
操作码
映射
REX
1205
0x62
1140
格式字段
1200 专用向量友好指令格式

图13A

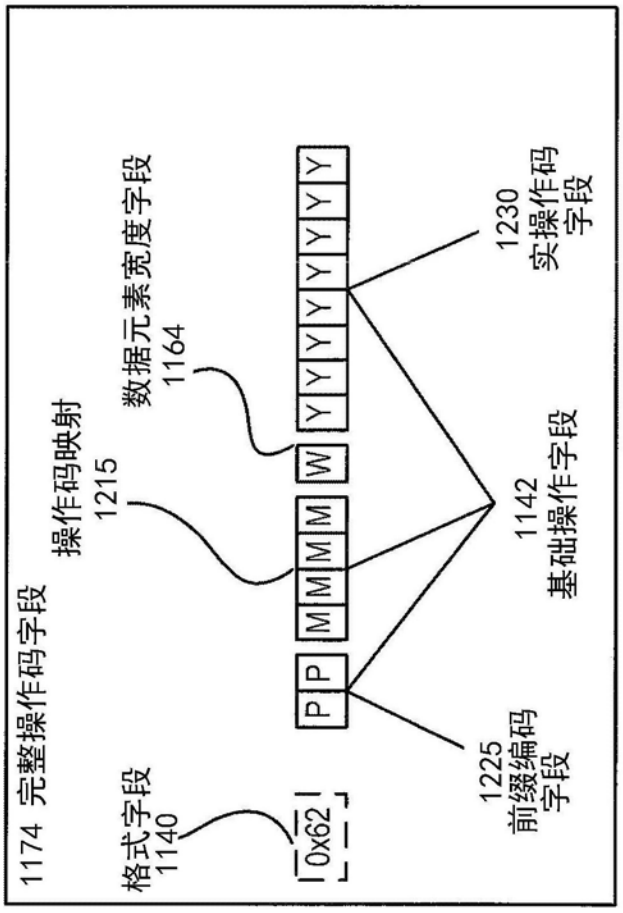
1174 完整操作码字段
操作码映射
1215
数据元素宽度字段
1164
格式字段
1140
0x62
P P M M M W Y Y Y Y Y Y Y
1230
实操作码
字段
1142
基础操作字段
1225
前缀编码
字段

图13B

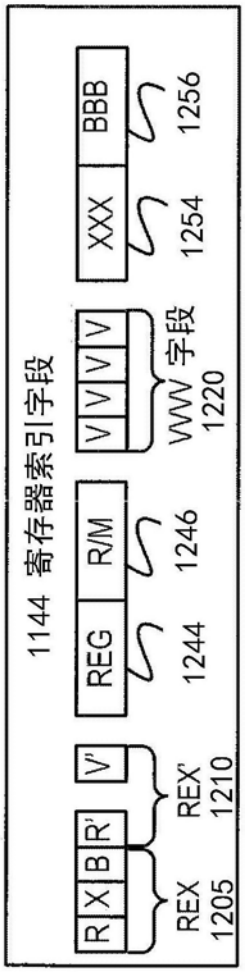
1144 寄存器索引字段
R
X
B
R'
V'
REG
R/M
V
V
V
V
XXX
BBB
REX
1205
REX'
1210
1244
1246
VVV 字段
1220
1254
1256

图13C

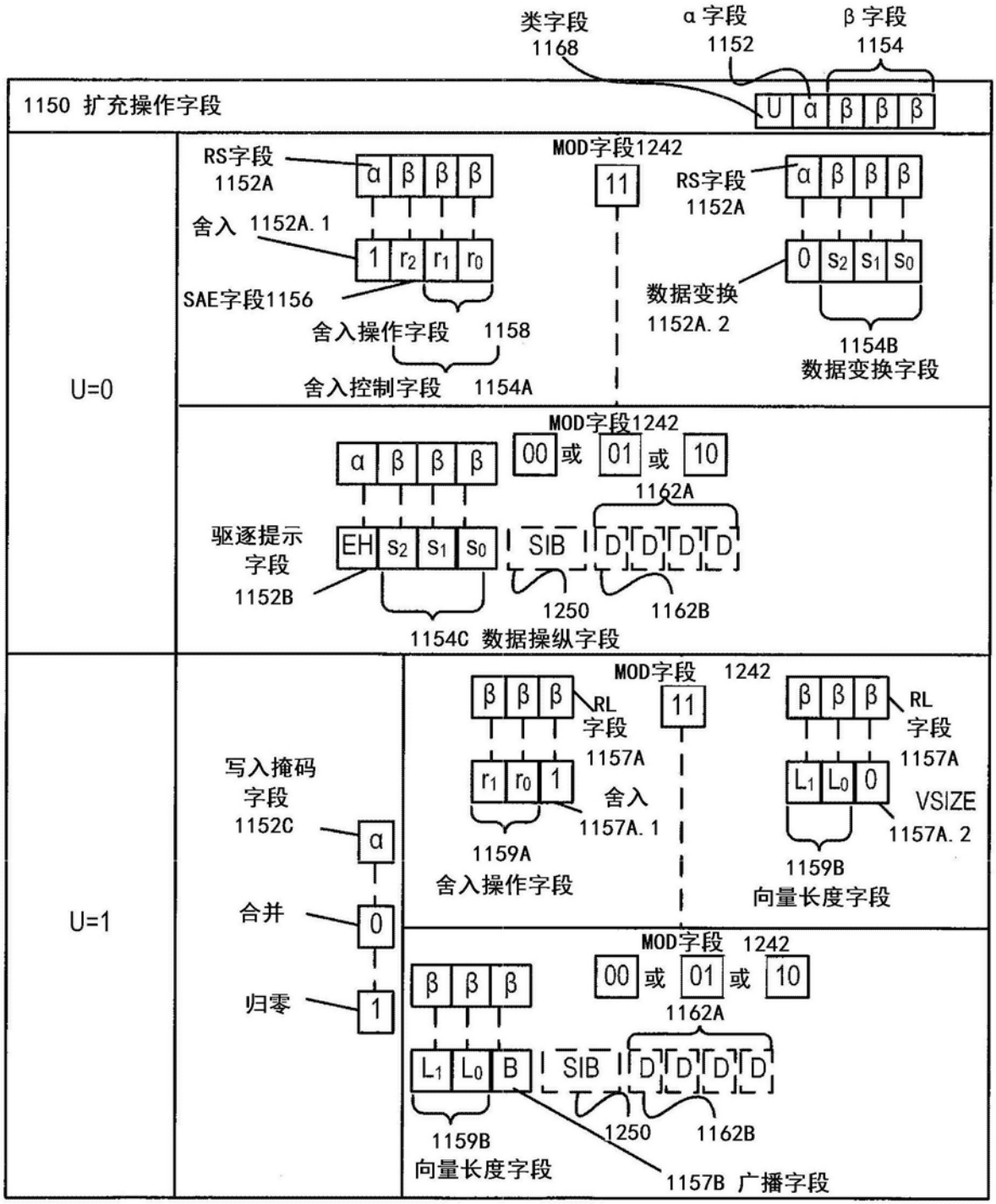
类字段
1168
α 字段
1152
β 字段
1154
1150 扩充操作字段
U α β β β
U=0
RS字段
1152A
α β β β
舍入 1152A. 1
1 r2 r1 r0
SAE字段1156
舍入操作字段 1158
舍入控制字段 1154A
MOD字段1242
11
RS字段
1152A
α β β β
0 S2 S1 S0
数据变换
1152A. 2
1154B
数据变换字段
MOD字段1242
α β β β
00 或 01 或 10
1162A
驱逐提示
字段
1152B
EH S2 S1 S0
SIB
D D D D
1250
1162B
1154C 数据操纵字段
U=1
写入掩码
字段
1152C
α
合并
0
归零
1
MOD字段 1242
β β β
RL
字段
1157A
11
r1 r0 1
舍入
1157A. 1
1159A
舍入操作字段
β β β
RL
字段
1157A
L1 L0 0
VSIZE
1157A. 2
1159B
向量长度字段
MOD字段 1242
β β β
00 或 01 或 10
1162A
L1 L0 B
SIB
D D D D
1250
1162B
1159B
向量长度字段
1157B 广播字段

图13D

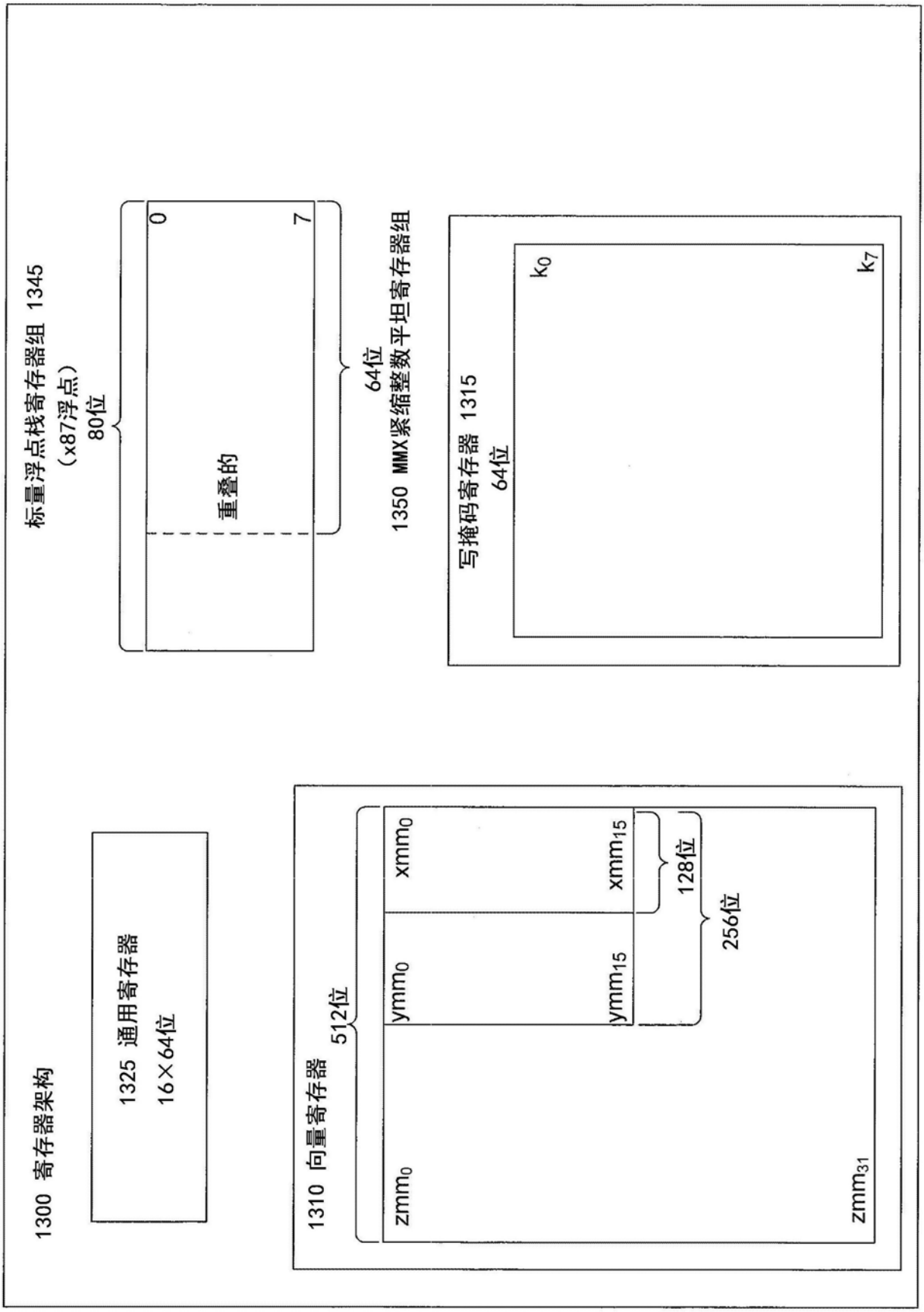
1300 寄存器架构
1325 通用寄存器
16×64位
1310 向量寄存器
512位
zmm0
zmm31
ymm0
ymm15
xmm0
xmm15
128位
256位
标量浮点栈寄存器组 1345
(x87浮点)
80位
0
7
重叠的
64位
1350 MMX紧缩整数平坦寄存器组
写掩码寄存器 1315
64位
k0
k7

图14

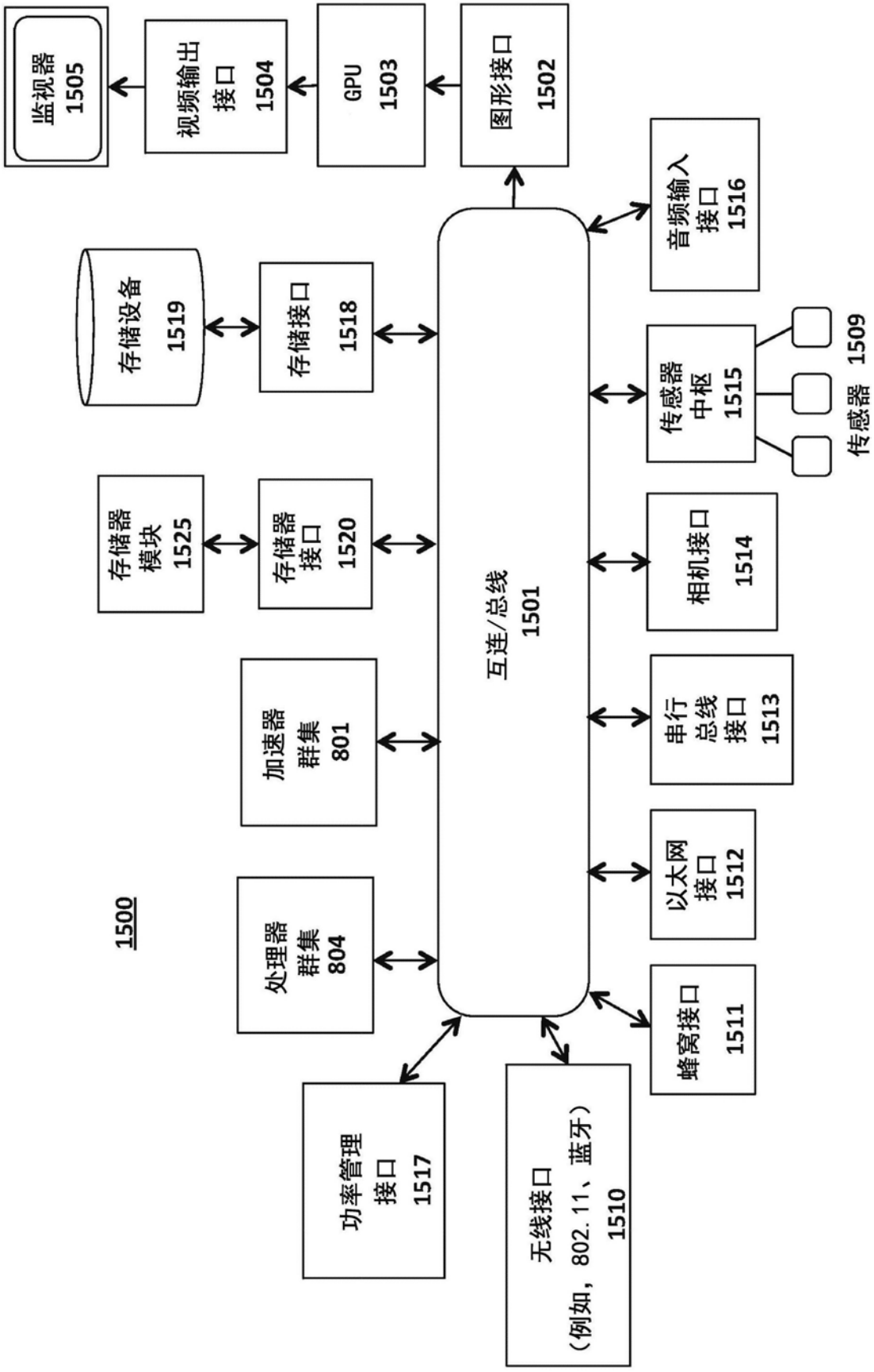
1500
互连/总线
1501
监视器
1505
视频输出
接口
1504
GPU
1503
图形接口
1502
音频输入
接口
1516
存储设备
1519
存储接口
1518
传感器
中枢
1515
传感器 1509
存储器
模块
1525
存储器
接口
1520
相机接口
1514
加速器
群集
801
串行
总线
接口
1513
处理器
群集
804
以太网
接口
1512
蜂窝接口
1511
功率管理
接口
1517
无线接口
（例如，802.11、蓝牙）
1510

图15