(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111344789 B
(45)授权公告日 2023.09.05

(21)申请号 201880073154.X
(22)申请日 2018.11.02
(65)同一申请的已公布的文献号
申请公布号 CN 111344789 A
(43)申请公布日 2020.06.26
(30)优先权数据
15/811,943 2017.11.14 US
(85)PCT国际申请进入国家阶段日
2020.05.12
(86)PCT国际申请的申请数据
PCT/IB2018/058619 2018.11.02
(87)PCT国际申请的公布数据
WO2019/097347 EN 2019.05.23
(73)专利权人 国际商业机器公司
地址 美国纽约
(72)发明人 M·K·克施温德 V·萨拉普拉
(74)专利代理机构 北京市中咨律师事务所 11247
专利代理师 刘都 于静
(51)Int.Cl.
G11C 8/00(2006.01)
(56)对比文件
US 2017031685 A1,2017.02.02
US 2017177364 A1,2017.06.22
CN 102163141 A,2011.08.24
CN 105938459 A,2016.09.14
US 2014283040 A1,2014.09.18
审查员 李勇

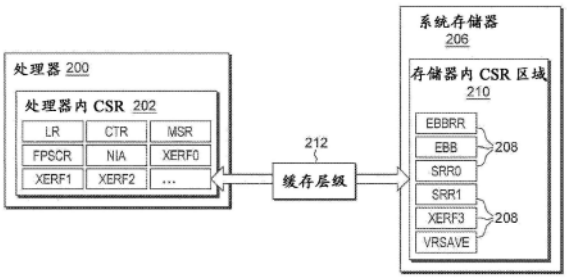
权利要求书2页 说明书37页 附图45页

(54)发明名称

基于功能亲和度分组的配置状态寄存器

(57)摘要

基于功能亲和度分组的配置状态寄存器。获得针对其分配了存储器的存储器内配置状态寄存器的标识。基于该标识，确定到存储器的偏移，该存储器内配置状态寄存器被存储在该偏移处。偏移是基于存储器内配置状态寄存器的功能亲和度而被分配给存储器内配置状态寄存器的。至少使用该偏移来访问存储器内配置状态寄存器。



CN 111344789 B

1.一种计算机可读存储介质,能够由处理电路读取并且存储用于执行用于促进计算机环境内的处理的方法的指令,所述方法包括:

获取针对其分配了存储器的存储器内配置状态寄存器的标识,其中:

所述存储器内配置状态寄存器为被配置用于特定操作的寄存器,并且所述存储器内配置状态寄存器为被配置用于该特定操作的多个存储器内配置状态寄存器中的一个,

所述多个存储器内配置状态寄存器中的一个具有与用于该特定操作的一个或多个其他存储器内配置状态寄存器不连续的寄存器编号,

所述多个存储器内配置状态寄存器基于其用于该特定操作被一起放置在存储器内,

该特定操作为基于事件的分支处理;

基于所述标识来确定到所述存储器的偏移,其中所述存储器内配置状态寄存器被存储在所述偏移处,所述偏移是基于为该存储器内配置状态寄存器配置的该特定操作分配给该存储器内配置状态寄存器的;以及

至少使用所述偏移来访问所述存储器内配置状态寄存器。

2.根据权利要求1所述的计算机可读存储介质,其中,所述偏移将所述存储器内配置状态寄存器置于与处理该特定操作使用的所述多个存储器内配置状态寄存器中的至少一个其他存储器内配置状态寄存器相同的缓存线中。

3.根据权利要求1所述的计算机可读存储介质,其中,所述偏移将所述存储器内配置状态寄存器置于与处理该特定操作使用的所述多个存储器内配置状态寄存器中的至少一个其他存储器内配置状态寄存器相邻的缓存线中。

4.根据权利要求1所述的计算机可读存储介质,其中,所述偏移是在所述存储器的特定单元内的索引位置。

5.根据权利要求4所述的计算机可读存储介质,其中,所述方法进一步包括:提供所述存储器的所述特定单元的版本指示。

6.根据权利要求1所述的计算机可读存储介质,其中,所述标识包括所述存储器内配置状态寄存器的寄存器编号。

7.根据权利要求1所述的计算机可读存储介质,其中,所述确定所述偏移包括在数据结构中执行查找。

8.根据权利要求1所述的计算机可读存储介质,其中,所述确定所述偏移包括使用计算来确定所述偏移。

9.根据权利要求1所述的计算机可读存储介质,其中,所述方法还包括:将所述偏移返回给请求者,以便所述请求者访问所述存储器内配置状态寄存器。

10.一种用于促进计算环境内的处理的计算机系统,所述计算机系统包括:

存储器;以及

与所述存储器通信的处理器,其中,所述计算机系统被配置为执行方法,所述方法包括:

获取针对其分配了存储器的存储器内配置状态寄存器的标识,其中:

所述存储器内配置状态寄存器为被配置用于特定操作的寄存器,并且所述存储器内配置状态寄存器为被配置用于该特定操作的多个存储器内配置状态寄存器中的一个,

所述多个存储器内配置状态寄存器中的一个具有与用于该特定操作的一个或多个其

他存储器内配置状态寄存器不连续的寄存器编号，

所述多个存储器内配置状态寄存器基于其用于该特定操作被一起放置在存储器内，

该特定操作为基于事件的分支处理；

基于所述标识来确定到所述存储器的偏移，其中所述存储器内配置状态寄存器被存储在所述偏移处，所述偏移是基于为该存储器内配置状态寄存器配置的该特定操作分配给该存储器内配置状态寄存器的；以及

至少使用所述偏移来访问所述存储器内配置状态寄存器。

11.根据权利要求10所述的计算机系统，其中，所述偏移将所述存储器内配置状态寄存器置于与处理该特定操作使用的所述多个存储器内配置状态寄存器中的至少一个其他存储器内配置状态寄存器相同的缓存线中。

12.根据权利要求10所述的计算机系统，其中，所述偏移将所述存储器内配置状态寄存器置于与处理该特定操作使用的所述多个存储器内配置状态寄存器中的至少一个其他存储器内配置状态寄存器相邻的缓存线中。

13.根据权利要求10所述的计算机系统，其中，所述偏移是在所述存储器的特定单元内的索引位置，并且其中，所述方法进一步包括：提供所述存储器的所述特定单元的版本指示。

14.一种促进计算环境内的处理的计算机实现的方法，所述计算机实现的方法包括：

获取针对其分配了存储器的存储器内配置状态寄存器的标识，其中：

所述存储器内配置状态寄存器为被配置用于特定操作的寄存器，并且所述存储器内配置状态寄存器为被配置用于该特定操作的多个存储器内配置状态寄存器中的一个，

所述多个存储器内配置状态寄存器中的一个具有与用于该特定操作的一个或多个其他存储器内配置状态寄存器不连续的寄存器编号，

所述多个存储器内配置状态寄存器基于其用于该特定操作被一起放置在存储器内，

该特定操作为基于事件的分支处理；

基于所述标识来确定到所述存储器的偏移，其中所述存储器内配置状态寄存器被存储在所述偏移处，所述偏移是基于为该存储器内配置状态寄存器配置的该特定操作分配给该存储器内配置状态寄存器的；以及

至少使用所述偏移来访问所述存储器内配置状态寄存器。

15.根据权利要求14所述的计算机实现的方法，其中，所述偏移将所述存储器内配置状态寄存器置于与处理该特定操作使用的所述多个存储器内配置状态寄存器中的至少一个其他存储器内配置状态寄存器相同的缓存线中。

16.根据权利要求14所述的计算机实现的方法，其中，所述偏移将所述存储器内配置状态寄存器置于与处理该特定操作使用的所述多个存储器内配置状态寄存器中的至少一个其他存储器内配置状态寄存器相邻的缓存线中。

17.根据权利要求14所述的计算机实现的方法，其中，所述偏移是在所述存储器的特定单元内的索引位置，并且其中，所述方法还包括：提供所述存储器的所述特定单元的版本指示。

基于功能亲和度分组的配置状态寄存器

技术领域

[0001] 一个或多个方面一般涉及计算环境内的处理，并且特别地涉及促进该处理。

背景技术

[0002] 计算环境的计算机包括控制计算机内的处理的中央处理单元(CPU)或处理器。中央处理单元的行为由控制寄存器控制。控制寄存器是执行特定任务的处理器寄存器，例如，诸如中断控制、切换寻址模式、分页控制和/或协处理器控制。

[0003] 控制寄存器通常被实现为锁存器，例如直接在处理器芯片上的固态元件。一些计算机使用大量的控制寄存器，如计算机的架构实现所定义的。因此，控制寄存器代表芯片的增长区域。

[0004] 此外，一些计算机支持多线程，其中中央处理单元可以同时执行多个进程或线程。每个线程使用单独的一组控制寄存器，因此会增加芯片上控制寄存器的数量。

[0005] 基于锁存器的控制寄存器的数量增加可能影响性能、芯片面积和/或功率消耗。例如，控制寄存器在上下文切换期间被切换，因此，控制寄存器的数量的提高增加了上下文切换的成本。此外，对于基于锁存器的控制寄存器，对控制的更新以编程顺序发生，这也可能影响性能。

[0006] 不同的架构可具有针对控制寄存器的不同的名称。例如，在纽约阿蒙克市的国际商业机器公司提供的Power架构中，控制寄存器被称为专用寄存器(SPR)。其它架构可使用其它名称。本文对控制寄存器的使用包括其它名称的控制寄存器，包括例如SPR以及其它。

发明内容

[0007] 通过提供用于促进计算环境内的处理的计算机程序产品来克服现有技术的缺点且提供额外的优点。计算机程序产品包括可由处理电路读取并且存储用于执行方法的指令的计算机可读存储介质。该方法包括例如获取针对其分配了存储器的存储器内配置状态寄存器的标识。基于该标识来确定到存储器中的偏移，其中存储器内配置状态寄存器被存储在该偏移处。偏移是基于存储器内配置状态寄存器的功能亲和度而被分配给存储器内配置状态寄存器的。至少使用偏移来访问存储器内配置状态寄存器。

[0008] 作为示例，所分配的偏移将存储器内配置状态寄存器置于与具有相同的功能亲和度的另一存储器内配置状态寄存器相同的缓存线中。这在访问存储器时提供了效率，促进了处理并改进了性能。

[0009] 在示例中，基于存储器内配置状态寄存器和另一存储器内配置状态寄存器两者在特定操作中被使用，存储器内配置状态寄存器和另一存储器内配置状态寄存器具有相同的功能亲和度。

[0010] 作为另一示例，所分配的偏移将存储器内配置状态寄存器置于与具有相同的功能亲和度的另一存储器内配置状态寄存器相邻的缓存线中。这在访问存储器时提供了效率，促进了处理并改进了性能。

[0011] 在示例中，偏移是在存储器的特定单元内的索引位置。此外，在示例中，提供存储器的特定单元的版本指示。版本指示提供了存储器管理中的灵活性并便于处理。

[0012] 在方面，该标识包括存储器内配置状态寄存器的寄存器编号。

[0013] 此外，在实施例中，确定偏移包括在数据结构中执行查找。在另一实施例中，确定偏移包括使用计算来确定偏移。

[0014] 在一方面，将偏移返回给请求者，以便请求者访问存储器内配置状态寄存器。

[0015] 本文还描述并要求保护与一个或多个方面相关的计算机实现的方法和系统。此外，本文还描述并要求保护与一个或多个方面相关的服务。

[0016] 通过本文描述的技术实现了额外的特征和优点。在本文中还详细描述了其它实施例和方面，并且被视为所要求的方面的一部分。

附图说明

[0017] 一个或多个方面作为示例在说明书结尾处的权利要求中被特别指出并清楚地要求保护。从下面结合附图的详细描述中，一个或多个方面的前述和目的、特征和优点将变得显而易见，在附图中：

[0018] 图1A描绘了用于结合和使用本发明的一个或多个方面的计算环境的示例；

[0019] 图1B描绘了用于结合和使用本发明的一个或多个方面的计算环境的另一示例；

[0020] 图1C描绘了根据本发明的一个或多个方面的图1A或图1B的处理器的进一步细节；

[0021] 图1D描绘了根据本发明的一个或多个方面使用的指令执行管线的示例的进一步细节；

[0022] 图1E描绘了根据本发明的一方面的处理器的示例的进一步细节；

[0023] 图2描绘了根据本发明的一方面的处理器内配置状态寄存器和存储器内配置状态寄存器的示例；

[0024] 图3描绘了根据本发明的一方面的与使用存储器内配置状态寄存器相关联的解码逻辑的示例；

[0025] 图4描绘了根据本发明的一方面的加载配置状态寄存器内部操作的示例；

[0026] 图5描绘了根据本发明的一方面的存储配置状态寄存器内部操作的示例；

[0027] 图6描绘了根据本发明的一方面的使用存储器内配置状态寄存器的示例；

[0028] 图7描绘了根据本发明的一方面的使用存储器内配置状态寄存器的另一示例；

[0029] 图8描绘了根据本发明的一方面的配置状态寄存器写入操作的示例；

[0030] 图9描绘了根据本发明的一方面的配置状态寄存器读取操作的示例；

[0031] 图10描绘了根据本发明的一方面的与向配置状态寄存器的移动或从配置状态寄存器的移动相关联的解码逻辑的实施例；

[0032] 图11描绘了根据本发明的一方面的与移动到配置状态寄存器指令相关联的进一步细节；

[0033] 图12描绘了根据本发明的一方面的从配置状态寄存器移动指令的进一步细节；

[0034] 图13A描绘了根据本发明的一方面的与复合配置状态寄存器读引用相关联的逻辑的一个实施例；

[0035] 图13B描述了根据本发明的一方面的与复合配置状态寄存器写引用相关联的逻辑

的一个实施例；

[0036] 图14描绘了根据本发明的一方面的复合配置状态寄存器的示例；

[0037] 图15A-15B描绘了根据本发明的一方面的配置状态寄存器的线性映射的示例；

[0038] 图16描绘了根据本发明的一方面的用于配置状态寄存器的重映射流逻辑的示例；

[0039] 图17A描绘了多配置状态寄存器存储操作的示例；

[0040] 图17B描绘了根据本发明的一方面的批量(bulk)存储配置状态寄存器操作的示例；

[0041] 图17C描绘了根据本发明的一方面的批量加载配置状态寄存器操作的示例；

[0042] 图18A描绘了根据本发明的一方面的指定架构配置控制的示例；

[0043] 图18B描绘了根据本发明的一方面的指定架构配置控制的另一示例；

[0044] 图19A描绘了根据本发明的一方面的执行上下文切换的示例；

[0045] 图19B描绘了根据本发明的一方面的执行上下文切换的另一示例；

[0046] 图20描绘了根据本发明的一方面的与移动到配置状态寄存器操作相关联的地址转换的实施例；

[0047] 图21A-21B描绘了根据本发明的各方面执行动态地址转换的示例；

[0048] 图22描绘了根据本发明的一方面的页表条目的示例；

[0049] 图23描述了根据本发明的一方面的与特定上下文相关联的特定配置状态寄存器的示例；

[0050] 图24描绘了根据本发明的一方面的向主机系统提供固定(pinning)通知的实施例；

[0051] 图25描绘了根据本发明的一方面的在页表条目中指定固定(pin)操作的实施例；

[0052] 图26描绘了根据本发明的一方面的在页表条目中指定解除固定(unpin)操作的实施例；

[0053] 图27描绘了根据本发明的一方面的在一个管理程序调用中组合固定和解除固定操作的示例；

[0054] 图28描绘了根据本发明的一方面的与基于单个调用执行固定和解除固定操作相关联的进一步细节；

[0055] 图29A-29C提供根据本发明的一个或多个方面的数据写入的各种示例；

[0056] 图30A-30C提供根据本发明的一个或多个方面的数据读取的各种示例；

[0057] 图31A-31B描绘了根据本发明的一方面的促进计算环境内的处理的实施例；

[0058] 图32A描绘了用于结合和使用本发明的一个或多个方面的计算环境的另一示例；

[0059] 图32B描绘了图32A的存储器的进一步细节；

[0060] 图33描绘了云计算环境的实施例；以及

[0061] 图34描绘了抽象模型层的示例。

具体实施方式

[0062] 根据本发明的一个方面，各种配置状态寄存器被提供在存储器内而不是在处理器内。如本文所使用的，术语"配置状态寄存器"包括控制寄存器、机器状态寄存器(MSR)(例如，程序状态字(PSW)或其它机器状态寄存器)、状态寄存器(例如，浮点状态控制寄存器)、

专用寄存器(SPR)、配置寄存器、和/或配置例如指令的操作的其它寄存器。

[0063] 所选择的配置状态寄存器(或在另一方面中,其部分)被提供在存储器内,其中这些寄存器被映射到系统存储器并且被包括在存储器层级中,该存储器层级耦合到处理器但与处理器分离。存储器层级结构包括例如加载/存储队列、一个或多个存储器缓存和系统存储器(本文也称为主存储器、中央存储装置、存储装置、主存储装置、存储器)。基于寄存器在存储器内而不是在处理器内,通过使用存储器地址来访问寄存器,并且访问请求可以被重新排序或推测性地处理。相反,针对处理器内的配置状态寄存器的访问请求不是被乱序地或推测性地处理。处理器内的配置状态寄存器被实现为例如固态元件(例如锁存器),例如直接在芯片上的。芯片上表示或涉及在单个集成电路中或与给定设备相同的集成电路中包括的电路。

[0064] 基于配置状态寄存器被存储在系统存储器中,某些指令(例如,移动到配置状态寄存器指令(例如,移动到SPR(mtspr)指令)和从配置状态寄存器移动指令(例如,从SPR移动(mfspr)指令))被指令解码逻辑的加载和存储指令或操作所替代。将所生成的加载和存储指令/操作提交到存储队列,并且执行典型的加载和存储处理。

[0065] 作为一个示例,包括配置状态寄存器的存储区域由操作系统和/或管理程序定义,并且留出用于存储基于存储器的寄存器。在一个实施例中,在架构上指定物理存储器区域(例如,物理存储器的第一或最后n个页)。

[0066] 在另一方面中,配置状态寄存器的一个或多个部分被提供在存储器内,而配置状态寄存器的一个或多个其它部分被提供在处理器内。在一个示例中,存储器内提供的部分是较不频繁使用的部分。

[0067] 在又一方面中,提供配置状态寄存器的重新映射,使得通常一起使用的配置状态寄存器(或配置状态寄存器的至少一部分)一起被放置在存储器内(例如,在单个缓存线或相邻的缓存线中)以改进处理性能。

[0068] 此外,在另一方面,提供指令或操作以执行多个配置状态寄存器的批量存储或加载。这是为了便于例如上下文切换,从而提高其性能。

[0069] 此外,在一个方面,通过定义一组控制来识别配置状态寄存器被存储在存储器中的位置,来促进处理并改进性能。

[0070] 在另一方面,通过操纵存储器内配置状态寄存器的存储器指针在上下文切换期间实现了效率。指针被操纵,而不是复制旧的配置数据。这通过在上下文切换期间提高速度并降低复杂度来改进计算环境内的处理。

[0071] 此外,在另一方面中,基于执行加载将用作基地址的地址的指令,地址转换被自动执行以便避免稍后在处理指令时的潜在页错误。

[0072] 在又一方面,配置状态寄存器由上下文或组(例如,管理程序、操作系统、进程、线程)来隔离,以通过增加管理灵活性来促进处理。

[0073] 作为另一方面,提供了对初始化的存储器后备状态的自动固定的指示。

[0074] 此外,在另一方面,使用半虚拟化的固定调用来有效地管理存储器页的固定。

[0075] 更进一步,在一个方面,保护系统存储器免受单事件干扰。

[0076] 本文描述了各个方面。此外,在不背离本发明的各方面的精神的情况下,许多变化是可能的。应当注意,除非另外不一致,否则本文描述的每个方面或特征及其变型可与任何

其它方面或特征组合。

[0077] 参见图1A描述了用于结合和使用本发明的一个或多个方面的计算环境的一个示例。在一个示例中，计算环境基于由纽约州阿蒙克市的国际商业机器公司(International Business Machines Corporation)提供的z/架构(z/Architecture)。在IBM公开案第SA22-7832-10号(2015年3月)“z/架构工作原理(z/Architecture Principles of Operation)”中描述了z/架构的一个实施例，其通过全文引用并入本文。注意，Z/ARCHITECTURE是美国纽约州阿蒙克市国际商业机器公司的注册商标。

[0078] 在另一示例中，计算环境基于由纽约州阿蒙克市国际商业机器公司提供的Power架构。在国际商业机器公司2015年4月9日的“Power ISA™版本2.07B”中描述了Power架构的一个实施例，其通过全文引用并入本文。进一步注意，POWER ARCHITECTURE是美国纽约州阿蒙克市国际商业机器公司的注册商标。

[0079] 计算环境还可基于其它架构，包括但不限于英特尔x86架构。也存在其它示例。

[0080] 如图1A所示，计算环境100包括例如以通用计算设备的形式示出的计算机系统102。计算机系统102可以包括但不限于经由一个或多个总线和/或其它连接110彼此耦合的一个或多个处理器或处理单元104(例如，中央处理单元(CPU))、存储器106(也称为系统存储器、主存储器、主存储装置、中央存储装置或存储装置)、以及一个或多个输入/输出(I/O)接口108。

[0081] 总线110表示几类总线结构中的一种或多种，包括存储器总线或者存储器控制器，外围总线，图形加速端口，处理器或者使用多种总线结构中的任意总线结构的局域总线。举例来说，这些架构包括但不限于工业标准架构(ISA)，微通道架构(MAC)，增强型ISA、视频电子标准协会(VESA)局域总线以及外围组件互连(PCI)。

[0082] 存储器106可以包括例如缓存器120，诸如共享缓存器，其可以耦合到处理器104的本地缓存器122。此外，存储器106可以包括一个或多个程序或应用130、操作系统132和一个或多个计算机可读程序指令134。计算机可读程序指令134可被配置为执行本发明的各方面的实施例的功能。

[0083] 计算机系统102还可以经由例如I/O接口108与一个或多个外部设备140、一个或多个网络接口142、和/或一个或多个数据存储设备144进行通信。示例性外部设备包括用户终端、磁带驱动、指向设备、显示器等。网络接口142使得计算机系统102能够与一个或多个网络(诸如，局域网(LAN)、广域网(WAN)和/或公用网络(例如，因特网))进行通信，提供与其它计算设备或系统的通信。

[0084] 数据存储设备144可存储一个或多个程序146、一个或多个计算机可读程序指令148和/或数据等。计算机可读程序指令可被配置为执行本发明的方面的实施例的功能。

[0085] 计算机系统102可以包括和/或耦接到可移动/不可移动、易失性/非易失性计算机系统存储介质。例如，计算机系统102可包括和/或耦接到不可移动、非易失性磁性介质(通常称为“硬盘驱动”)、用于从可移动非易失性磁盘(例如，“软盘”)读取及写入至可移动非易失性磁盘的磁盘驱动器，和/或用于从可移动非易失性光盘(诸如，CD-ROM、DVD-ROM或其它光学介质)读取或写入至可移动非易失性光盘的光盘驱动。应理解，其它硬件和/或软件组件可与计算机系统102结合使用。示例包括但不限于：微代码、设备驱动、冗余处理单元、外部磁盘驱动阵列、RAID系统、磁带驱动、以及数据档案存储系统等。

[0086] 计算机系统102可以与众多其它通用或专用计算系统环境或配置一起操作。可适合于供计算机系统102使用的公知的计算系统、环境和/或配置的示例包括但不限于个人计算机(PC)系统、服务器计算机系统、瘦客户机、厚客户机、手持型或膝上型设备、多处理器系统、基于微处理器的系统、机顶盒、可编程消费电子产品、网络PC、小型计算机系统、主计算机系统、以及包括任何上述系统或设备的分布式云计算环境等。

[0087] 在另一实施例中,计算环境支持虚拟机。参见图1B描述了这种环境的一个示例。在一个示例中,计算环境161包括提供虚拟机支持的中央处理器复合体(CPC)163。CPC 163经由一个或多个控制单元169耦合到一个或多个输入/输出(I/O)设备167。中央处理器复合体163包括例如耦合到一个或多个处理器(又名中央处理单元(CPU))171的存储器165(又名系统存储器、主存储器、主存储装置、中央存储装置、存储装置)以及输入/输出子系统173,下面描述其中的每一个。

[0088] 存储器165包括例如一个或多个虚拟机175、管理虚拟机的虚拟机管理器(诸如管理程序177)和处理器固件179。管理程序177的一个示例是由纽约阿蒙克市的国际商业机器公司提供的z/VM。管理程序有时被称为主机。此外,如本文所使用的,固件包括例如处理器的微代码。其包括例如在实现更高级的机器代码中使用的硬件级指令和/或数据结构。在一个实施例中,其包括例如专有代码,该专有代码通常作为包括可信软件的微代码或底层硬件专用的微代码来传递,并且控制操作系统对系统硬件的访问。

[0089] CPC的虚拟机支持提供了操作大量虚拟机175的能力,每个虚拟机能够与不同的程序185一起操作并且运行客户操作系统183,诸如Linux。每个虚拟机175能够用作单独的系统。也就是说,每个虚拟机可以被独立地重置,运行客户机操作系统,并且利用不同的程序来操作。在虚拟机中运行的操作系统或应用程序看起来可以访问整个系统,但是实际上,仅系统的一部分是可用的。

[0090] 存储器165被耦合到处理器(例如CPU)171,其是可分配给虚拟机的物理处理器资源。例如,虚拟机175包括一个或多个逻辑处理器,每个逻辑处理器表示可以动态分配给虚拟机的物理处理器资源171的全部或份额。

[0091] 此外,存储器165被耦合到I/O子系统173。输入/输出子系统173引导输入/输出控制单元169和设备167与主存储器165之间的信息流。输入/输出子系统611被耦合到中央处理复合体,因为其可以是中央处理复合体的一部分或者与中央处理复合体分离。

[0092] 参见图1C描述了关于处理器的一个示例的进一步细节,诸如处理器104(或处理器171)。处理器(诸如处理器104(或处理器171))包括用于执行指令的多个功能组件。这些功能组件例如包括指令提取组件150,用于提取要执行的指令;指令解码单元152,用于对所提取的指令进行解码并获取所解码的指令的操作数;指令执行组件154,用于执行所解码的指令;存储器访问组件156,用于在必要时访问存储器以用于指令执行;以及写回组件160,用于提供所执行指令的结果。根据本发明的一个方面,这些组件中的一个或多个可以用于执行与基于存储器的配置状态寄存器处理166相关联的一个或多个指令和/或操作。

[0093] 在一个实施例中,处理器104(或处理器171)还包括一个或多个寄存器168,以由一个或多个功能组件使用。处理器104(或处理器171)可以包括比本文提供的示例更多、更少和/或其它的组件。

[0094] 参见图1D来描述关于处理器(诸如处理器104或处理器171)的执行管线的进一步

细节。尽管在此描绘和描述了管线的各个处理阶段，但是应当理解，在不脱离本发明的各方面的精神的情况下，可以使用附加的、更少的和/或其它阶段。

[0095] 参见图1D，在一个实施例中，指令从指令队列中提取170，并且可以执行该指令的分支预测172和/或解码174。解码的指令可被添加到一组指令176以一起处理。分组的指令被提供给映射器178，该映射器确定任何依赖性、分配资源并将一组指令/操作分派到适当的发布队列。对于不同类型的执行单元，存在一个或多个发布队列，包括例如分支、加载/存储、浮点、定点、向量等。在发布阶段180期间，指令/操作被发布到适当的执行单元。读取任何寄存器182以提取其源，并且指令/操作在执行阶段184期间执行。如所指出的，作为示例，执行可以是用于分支、加载(LD)或存储(ST)、定点操作(FX)、浮点操作(FP)或向量操作(VX)的。在写回阶段186期间，将任何结果写入到适当的寄存器。随后，指令完成188。如果存在中断或清除190，则处理可返回到指令提取170。

[0096] 此外，在一个示例中，寄存器重命名单元192被耦合到解码单元，寄存器重命名单元192可用于寄存器的保存/恢复。

[0097] 参见图1E来描述关于处理器的附加细节。在一个示例中，处理器(诸如，处理器104(或处理器171))是管线式处理器，其可包括例如预测硬件、寄存器、缓存器、解码器、指令定序单元和指令执行单元。预测硬件包括例如局部分支历史表(BHT)105a、全局分支历史表(BHT)105b和全局选择器105c。通过指令提取地址寄存器(IFAR)107访问预测硬件，该指令提取地址寄存器107具有用于下一个指令提取的地址。

[0098] 还将同一地址提供到指令缓存器109，其可提取被称为"提取组"的多个指令。目录111与指令缓存器109相关联。

[0099] 缓存器和预测硬件在大致相同的时间以相同的地址被访问。如果预测硬件具有可用于提取组中的指令的预测信息，则将该预测转发到指令定序单元(ISU)113，其转而将指令发布到执行单元以供执行。预测可用于结合分支目标计算115和分支目标预测硬件(例如链接寄存器预测栈117a和计数寄存器栈117b)来更新IFAR107。如果没有预测信息可用，但一个或多个指令解码器119在提取组中发现分支指令，则针对该提取组而创建预测。所预测的分支被存储在预测硬件中，诸如分支信息队列(BIQ)125，并且被转发到ISU 113。

[0100] 分支执行单元(BRU)121响应于由ISU 113向其发出的指令而操作。BRU 121具有对条件寄存器(CR)文件123的读取访问。分支执行单元121进一步具有对分支扫描逻辑存储在分支信息队列125中的信息的访问，以确定分支预测的成功，并且可操作地耦合到与微处理器支持的一个或多个线程相对应的指令提取地址寄存器(IFAR)107。根据至少一个实施例，BIQ条目与标识符相关联，并由标识符识别，例如由分支标签BTAG识别。当与BIQ条目相关联的分支完成时，对其进行如此标记。BIQ条目被保持在队列中，并且当最旧的队列条目被标记为包括与已完成的分支相关联的信息时，该队列条目被顺序地解除分配。BRU 121还可操作地被耦合，以便当BRU 121发现分支误预测时引起预测器更新。

[0101] 当执行指令时，BRU 121检测预测是否错误。如果是，预测将被更新。为此目的，处理器还包括预测器更新逻辑127。预测器更新逻辑127响应于来自分支执行单元121的更新指示并且被配置为更新局部BHT 105a、全局BHT 105b和全局选择器105c中的一者或多者中的阵列条目。预测器硬件105a、105b和105c可具有由指令提取和预测操作所使用的与读取端口不同的写入端口，或者可共享单个读/写端口。预测器更新逻辑127可进一步操作地耦

合到链接栈117a和计数寄存器栈117b。

[0102] 现在参考条件寄存器文件(CRF)123,CRF123可由BRU 121读访问,并且可以由执行单元写入,执行单元包括但不限于定点单元(FXU)141、浮点单元(FPU)143和向量多媒体扩展单元(VMXU)145。条件寄存器逻辑执行单元(CRL执行)147(也称为CRU)和专用寄存器(SPR)处理逻辑149具有对条件寄存器文件(CRF)123的读和写访问。CRU 147对存储于CRF文件123中的条件寄存器执行逻辑运算。FXU 141能够对CRF 123执行写更新。

[0103] 处理器104(或处理器171)还包括加载/存储单元151、各种复用器153和缓冲器155、以及地址转换表157和其它电路。

[0104] 参见图2描述了关于由处理器200(诸如处理器104或处理器171)使用的各种寄存器的进一步细节。如图所示,处理器200包括多个处理器内配置状态寄存器(CSR)202。作为示例,处理器内配置状态寄存器包括链接寄存器(LR)、计数器寄存器(CTR)、机器状态寄存器(MSR)、浮点状态控制寄存器(FPSCR)、下一指令地址(NIA)寄存器以及一个或多个整数异常寄存器(XER)寄存器。此外,根据本发明的一个方面,耦合到处理器200的系统存储器206包括一个或多个存储器内配置状态寄存器208。作为示例,存储器内配置状态寄存器包括基于事件的分支返回寄存器(EBBRR)、基于事件的分支寄存器(EBB)、状态恢复寄存器(SRR)、整数异常寄存器(XER)、以及向量寄存器保存寄存器(VRSAVE)。在一个示例中,存储器内配置状态寄存器208被存储在系统存储器206的存储器内配置状态寄存器区域210中。

[0105] 被频繁访问(例如,在一行中的若干次访问)的配置状态寄存器可以被移动到缓存层级212,缓存层级212耦合到处理器200和系统存储器206。

[0106] 根据一个方面,基于一个或多个配置状态寄存器被移动或放置在存储器内,对配置状态寄存器的处理器内访问被对存储器的访问所替代。参见图3描述了确定访问类型的解码逻辑的一个示例。该处理由例如处理器的解码单元和/或另一单元执行。

[0107] 参见图3,首先,接收指令,步骤300。在询问302,确定该指令是否是移动到配置状态寄存器指令,例如移动到SPR(mtspr)指令。如果指令是移动到配置状态寄存器指令,则在询问304,进一步确定指令中指示的配置状态寄存器是否是存储器内配置状态寄存器。如果不是,则执行移动到配置状态寄存器指令的常规处理,步骤306。然而,如果配置状态寄存器是存储器内配置状态寄存器,则生成存储配置状态寄存器内部操作以将配置状态寄存器存储在存储器中(例如,将配置状态寄存器的新内容存储在存储器中),步骤308。

[0108] 返回询问302,如果所接收的指令不是移动到配置状态寄存器指令,则进一步确定该指令是否是从配置状态寄存器移动指令,例如从SPR移动(mfspr)指令,询问312。如果指令是从配置状态寄存器移动指令,则确定指令中指示的配置状态寄存器是否在存储器内,询问314。如果不是,则执行从配置状态寄存器移动的常规(conventional)处理,步骤316。否则,生成加载配置状态寄存器内部操作以从存储器获得寄存器的内容,步骤318。

[0109] 返回到询问312,如果所接收的指令不是移动到配置状态寄存器指令或从配置状态寄存器移动指令,则可执行进一步确定以确定所接收的指令是否为使用配置状态寄存器的另一指令,询问322。如果是,则可取决于正由指令执行的功能而产生读取和/或写入内部操作,步骤324。否则,处理继续到步骤332,其中执行常规指令解码处理。

[0110] 在本发明的其它方面,例如,响应于根据接收到中断请求而进入异常处理程序,执行加载配置状态寄存器和存储配置状态寄存器值的内部操作,连同执行不对应于指令的处

理器操作。

[0111] 参见图4描述了关于加载配置状态寄存器内部操作的进一步细节。该处理由处理器执行。参见图4，首先，从寄存器或位置(例如，基址寄存器，诸如线程控制基址寄存器(TCBR))获得存储器基地址(基址(base))，寄存器或位置包含存储器单元(例如，存储器页)的地址，该存储器单元的地址是包括配置状态寄存器的存储器的基地址，步骤400。另外，获得操作中指示的寄存器编号，步骤402。将寄存器编号映射到存储器中的偏移，步骤404。例如，每个配置状态寄存器编号(或在另一实施例中的其它标识)被映射到存储器中的特定位置。该位置是距基地址的特定量(例如，偏移)。然后，执行从地址(基地址加偏移)的加载，步骤406，并且返回加载的值，步骤408。

[0112] 如本文所使用的，基址是指包括存储器内配置状态寄存器的存储器的基地址，并且基址寄存器是指包括基址的寄存器。基址寄存器的一个示例是线程控制基址寄存器(TCBR)，但是其它上下文(例如，操作系统等)可以使用其它基址寄存器。

[0113] 参见图5描述了关于存储配置状态寄存器内部操作的进一步细节。该处理由处理器执行。参照图5，首先，例如从基址寄存器(例如，从TCBR)获得存储器基地址(基址)，步骤500，以及获得在操作中指示的寄存器编号，步骤502。将寄存器编号映射到存储器中的偏移处，步骤504，且将存储操作数(例如，寄存器的内容)存储在由基地址加偏移所指定的地址处，步骤506。

[0114] 如上所述，与从配置状态寄存器移动指令或移动到配置状态寄存器指令不同的指令可以使用配置状态寄存器。因此，参见图6描述了与这些指令中的一者相关联的处理。该处理由处理器执行。参见图6，在该实施例中，获得包括配置状态寄存器读取引用的指令/操作，步骤600。基于此，例如从基址寄存器(例如，从TCBR)获得指令/操作中指示的配置状态寄存器的存储器基地址(基址)，步骤602，以及获得指令/操作中指示的寄存器编号，步骤604。将寄存器编号映射到存储器中的偏移处，步骤606，并且将来自由基址加偏移所指定的地址的内容加载到临时寄存器中，步骤608。临时寄存器然后被使用，步骤610。

[0115] 如参见图7所述，执行用于配置状态寄存器写入引用的类似的处理。该处理由处理器执行。参见图7，在一个示例中，获得配置状态寄存器写入引用，步骤700。基于此，例如从基址寄存器(例如，从TCBR)获得在指令/操作中指示的配置状态寄存器的存储器基地址(基址)，步骤702，此外，获得在指令/操作中指定的寄存器编号，步骤704。将寄存器编号映射到偏移处，步骤706，并且将写入引用中包括的内容(例如，临时寄存器中)存储在基址加偏移处指定的地址处，步骤708。

[0116] 参见图8描述了关于配置状态寄存器写入操作(例如移动到配置状态寄存器(例如mtspr))的操作实现图的进一步细节。该处理由处理器执行。参见图8，在一个示例中，转换操作中指定的寄存器编号，步骤800。例如，确定(例如，通过使用查找表或计算)对应于或映射到寄存器编号(或其它指示)的存储器地址。此外，分配存储队列条目，步骤802，并且将对应于对象配置状态寄存器的地址存储到存储队列条目中，步骤804。此外，将对应于写入到对象配置状态寄存器的数据的内容(例如，数据值)写入到存储队列条目中，步骤806。在一个示例中，步骤804和806可被无序地执行。

[0117] 针对所指示的地址的读取来监测存储队列条目(例如，从存储队列旁路)，步骤808。此外，可以基于错误预测事件来清除存储队列条目，在一个示例中，错误预测事件可以

发生到架构顺序点，步骤810。

[0118] 将内容(例如，数据值)写入到存储器层级中的地址，例如，第一级缓存器，步骤812。基于读取请求提供来自第一级缓存器的数据，步骤814。此外，基于缓存替换策略，将数据从第一级缓存器发出到缓存层级中的一个或多个下一级，步骤816。基于读取请求提供来自缓存层级的一个或多个下一级的数据，步骤818。基于缓存替换策略，将来自缓存级的数据发出到例如DRAM的系统存储器，步骤820。基于读取请求提供来自系统存储器的数据，步骤822。

[0119] 参见图9描述了关于配置状态寄存器读取操作的操作实施图的进一步细节。该处理由处理器执行。参见图9，在一个示例中，由读取操作指定的寄存器编号被转换为对应的存储器地址，步骤900。获得用于指示加载队列中用于跟踪加载请求的位置的加载序列号和跟踪依赖性的加载标记，步骤902。执行在存储队列中与被读取的配置状态寄存器对应的地址处的数据的存在的测试，步骤904(即，其是要从存储队列中的存储器读取的数据)。如果在存储队列中找到要被读取的配置状态寄存器的数据，询问906，则从存储队列获得该值，步骤908，并且处理完成。

[0120] 返回到询问906，如果在存储队列中没有找到要被读取的配置状态寄存器的数据，则进一步确定要被读取的配置状态寄存器的数据是否在第一级缓存器中找到，询问910。如果是，则从第一级缓存器获得该值，步骤912，并且处理完成。

[0121] 返回到询问910，然而，如果在第一级缓存器中没有找到该数据，则进一步确定要被读取的配置状态寄存器的数据是否在一个或多个下一级缓存器中找到，询问914。如果在一个或多个下一级缓存器中找到该数据，则从下一级缓存器获得数据，步骤916，并且处理完成。

[0122] 返回到询问914，如果该数据不在一个或多个下一级缓存器中，则向加载队列发出读取请求以从系统存储器取得数据，步骤918。当来自系统存储器的加载完成时，获得对应于配置状态寄存器的数据，步骤920。

[0123] 根据本发明的一个方面，执行存储器内的存储器单元(例如，页)的分配以便提供软件兼容性。例如，由固件执行分配以使处理器能够执行传统的管理程序，并且使管理程序能够执行传统的操作系统，等等。

[0124] 在一个示例中，在系统的初始启动时，固件在固件拥有的存储器中分配用于存储器内配置状态寄存器的存储器内页。在一个示例中，如果管理程序不知道存储器内配置状态寄存器，则固件拥有的页在系统的整个执行中被使用，而无需对基址寄存器(例如TCBR等)的任何另外的软件引用。

[0125] 这样，管理程序将通过使用例如从配置状态寄存器移动(例如mfspr)读取上下文，并用例如移动到配置状态寄存器(例如mtspr)来重新加载上下文，来简单地执行上下文切换。这在计算机系统内提供了显著的设计简单性和性能优势。

[0126] 此外，在一个示例中，当管理程序知道存储器备份页时，其可以将每个新分区配置为具有一组备份页。此外，如果操作系统不知道存储器内配置状态寄存器，则管理程序所拥有的页在系统的整个执行中被使用，而无需对例如基址寄存器(例如TCBR等)的任何另外的软件引用。如果管理程序也不知道，则操作系统将使用固件拥有的页。

[0127] 这样，操作系统将通过使用例如从配置状态寄存器的移动(诸如，例如mfspr)来读

取上下文,并且例如用到配置状态寄存器的移动(诸如,例如mtspr)来重新加载上下文,来简单地执行上下文切换。这提供了显著的设计简单性和性能优势,便于计算机系统内的处理。

[0128] 如本文描述,根据一个或多个方面,将所选择的配置状态寄存器存储在系统存储器中。因此,通过指令解码逻辑由加载和存储指令来代替移动到配置状态寄存器和从配置状态寄存器移动。将所生成的加载和存储提交到存储队列,并且执行正常的加载和存储处理。在一个示例中,不是持续需要的配置状态寄存器(例如,除了寄存器外的那些,诸如程序计数器(PC)、数据和地址断点寄存器、PSW、浮点控制等)是存储在存储器中的寄存器。

[0129] 作为示例,存储区域由操作系统和管理程序定义,并且留出用于存储基于存储器的寄存器。在一个实施例中,在架构上指定物理存储器区域(例如,物理存储器的第一或最后n个页)。

[0130] 在至少一个实施例中,存储器内配置状态寄存器被映射到正常可缓存存储器。当配置状态寄存器要被更新时,其被存储到存储队列中。存储队列不仅是排队机制,而且有效地提供了重命名存储位置的方式,以便能够推测性地执行存储器访问。地址的推测值的多个版本可以在存储队列中(除了在缓存器或系统存储器中的架构顺序点处的认证的架构值之外)。在缓存条目已被分配后,缓存条目可被无序地更新。同样,可以通过从存储队列中清除条目来撤销存储。

[0131] 因此,可以通过使用存储队列来更新存储器内配置状态寄存器,并且无序地读回而没有性能成本,其中,由于实现用于处理器内配置状态寄存器的推测性执行的装置通常是过于昂贵的,所以基于内核锁存的配置状态寄存器迫使产生两种串行化和有序访问成本代价。

[0132] 此外,当值不在存储队列中时,对值的读取可以比从锁存器更有效地进行,这是因为频繁使用的存储器控制(例如,存储器内配置状态寄存器)将在缓存器中找到并且可以在少至2-3个周期(访问第一级缓存器的时间)内可用,因此比访问处理器中的基于锁存器的配置状态寄存器所需的专用逻辑快得多。

[0133] 在一个实施例中,当页被分配来保持配置状态寄存器时,该架构不允许使用存储器操作数来访问该页。这避免了在存储器操作与从配置状态寄存器移动指令/移动到配置状态寄存器指令之间的互锁。

[0134] 根据本发明的另一方面,配置状态寄存器的一个或多个部分被提供在存储器内,而配置状态寄存器的一个或多个其它部分被提供在处理器内。例如,配置状态寄存器可以具有多个部分(例如,字段),并且这些部分中例如被频繁访问的一个或多个部分可以保留在处理器内,而例如不频繁使用的其它部分可以被移动到存储器。这将参照图10-14进一步详细描述。

[0135] 首先,参照图10,处理器的解码单元(或另一组件)接收指令,步骤1000。由解码单元(或另一组件)确定该指令是否是移动到配置状态寄存器指令(mtcsr),诸如移动到SPR(mtspr)指令,询问1002。如果该指令是移动到配置状态寄存器指令,则处理该指令,步骤1004,如下所述。

[0136] 返回到询问1002,如果指令不是移动到配置状态寄存器指令,则进一步确定指令是否是从配置状态寄存器移动指令(mfsr),例如从SPR移动(mfspr)指令,询问1006。如果指

令是从配置状态寄存器移动指令，则处理该指令，如下所述，步骤1008。返回到询问1006，如果指令既不是移动到配置状态寄存器指令也不是从配置状态寄存器移动指令，则执行常规指令解码，步骤1010。

[0137] 在另一实施例中，可以进行关于指令是否是使用配置状态寄存器的另一指令的其它询问，并且如果是，则可以适当地处理该指令，本文描述了其示例。在又一实施例中，可以类似地执行不对应于指令的处理器操作(例如，启动异常处理序列)。

[0138] 参见图11描述了关于处理移动到配置状态寄存器指令的进一步细节。在一个示例中，配置状态寄存器可以是专用寄存器(SPR)，而指令是移动到SPR(mtspr)指令。然而，这仅是一个示例。其它配置状态寄存器可类似地被处理。该逻辑由处理器执行，诸如，例如处理器的解码单元。在其它示例中，一个或多个其它组件执行该逻辑。

[0139] 参见图11，基于获得(例如，接收、提供、选择等)移动到配置状态寄存器指令(诸如mtspr指令)，确定由指令指定的配置状态寄存器(CSR)的至少一部分是否是在存储器内，询问1100。如果不是，则执行移动到配置状态寄存器指令(例如mtspr)的常规处理，步骤1102。

[0140] 返回到询问1100，如果配置状态寄存器的至少一部分在存储器内，则进一步确定整个配置状态寄存器是否是在存储器内，询问1104。如果整个配置状态寄存器在存储器内，则生成存储配置状态寄存器内部操作，步骤1106。参见图5描述了与该内部操作相关联的处理的示例。

[0141] 返回到询问1104，如果配置状态寄存器的仅一个或多个部分在存储器内，则针对一个或多个存储器内配置状态寄存器部分生成一个或多个存储配置状态寄存器操作，步骤1110。此外，针对一个或多个核内配置状态寄存器部分生成更新的内部操作，在步骤1112。更新的内部操作可以是一个或多个指令、状态机或执行将一个或多个通用寄存器的内容(包括所指定的核内部分的数据)复制到核内配置状态寄存器的适当部分的操作的其它设备。处理完成。

[0142] 参见图12描述了关于与处理从配置状态寄存器移动指令相关联的处理的进一步细节。在一个示例中，配置状态寄存器可以是专用寄存器(SPR)，而指令是从SPR移动(mfspr)指令。然而，这仅是一个示例。其它配置状态寄存器可被类似地处理。该逻辑由处理器执行，诸如，例如处理器的解码单元。在其它示例中，一个或多个其它组件执行该逻辑。

[0143] 参见图12，基于获得(例如，接收、提供、选择等)从配置状态寄存器移动指令，例如mfspr指令，确定配置状态寄存器的至少一部分是否是在存储器内。如果不是，则执行用于从配置状态寄存器移动指令的常规处理，步骤1202。

[0144] 返回到询问1200，如果配置状态寄存器的至少一部分在存储器内，则进一步确定整个配置状态寄存器是否是在存储器内，询问1204。如果整个配置状态寄存器在存储器内，则生成加载配置状态寄存器内部操作，步骤1206。参见图4描述了与该操作相关联的处理的示例。

[0145] 返回到询问1204，如果配置状态寄存器的仅一个或多个部分在存储器内，则针对一个或多个存储器内配置状态寄存器部分生成一个或多个加载配置状态寄存器内部操作，步骤1210。此外，针对一个或多个核内配置状态寄存器部分生成一个或多个读取内部操作，步骤1212。

[0146] 另外，在一个实施例中，生成一个或多个内部操作以将存储器内部分和核内部分

组合成在架构上限定的配置状态寄存器图像，步骤1214。这可以包括使用例如Insert Under Mask(插入下掩码)指令、或OR、AND、和/或NOT逻辑电路，如下面进一步描述的。

[0147] 参见图13A描述了关于复合(composite)配置状态寄存器的使用的进一步细节，其中一个或多个部分在处理器内而一个或多个部分在存储器内，其中描述了读取引用。该逻辑由处理器执行，诸如，例如处理器的解码单元。在其它示例中，一个或多个其它组件执行该逻辑。

[0148] 参见图13A，基于复合配置状态寄存器读取引用1300，确定被访问的特定部分(也称为组件，例如，字段)是在存储器内还是在处理器内，询问1310。如果特定部分在处理器内，则访问处理器内组件，步骤1320，并且处理继续到询问1350，如下所述。然而，如果特定组件在存储器内，询问1310，则执行处理，如参见图6所描述的。例如，获得存储器基地址(基址)，步骤1330，以及获得在指令中指示的引用复合配置状态寄存器的寄存器编号，步骤1332。将寄存器编号映射到偏移，步骤1334，并且执行从地址(基址+偏移)到临时寄存器的加载，步骤1336。然后使用临时寄存器，步骤1338。此后，或在步骤1320之后，确定是否要访问复合配置寄存器的另一组件，询问1350。如果是，则处理继续到询问1310。否则，处理完成。

[0149] 参见图13B描述了关于复合配置状态寄存器的使用的进一步细节，其中一个或多个部分在处理器内而一个或多个部分在存储器内，其中描述了写入引用。该逻辑由处理器执行，诸如，例如处理器的解码单元。在其它示例中，一个或多个其它组件执行该逻辑。

[0150] 参见图13B，基于复合配置状态寄存器写入引入(reference)1360，确定被访问的特定部分(也称为组件，例如，字段)是在存储器内还是在处理器内，询问1370。如果特定部分在处理器内，则访问处理器内组件，步骤1390，并且处理继续到询问1388，如下所述。然而，如果特定组件在存储器内，询问1370，则执行处理，如参见图7所描述的。例如，获得存储器基地址(基址)，步骤1380，以及获得在指令中指示的引用复合配置状态寄存器的寄存器编号，步骤1382。将寄存器编号映射到偏移，步骤1384，并且对由基址+偏移限定的地址执行存储，步骤1386。此后，或在步骤1390之后，确定复合配置寄存器的另一组件是否要被访问，询问1388。如果是，则处理继续到询问1370。否则，处理完成。

[0151] 图14中描绘了复合配置状态寄存器的一个示例。如图所示，在该示例中，复合配置状态寄存器1400是专用寄存器(SPR)1，其对应于整数异常寄存器(XER)。该寄存器包括多个字段1402。在一个示例中，一个或多个字段是处理器内字段1404，而另一个字段1406是存储器内字段。在此特定示例中，Xerf0、1、2(即XER的字段0、1及2)在处理器内被重命名为SO(summary overflow(摘要溢出))、OV(overflow(溢出))和CA(进位)，而Xerf3(XER的字段3)(在此示例中未被重命名)是存储器内的字节计数字段。利用这种配置，可以生成以下IOP序列，并将其用于针对复合配置状态寄存器分别执行mtspr和mfspr：

[0152] mtspr_xer mtxerf2

[0153] mtxerf0

[0154] mtxerf1

[0155] stxerf3

[0156] 如上所述，用于XER寄存器的mtspr包括：移动到XER的字段2(mtxerf2)，其中通用寄存器的内容被复制到XER字段2；移动到XER的字段0(mtxerf0)，其中通用寄存器的内容被

复制到XER字段0;以及移动到XER的字段1(mtxerf1),其中通用寄存器的内容被复制到XER字段1。用于XER寄存器的mtspr还包括由存储操作执行的存储到XER的字段3(stxerf3),这是由于字段3是在存储器内。

[0157] mfspr_xer mfxerf2

[0158] mfxerf0

[0159] or

[0160] ldxerf3

[0161] or

[0162] mfxerf1

[0163] or

[0164] 对于从XER的移动,每个字段从处理器内或从存储器内被读取,并且这些字段通过例如OR运算来组合。例如,读取字段2和字段0的内容,并且执行OR运算以提供结果1;然后,读取字段3的内容(例如,使用加载,诸如,例如加载xref3内部操作,Idxref3,因为字段3在存储器内)并与结果1进行OR运算以产生结果2。此外,读取字段1的内容,并与结果2进行OR运算,以提供最终结果,该最终结果是XER与其字段的图像,而不管其是在处理器内还是在存储器内。

[0165] 如本文所述,根据本发明的一个方面,从配置状态寄存器移动指令产生一系列从处理器内部分的移动,并对存储在存储器内的该部分进行读取。所读取的存储器内和处理器内部分的内容例如使用例如一系列OR指令的来整理。此外,移动到配置状态寄存器指令产生一系列到处理器内部分的移动,并对存储在存储器内的部分进行存储。

[0166] 在一个方面,当存储器被分配给配置状态寄存器时,偏移在架构上(例如,被限定的并且外部可见的)或在微架构上(被限定的但外部不可见的)被指定。例如,偏移可以直接从配置状态寄存器编号(或其它指示)得到。

[0167] 作为一个映射示例,每个配置状态寄存器被映射到对应的偏移处(双字形式),即基址*配置状态寄存器#,其中配置状态寄存器1在第一位置处,配置状态寄存器2在第一位置加上限定的字节数(例如,8)处,等等。

[0168] 然而,配置状态寄存器编号是不连续的,浪费了存储器和缓存效率。因此,在根据本发明的一个方面的另一实施例中,配置状态寄存器编号不用于直接得出到存储器页的偏移,而是基于功能亲和度为配置状态寄存器分配偏移。因此,在共同操作中一起使用的配置状态寄存器被分配给相同或相邻的缓存线,以增强缓存局部性。例如,EBB处理使用以下寄存器:例如EBBHR、EBBRR、BESCR和TAR。TAR与其它TAR不相邻。然而,它们都将被分配给存储器,使得它们在相同的缓存线或相邻的缓存线中结束。

[0169] 图15A-15B中描述了线性映射的一个示例。如图所示,在一个示例中,线性映射1500是稀疏的。例如,在一个示例中,使用8KB(2页),即使少于1K的配置状态寄存器被映射。此外,联合使用的配置状态寄存器,诸如EBBHR、EBBRR、BESCR和TAR,是不连续的。另外,一组配置状态寄存器不在对齐边界上以确保它们在同一缓存线中(例如,779MMCR0、780SIAR、781SDAR、782MMCR1)。此外,一些配置状态寄存器可以指代同一寄存器;例如不同的访问许可、子字段等。这是缓存器的低效使用。缺乏预提取(以确保每个活动仅遭受一个缓存未命中);以及过大的缓存足迹(导致工作集增加,这降低了命中率)。因此,根据本发明的一个方

面,配置状态寄存器不被存储在例如基址+(idx*8)处。相反,配置状态寄存器被存储在例如基址+remap[idx]处。

[0170] 这种重新映射确保一组是相邻的,以便共享缓存线;其消除/减少了稀疏性,提供了更有效的缓存使用;并且处理多个名字。作为一个示例,重新映射是静态的,并且在处理器设计时执行,并且在诸如表的数据结构中提供,或者通过计算定义等式来提供。作为另一示例,重新映射是动态的并且通过使用来确定。例如,如果寄存器的跟踪显示一起使用一组寄存器中的寄存器,则这些寄存器被分组并彼此相邻步置。还存在其它可能。

[0171] 参照图16描述了重新映射的进一步细节。该处理由处理器执行。在一个示例中,由处理器获得配置状态寄存器编号,步骤1600。基于配置状态寄存器编号,确定到存储器单元(例如,页)中的索引位置(也称为偏移),步骤1602。这可以通过查表找或通过计算来确定。索引位置被返回给请求者(例如,内部操作),步骤1604。

[0172] 在进一步的示例中,使用包括映射逻辑的电路。配置状态寄存器编号被输入到映射逻辑,并且输出是页索引。

[0173] 如上所述,在一个实施例中,如在架构中定义的配置状态寄存器编号被重新映射,使得一起使用的那些配置状态寄存器被彼此靠近地放置,以便提供更高效的缓存。这减少了用于配置状态寄存器的缓存线的数量,并且与其它程序竞争较少地使用缓存器。此外,其确保在缓存线加载了特定配置状态寄存器并且为该值的缓存未命中付出代价后,其它配置状态寄存器(其也可以结合该配置状态寄存器使用)将在缓存器中命中,并且因此不涉及其它缓存未命中代价。

[0174] 在另一实施例中,当不同的偏移分配是可能的情况下,sync_o_csr指令将版本戳写入到页中。当在不同主机之间迁移分区时,版本戳可用来调整偏移;和/或直接在硬件中调整偏移(例如,当sync_i_csr可以读取分区的偏移版本号时)或在软件中调整偏移。

[0175] 根据本发明的另一方面,提供了用于执行批量操作以存储或加载多个配置状态寄存器的能力。配置状态寄存器的单独加载或存储是昂贵的,因为每个读取或写入要按顺序执行,并且要在下一指令开始之前完成。此外,由于缺乏对处理器内配置状态寄存器的重命名而不允许回滚,因此要确保正确的异常/错误排序。然而,根据本发明的一个或多个方面,提供了批量配置状态寄存器加载到存储器单元(诸如页)以及从存储器单元存储。

[0176] 例如,存储配置状态寄存器指令或操作(例如,ST_CSR)用于将多个处理器内配置状态寄存器存储在存储器内(即,将与当前上下文(例如应用、线程等)相关联的配置状态寄存器的内容存储在针对特定配置状态寄存器限定的选择存储器位置中);以及加载配置状态寄存器指令或操作(例如,LD_CSR)用于将存储在存储器内的配置状态寄存器加载回处理器内(即,将与当前上下文相关联的配置状态寄存器的内容从存储器加载回处理器中)。

[0177] 处理器内(也称为非存储器后备(backed))配置状态寄存器(其可存储在存储器中)被分配存储器中的位置,例如,在后备存储器单元(例如,页)中。在一个示例中,存储器位置和/或单元是明确定义的预定义位置/单元,并且因此,指令不需要具有以指定位置/单元的操作数。在另一实施例中,特定位置/单元可被指定为指令的操作数。此外,在另一实施例中,每个指令可包括以指示要被存储/加载的特定寄存器的操作数。其它变化也是可能的。

[0178] 此外,页仅是存储器单元的一个示例。其它单元也是可能的。而且,尽管页通常是

4KB,但在其它实施例中,其可以是其它大小。存在许多可能性。

[0179] 除了上述ST_CSR和LD_CSR指令之外,根据本发明的一个方面,可使用的另一指令是mtspr TCBR,next_u->csr_page。该指令被用于加载存储器区域的基地址,该存储器区域的基地址被用于存储存储器内配置状态寄存器,以用于寄存器(诸如TCBR)中的特定上下文(例如,处理器、线程等)。然后,如本文描述,在采用基地址的处理中使用该地址。在该指令中,next_u->csr_page是指存储发出指令的上下文的数据的用户数据结构。该数据包括存储存储器内配置状态寄存器的存储器单元(例如,页)的基地址。尽管mtspr被指定,但是也可以使用其它移动到配置状态寄存器(mtspr)指令。此外,TCBR仅仅是基址寄存器的一个示例。也可以指定其它基址寄存器。许多变化是可能的。

[0180] 除了上述指令之外,可以提供两个同步指令以使缓存器与存储器或处理器内寄存器同步。例如,sync_o_csr用于同步缓存器和一个或多个存储器内配置状态寄存器,而sync_i_csr用于同步缓存器和一个或多个处理器内配置状态寄存器。

[0181] 如本文描述,在一个方面中,多个配置状态寄存器被加载或存储。不存在介入异常;即避免了异常;该操作在该操作的所有配置状态寄存器上完成或者不完成。此外,不存在页错误(例如,页被固定;而且在页已被加载后,保证对同一页的引用)。如果需要,硬件可以使进程可重新启动;例如加载或存储微代码或状态机中的序列。

[0182] 如参见图17A-17C所描述的,通过使用以存储/加载多个配置状态寄存器的指令,某些昂贵操作(诸如,确保顺序点和完整指令顺序)被较不频繁地使用。图17A-17C的处理由处理器执行。

[0183] 参见图17A,参见图17A描述了使用单独指令(即,不是批量操作)将多个配置状态寄存器存储到存储器的一个示例。在一个示例中,为了将配置状态寄存器移动到存储器,使用从配置状态寄存器移动指令和存储指令。例如,在一个示例中,配置状态寄存器是SPR,而从配置状态寄存器移动指令是mfspr。可以使用其它配置状态寄存器和对应的指令。

[0184] 在该示例中,基于mfspr指令(例如从SPR移动(mfspr)指令)的执行,来确保顺序点,步骤1700。然后,由指令指定的配置状态寄存器(例如,SPR)的内容从配置状态寄存器复制到例如通用寄存器(GPR)中,步骤1702。指令按顺序完成,步骤1704。此后,经由存储指令(STD),将通用寄存器的内容存储到存储器,步骤1706。该相同的过程针对要被存储在存储器中的每个配置状态寄存器而被重复(例如,步骤1708-1722),其可以是许多配置状态寄存器。因此,对于要被存储到存储器的每个配置状态寄存器,需要确保顺序点并完成指令顺序操作。

[0185] 然而,根据本发明的一个方面,提供了存储配置状态寄存器(ST_CSR)指令,其使用单次确保顺序点和单次完成指令顺序操作来将多个配置状态寄存器存储在存储器中,如参见图17B所述。

[0186] 参见图17B,基于存储配置状态寄存器指令(ST_CSR)的执行,到达顺序点,步骤1750。然后,将所选择的配置状态寄存器的内容加载到临时寄存器中,步骤1752。此外,临时寄存器的内容然后被存储到存储器(例如,存储器控制页),步骤1754。针对一个或多个附加配置状态寄存器,加载/存储操作被重复一次或多次1756-1762。在将所选择的配置状态寄存器复制到存储器(其可以是许多此类寄存器)后,按顺序完成指令,步骤1770。

[0187] 在一个示例中,ST_CSR指令不具有以指定要被复制的寄存器的操作数;相反,当前

上下文(例如,进程、线程等)的所有处理器内配置状态寄存器被复制。在另一示例中,操作数可以被包括并且用于指定要被复制到存储器的一个或多个配置状态寄存器。其它变化也是可能的。

[0188] 在进一步的示例中,可以使用批量加载操作(例如,LD_CSR)将多个配置状态寄存器从存储器复制到处理器内。

[0189] 参见图17C,基于加载配置状态寄存器(LD_CSR)指令的执行,确保顺序点,步骤1780。然后,从存储器获得所选择的配置状态寄存器的内容并将其加载到临时寄存器中,步骤1782。然后,临时寄存器的内容被存储在对应的处理器内配置状态寄存器中,步骤1784。对于一个或多个(并且可能许多)附加配置状态寄存器,加载/存储操作被重复一次或多次1786-1792。此后,按顺序完成指令,步骤1794。

[0190] 在一个示例中,LD_CSR指令不具有以指定要被复制的寄存器的操作数;相反,当前上下文的所有配置状态寄存器(例如,进程、线程等)被复制。在另一示例中,操作数可以被包括并且用于指定要从存储器复制的一个或多个配置状态寄存器。

[0191] 在一个方面中,用以执行批量操作的指令对一组配置状态寄存器执行相同的操作(例如,存储、加载等),其中该组配置状态寄存器由共同特性限定。共同特性可以是:例如寄存器的数值范围;具有相同的访问许可或上下文(例如,用户、操作系统、管理程序);具有相同的功能目的(例如,异常处理、计时等);或者具有相同的实现属性(例如,一组配置状态寄存器被存储在存储器内),作为示例。

[0192] 批量存储和/或加载操作的使用改进了计算机内的处理。例如,通过使用批量存储操作来将多个配置状态寄存器有效地复制到存储器,可以更快且更有效地执行上下文切换处理。也可以实现其它益处。

[0193] 在另一方面中,为了便于处理,提供了架构布置控制以指示配置状态寄存器被存储在存储器中的位置。例如,硬件定义了一组控制来识别配置状态寄存器被存储在存储器中的位置。作为示例,提供至少一个配置状态寄存器来指定用于存储应用状态的基地址。作为一个示例,基地址是客机物理地址,即客机操作系统指定其自己的地址空间中的地址。例如,当地址被指定时,指定客机级地址(例如,客机实、客机物理或客机虚拟地址),因为允许客机指定主机物理地址可能损害虚拟化和安全性。

[0194] 参见图18A描述了关于指定架构配置控制的进一步细节。在一个示例中,处理器执行该逻辑。首先,接收指示关于当前执行环境(例如,应用状态、线程状态、操作系统状态、管理程序状态、特定的客机或主机操作系统级等)的存储器后备位置的地址(即,基地址),步骤1800。例如,管理程序使用管理程序地址(例如,主机虚拟或绝对、物理、实地址),并且操作系统使用关于虚拟机/逻辑分区的客机实地址或客机虚拟地址。处理器获得指示存储器页的位置以存储配置状态寄存器的地址(即,基地址)。

[0195] 可选地,该基地址被转换为物理地址,步骤1802。(转换可能已经被执行或者地址不需要被转换。)在一个实施例中,该转换可能导致页错误。在进一步的实施例中,协作的硬件和软件被用于避免页错误,例如,通过在执行本技术的一个或多个方面之前固定页。

[0196] 另外,在一个示例中,采集被转换的地址,步骤1804。即,在一个示例中,对被转换的地址进行缓存。在另一实施例中,存储未转换的地址和被转换的地址两者。

[0197] 在另一实施例中,参见图18B,获得关于被引用的存储器控制(例如,配置状态寄存

器)的采集地址,步骤1850。另外,在步骤1852,地址可被转换为物理地址,并且在步骤1854,访问存储器内控制(例如,配置状态寄存器)。该转换可能导致页错误,但是使用协作的硬件和软件,可以通过例如在执行本技术的一个或多个方面之前对页进行固定来避免页错误。

[0198] 在一个实施例中,基地址被存储在配置状态寄存器中,例如,不在存储器中,以避免循环依赖性。其它示例是可能的。

[0199] 此外,在一个实施例中,基地址被转换成物理/实地址;在另一个实施例中,基地址被转换成下一级的监管(supervisor)地址(即,当操作系统设置页地址时,其被转换成监管地址)。其它示例是可能的。

[0200] 作为一个示例,未转换的和被转换的(物理/实)基地址两者被存储。这消除了对每个配置状态寄存器访问执行地址转换(例如,动态地址转换(DAT))并处理页错误的需要。

[0201] 在一个实施例中,在处理器寄存器中维持被转换的(实/物理)基地址,并且在存储器内配置状态寄存器中维持未转换的基地址。在该实施例中,响应于以再次读出配置状态寄存器的基地址的软件请求而提供未转换的地址。已转换的地址可用于从其存储器内位置访问此地址。存在其它可能。

[0202] 如本文描述,提供了诸如配置状态寄存器的控制,其包括指定一个或多个存储器内配置状态寄存器被存储在存储器中的位置的基地址。这些存储器内配置状态寄存器是在架构上被限定为处理器内寄存器的寄存器,但是根据本发明的一个或多个方面,这些寄存器已经被转换成存储器内配置状态寄存器。这些存储器内配置状态寄存器不同于被存储在存储器中的配置值,因为至少这些值不是寄存器,并且其在架构上被定义为在存储器内。它们不是结构上定义的处理器寄存器。

[0203] 在另一方面,在程序环境和例如操作系统或其它监管环境之间和/或在不同应用或线程之间等的上下文切换期间实现了效率。当要执行上下文切换时,保存先前上下文的数据。在一个示例中,为了保存上下文数据,从处理器内寄存器中读出配置状态寄存器的内容并将其保存到存储器。然后,加载下一上下文的数据,其包括加载配置状态寄存器以恢复执行。这是昂贵的过程。

[0204] 根据本发明的一个方面,即使采用存储器内配置状态寄存器执行,其中基于存储队列的推测和无序执行加速了该过程,但仍然存在与保存和恢复上下文相关联的显著成本。

[0205] 因此,根据本发明的一个方面,上下文切换是通过配置状态寄存器页操纵来执行的。在一个示例中,配置状态寄存器在存储器内的存储的位置是可配置的。当切换上下文时,不是将旧的配置状态寄存器数据复制出配置状态寄存器存储器单元(例如,页)并将新数据复制到配置状态寄存器页中,而是选择不同的配置状态寄存器存储器单元(例如,页),从而改变处理器所看到的配置状态寄存器的值。

[0206] 根据本发明的一个方面,通过修改例如包括在基配置状态寄存器(本文称为基址寄存器)中的基指针或基地址(本文称为基址)来执行上下文切换,该基配置状态寄存器指示一个或多个配置状态寄存器(本文称为CSR后备存储器)在存储器中的位置,以避免卸载和重新加载配置状态寄存器的需要。

[0207] 可存在可以从该方面受益的若干类型的上下文切换,包括操作系统上下文切换,其中操作系统执行在不同应用之间的切换;管理程序上下文切换,其中管理程序或虚拟机

监测器在不同分区或虚拟机之间切换;以及在不同硬件线程之间的硬件线程上下文切换。每个上下文切换影响不同的寄存器。例如,当切换出应用作为使用操作系统的上下文切换的一部分时,对应于该应用的几个配置状态寄存器被改变到另一位置,而不是其它配置状态寄存器(例如,不是操作系统配置状态寄存器)。此外,通过管理程序上下文切换,可存在更多的寄存器要被切换出。类似地,用于硬件线程上下文切换。下面描述关于硬件线程上下文切换的进一步细节。

[0208] 在一个实施例中,对于硬件线程上下文切换,处理器使用线程调度器来从加载到处理器中的多个线程中进行选择。然而,根据本发明的一个方面,硬件可以从可由硬件调度的多个线程中进行选择,其中该多个线程超过了处理器中加载的硬件线程上下文的数量。也就是说,根据本发明的一个方面,如本文所述的,上下文切换的能力允许硬件使用比处理器中加载的线程更多的线程。选择线程,并且硬件通过选择该线程的存储器内配置信息来调度该线程。在一个实施例中,一些可执行寄存器被存储在片上寄存器文件或快速第二级存储器中。在另一实施例中,通用寄存器也被存储到存储器内配置页,并且在线程被调度时从该存储器内配置页加载。这是按需执行的(例如,当第一次访问时每个寄存器)或批量执行的(例如,在调度时间的所有寄存器)。

[0209] 在一个实施例中,硬件响应于硬件标准而调整基指针本身,而不是让软件代理(例如,操作系统或管理程序)决定将指针改变到另一配置状态寄存器基址。在硬件中选择多个线程中的一个线程,并且基于多个线程,选择指向具有配置状态寄存器的系统存储器页的指针中的一个指针。

[0210] 参见图19A和19B描述了与执行上下文切换相关的进一步细节。图19A描绘了用于执行上下文切换的一个过程,其中数据被复制;以及图19B描绘了用于执行上下文切换的另一过程,其中根据本发明的一个方面指针被修改。

[0211] 首先参见图19A,当上下文切换要被执行时,对上下文复制出(copy-out)进行初始化以复制出旧的上下文数据,步骤1900。这包括定位针对先前上下文的上下文结构(例如,监视结构)以及识别针对要被读出(read-out)的上下文的第一配置状态寄存器。

[0212] 作为示例,上下文可以是虚拟机、逻辑分区、进程、线程等中的一者。复制出过程继续。

[0213] 选择要被存储的配置状态寄存器,步骤1902。在此迭代中,其是上面识别的配置状态寄存器。读取所选择的配置状态寄存器,步骤1904,并且将配置状态寄存器的内容存储到由例如监管软件(或硬件)维护的存储器内结构,用于存储上下文切换数据,步骤1906。然后,确定是否存在更多的要进行上下文切换的配置状态寄存器,询问1908。如果是,则处理继续到步骤1902。

[0214] 在复制出针对先前上下文的数据之后,针对新的上下文执行复制入(copy-in)过程。因此,对上下文复制入进行初始化,其中定位针对下一上下文的上下文结构,并且识别针对上下文写入(write-in)的第一配置状态寄存器,步骤1910。选择要被加载到存储器内的配置状态寄存器,步骤1912,并且从上下文结构(例如,监管结构)读取所选择的配置状态寄存器的内容,步骤1914。读取的上下文数据被写入存储器内的配置状态寄存器,步骤1916。确定是否存在更多的要进行上下文切换的配置状态寄存器,询问1920。如果是,处理继续到步骤1912。否则,处理完成。

[0215] 参见图19B,描述了根据本发明的一个方面的上下文切换过程的另一个示例。在该示例中,指向数据的指针被操纵,而不是复制数据。首先,初始化上下文复制出,其中定位针对先前上下文的上下文结构(例如,监管器或硬件结构),步骤1950。同样,上下文可以是虚拟机、逻辑分区、进程、线程等中的一个。然后,处理器内配置状态寄存器,步骤1952,使得寄存器的内容被存储在存储器中。在一个实施例中,这是使用例如参照图19A描述的复制循环来实现的。在另一个实施例中,这是使用例如批量复制操作(例如,ST_CSR)来实现的。

[0216] 存储器内配置状态寄存器数据单元的地址(即,例如,用于存储针对该上下文的配置状态寄存器的存储器页的地址(基地址))可以被读取,步骤1954,并且被存储到上下文结构,步骤1956。在一个示例中,该地址不改变,因此不需要重复读取该地址并将其存储到上下文结构。相反,在第一次或当该页被移动到新位置时,存储该地址。该值被存储到由例如监管软件或硬件维护的存储器内结构,用于存储上下文切换数据。

[0217] 在执行了复制出过程之后,利用复制入过程来指向新的上下文数据。因此,对上下文复制入过程进行初始化,其中针对下一个上下文来定位上下文结构(例如,监管器或硬件结构),并且针对上下文写入来识别第一配置状态寄存器,步骤1960。加载针对下一上下文的配置状态寄存器数据单元的地址,步骤1962。即,例如,获得以存储针对新上下文的配置状态寄存器的存储器页的地址(基地址)。另外,存储器内配置状态寄存器数据页地址(基地址)被写入,步骤1964。此外,处理器内配置状态寄存器被处理,步骤1966。作为示例,例如使用复制循环或批量加载(例如,LD_CSR)从存储器加载针对该上下文的处理器内配置状态寄存器。

[0218] 如上所述,指针操纵可以用于上下文切换。这对于虚拟机迁移也是成立的。

[0219] 在一个实施例中,存储器内寄存器可用于加速虚拟机(或逻辑分区)迁移和/或实时机器迁移。在一个示例中,根据常规技术来迁移页;然而,存储器内配置状态寄存器不被移动。作为示例,ST_CSR指令或操作被用于采集处理器内配置状态寄存器,但是没有配置状态寄存器被移动。相反,存储器内配置存储器页被移动。

[0220] 在进一步的实施例中,对于实时机器迁移,当机器已经静止时,存储器内配置状态寄存器状态被移动。如果存在多个上下文(例如,多个线程/进程等),则这可以包括多个页。

[0221] 在一个实施例中,当主机和目标迁移格式(例如,当配置状态寄存器通过不同的架构实现被映射到存储器内配置寄存器页内的不同偏移处时)不兼容时,由迁移代理执行调整。在一个该实施例中,提供了配置状态寄存器格式的架构或微架构版本号,并且接收系统负责调整布局。在另一实施例中,接收和发送系统协商传送格式。在另一实施例中,传送格式被定义,例如,配置状态寄存器值的线性列表或一对<钥,值>,其中,钥是配置状态寄存器编号,并且值是配置状态寄存器的值。

[0222] 在其它实施例中,处理器可以支持多个版本的布局,以基于外部配置的配置状态寄存器布局图来适配重映射逻辑(例如,软件通过将布局标识符加载到配置状态寄存器中来指定)。

[0223] 在至少一些处理器实施例中,该设计可以将一些值缓存在被识别为存储器内寄存器的寄存器的定制处理器内位置。因此,当执行上下文切换时,被缓存的副本将被同步,即,使旧的缓存的值无效。在一个示例中,提供上下文同步指令(csync),其向处理器指示使旧的缓存的值无效。例如,每当执行上下文切换时,执行该操作。在一个示例中,使对应于上下

文的所有配置状态寄存器的缓存的值无效。在其它示例中，使针对特定寄存器的缓存的值无效。

[0224] 如本文描述，在上下文切换时，可指示新页，而不是复制配置状态寄存器。以此方式，配置状态寄存器上下文图像已经被保存，并且至少对于存储器内维护的配置状态寄存器减少了上下文切换。

[0225] 此外，在一个实施例中，启用了具有页替换的上下文切换。存储器图像(至少对于应当存在于存储器中的那些寄存器)在加载新的存储器上下文页之前被同步。

[0226] 根据一个实施例，提供了配置状态寄存器同步指令。具体地，这可以清除任何缓存的值，并且在已经执行了切换之后抑制缓存，并且相反地可以指示其可以再次将值保存到缓存器。示例指令包括：

[0227] 同步输出CSR：sync_o_CSR

[0228] 加载新的后备页 mtspr TCBR,next_u->spr_page

[0229] 同步输入SPR sync_i_CSR

[0230] 在另一实施例中，到基址寄存器(例如TCBR)的移动到配置状态寄存器指令(例如mtspr指令)自动执行缓存的输出值和输入值的同步。

[0231] 在另一方面，基于加载客机基地址，执行从客机基地址到对应的主机基地址的地址转换，以便避免页错误的可能，其中客机基地址指示存储器中用于存储一个或多个存储器内配置状态寄存器的位置并且可以存储在基配置状态寄存器中。该转换例如立即(即基于接收基地址并在使用基地址之前，例如在存储引用期间)被执行。例如，当虚拟地址被加载到基址寄存器中(例如，加载到基址，诸如将客机基地址加载在要被用作基地址的配置状态基址寄存器(例如，TCBR)中)时，系统自动执行到物理存储器地址的地址转换，并且采集被转换的物理地址连同虚拟地址。根据一个架构实施例，加载到基址寄存器被识别为导致或可能导致转换将被执行的指令。

[0232] 当地址不能被转换时，发生转换错误。页将是可访问的，以便根据例如页表条目(下面描述)中的指定许可来读取和写入，或者引发页错误。

[0233] 参见图20描述了关于基于导致或可能导致基地址的转换将被执行的操作的执行而自动执行地址转换的进一步细节。该处理由处理器执行，并且在一个示例中，该处理是基于移动到配置状态寄存器指令(例如，mtspr指令)到配置状态寄存器存储器基址寄存器的执行来执行的。该处理也可以基于其它指令的执行来执行。

[0234] 基于执行指令，接收指示关于当前执行环境的存储器后备位置的地址(例如，基地址)，步骤2000。根据mtspr的一个定义，mtspr被定义为可能引起动态地址转换(DAT)页错误。因此，根据本发明的一个方面，地址转换作为mtspr的一部分而自动执行，甚至在确定需要转换之前进行。

[0235] 使用例如DAT表将所接收的基地址转换成物理基地址，下面进一步描述其一个示例，步骤2002。确定是否发生了DAT转换错误，询问2004。如果DAT转换错误已经发生，则指示DAT页错误，步骤2006。进入页错误处理程序软件例程，步骤2008，并且执行页错误处理程序，步骤2010。在一个示例中，根据页错误处理程序，如果指令是可允许的错误(例如，页调出)，则该指令被重启。否则，上下文或执行环境(例如，操作系统、管理程序、虚拟机、线程、进程等)接收可能导致上下文终止的错误指示。

[0236] 返回到询问2004,如果不存在DAT转换错误,则采集未转换的基地址(例如,存储在寄存器中),步骤2012。此外,采集(例如,缓存)被转换的基地址,步骤2014。可选地,对象页被固定,步骤2016。

[0237] 参见图21A-21B描述了关于动态地址转换的一个示例的进一步细节。该处理由处理器执行。

[0238] 动态地址转换是将虚拟地址转换成对应的实(或绝对)地址的过程。动态地址转换可以针对由CPU生成的指令和数据地址来指定。虚拟地址可以是主虚拟地址、次虚拟地址、AR(访问寄存器)指定的虚拟地址或归属虚拟地址。分别借助于主、次、AR指定的或归属地址空间控制元素(ASCE)来转换地址。在选择适当的地址空间控制元素之后,转换过程对于所有四种类型的虚拟地址都是相同的。地址空间控制元素可以是段表指定或区域表指定。段表指定或区域表指定使得转换将借助于由操作系统在实或绝对存储中建立的表来执行。

[0239] 在使用段表指定或区域表指定时的转换过程中,识别三种类型的信息单元,包括区域、段和页。因此,虚拟地址被分成四个字段。在一个示例中,对于64位地址,位0-32被称为区域索引(RX),位33-43被称为段索引(SX),位44-51被称为页索引(PX),并且位52-63被称为字节索引(BX)。虚拟地址的RX部分本身被分成三个字段。在一个实施例中,位0-10被称为区域第一索引(RFX),位11-21被称为区域第二索引(RSX),而位22-32被称为区域第三索引(RTX)。

[0240] 参考图21A描述了将虚拟地址转换为实地址的一个示例。该过程在这里被称为DAT走(walk)(或页走),其中,地址转换表被走以将一个地址(例如,虚拟地址)转换为另一地址(例如,实地址)。在该示例中,地址空间控制元素(ASCE)2100包括表原点2102,以及指定类型(DT)控制2104,其是转换的开始级别的指示(即,在层级地址转换中的哪个级别开始的指示)。使用表原点2102和DT2104,定位特定表的原点。然后,基于该表,虚拟地址的位被用于索引到特定表中以获得下一级表的原点。例如,如果选择了区域第一表(RFT)2106,则虚拟地址的位0-10(RFX)2108被用于索引到区域第一表中以获得区域第二表2110的原点。然后,虚拟地址的位11-21(RSX)2112被用于索引到区域第二表(RST)2110中以获得区域第三表2114的原点。类似地,虚拟地址的位22-32(RTX)2116被用于索引到区域第三表(RTT)2114中以获得段表2118的原点。然后,虚拟地址的位33-43(SX)2120被用于索引到段表2118中以获得页表2122的原点,并且虚拟地址的位44-51(PX)2124被用于索引到页表2122中以获得具有页帧实地址(PFRA)2126的页表条目(PTE)2125。然后,将页帧实地址与偏移2128(位52-63)组合(例如,连接)以获得实地址。在一个实施例中,然后可以应用加前缀以获得相应的绝对地址。

[0241] 参考图21B描述了地址转换的另一示例。在该示例中,执行DAT走以将初始客户机虚拟地址转换成最终主机实地址。在这个示例中,地址空间控制元素(ASCE)2100是客机地址空间控制元素,并且ASCE 2100的DT 2104指示由客户机地址转换结构2160确定的客户机转换将在由表原点2102指向的区域第一表2106处开始。因此,初始客户机虚拟地址(例如,RFX 2108)的适当位被用于索引到区域第一表2106中以获得区域第一表的条目的指针。区域第一表项(RFTE)的地址是客机实地址或绝对地址。在应用了主存储原点和限制的情况下,在适当时,该客机实或绝对地址对应于主机虚拟地址。然后,使用主机地址转换结构2170来转换该中间主机虚拟地址。具体地,地址空间控制元素(ASCE)2150是用于指示主机

地址转换结构2172中的转换的开始级的主机地址空间控制元素。基于由DT 2154指示的开始级(例如,区域第一表),主机虚拟地址的特定位被用于索引到所指示的表中,其中表原点2152将要用于使用主机地址转换结构2172的转换,如参考图21A所描述的。继续进行与客户机RFTE相对应的主机虚拟地址的转换,直到获得主机页帧实地址(PFRA)2174a。

[0242] 中间主机页帧实地址处的数据是指向客机地址转换结构的下一级(例如,在该特定示例中,客机区域第二表2110)的指针,并且转换继续,如上所述。具体地,主机地址转换结构2176、2178、2180和2182用于分别转换与客机区域第二表2110、区域第三表2114、段表2118和页表2122相关联的中间主机虚拟地址,从而分别产生主机PFRA 2174b、2174c、2174d和2174e。主机页帧实地址2174e包括客户机页表条目2125的地址。客机页表条目2125包括客机页帧实地址2126,其与从初始客机虚拟地址的偏移连接以获得相应的客机绝对地址。在一些情况下,主存储原点和限制然后被应用来计算对应的主机虚拟地址,然后如上所述,使用地址转换结构2184来转换该主机虚拟地址,以获得主机页帧实地址2174f。然后,将主机页帧实地址与主机虚拟地址的偏移(例如,位52-63)结合(例如,连接),以获得最终主机实地址。这完成了客机虚拟地址到主机实地址的转换。

[0243] 尽管在以上示例中,转换开始于区域第一表,但这仅是一个示例。对于客户机或主机,转换可以在任何级开始。

[0244] 在一个实施例中,为了改进地址转换,虚拟地址到实或绝对地址转换映射被存储在转换后备缓冲器(TLB)中。TLB是由存储器管理硬件用来改进虚拟地址转换速度的缓存器。下一次请求虚拟地址的转换时,将检查TLB,如果TLB中存在TLB命中,则从其中取得实或绝对地址。否则,如上所述,执行页走。

[0245] 如所指示的,客机转换可被包括在TLB中。这些条目可以是隐含地包括一个或多个主机转换的复合客机/主机条目。例如,客机虚拟TLB条目可缓冲从初始客机虚拟地址向下到最终主机实或绝对地址的整个转换。在这种情况下,客机TLB条目隐含地包括所有中间主机转换2172、2176、2178、2180和2182以及最终主机转换2184,如上文图21B中所述。在另一个示例中,层级TLB可以包括第一级TLB中的条目和分离的第二级TLB中的条目,其中第一级TLB缓冲从初始客机虚拟地址向下到客户页表2122的相关原点的转换,第二级TLB缓冲从客机页表条目地址向下到最终主机实或绝对地址的转换。在这个示例中,第一级TLB中的客机条目隐含地包括中间主机转换2172、2176、2178和2180,它们对应于支持客机区域和段表的主机转换,而第二级TLB中的客机条目隐含地包括支持客机页表的中间主机转换2182和最终主机转换2184,如图21B所示。转换后备缓冲器的许多实现是可能的。

[0246] 在以上示例中,页帧实地址被包括在页表的页表条目中。页表包括一个或多个条目,并且页表条目的进一步细节将参见图22来描述。

[0247] 在一个示例中,页表条目(PTE)2200与存储器的特定页相关联,并且包括例如:

[0248] (a)页帧实地址(PFRA)2202:该字段提供了实(例如,主机实)存储地址的最左位。当这些位与在右边的虚拟地址的字节索引字段连接时,获得实地址。

[0249] (b)页无效指示符(I)2204:该字段控制与页表条目相关联的页是否可用。当指示符为0时,通过使用页表条目进行地址转换。当指示符是1时,页表条目不能用于转换。

[0250] (c)页保护指示符2206:该字段控制是否允许到该页中的存储访问。

[0251] (d)固定指示符2208:根据本发明的一方面,使用该字段来指示该页是否要被固

定。在一个示例中，1指示该页将被固定，而0指示该页将不被固定。

[0252] 页表条目可以包括比本文描述的更多、更少和/或不同的字段。例如，在Power架构中，PTE可包括指示对应的存储器块是否已被引用的引用指示符，和/或指示对应的存储器块已被存储到其中的改变指示符。其它变化是可能的。

[0253] 在本发明的又一方面，例如，配置状态寄存器基于主机和客机属性、上下文和/或执行环境(例如，线程状态、应用状态、操作系统状态、管理程序状态、特定客机或主机操作系统级等)而被分离和分配，以便实现增加的管理灵活性。作为示例，配置状态寄存器可以由管理程序、操作系统、应用、线程编号或其它执行环境等来分离。

[0254] 作为特定示例，管理程序特权配置状态寄存器被存储在管理程序分配的存储器单元(例如，页)中，操作系统特权配置状态寄存器被存储在存储器的操作系统单元(例如，页)中，等等。此外，当多个线程被支持并且配置状态寄存器针对每个线程而被复制时，则可以支持分离的存储器单元(例如，页)以用于每个线程。在图23中描述了这种分离的示例。

[0255] 如图23所示，线程或进程2302使用一组配置状态寄存器2300，由操作系统2306使用另一组配置状态寄存器2304，由管理程序2310使用另一组配置状态寄存器2308，并且由管理程序2314使用另一组配置状态寄存器2312。其它示例也是可能的。

[0256] 在一个示例中，用于特定执行环境的配置寄存器是针对该执行环境静态定义的，并且包括可由执行环境读取或写入的那些寄存器。在另一示例中，寄存器基于使用而被动态地分配。其它示例也是可能的。

[0257] 在一个实施例中，单独的存储器区域(例如，作为多个可分配的转换单元)被分配给每个单独可控的执行环境(例如，线程、进程、操作系统、管理程序)，并且因此，与该执行环境相关联的一组配置状态寄存器被分配给该存储器区域。作为示例，配置状态寄存器基于逻辑所有权而被分配给对应的存储器区域，因为一些配置状态寄存器可以从多个执行环境可访问(例如，可从操作系统读访问，以及从管理程序读/写(R/W))。

[0258] 尽管不同的执行环境可具有不同的特权级别，但在一个方面中，更高级的特权级别具有对更低级的特权级别的访问控制。特定的特权级别可以用LD_CSR和ST_CSR指令以及本文描述的同步操作来指定。

[0259] 在一个示例中，如上所述，配置状态寄存器编号被重新映射；即，配置状态寄存器编号的索引被压缩以相对于每个组共同定位寄存器。

[0260] 通过为每个执行环境提供特定一组的寄存器，促进了某些类型的处理，包括上下文切换。如上所述，基于执行环境来分配特定寄存器组以及将分离的存储器单元分配给这些寄存器组，促进了寄存器的管理以及使用这些寄存器的处理(包括上下文切换)。

[0261] 如这里所述，上下文切换可以通过改变配置状态基址寄存器中的基地址指针来执行，而不是通过卸载和重新加载配置状态寄存器状态。为了实现这一点，用户模式状态将可独立地从监管器状态切换，以便切换用户上下文；操作系统状态将可独立地从管理程序状态切换以便切换虚拟机；并且如果每个硬件线程或子处理器状态将独立于其它线程/子处理器进行切换，则这些线程或子处理器状态将是可独立切换的。分离的存储器区域和分离的可分配的配置状态寄存器促进了这一点。

[0262] 根据本发明的一个方面，分离的配置状态基址寄存器指定每组存储器内配置状态寄存器的位置(基址)。此外，在至少一个实施例中，对每个基址寄存器的访问控制将具有合

适的访问许可。例如,对于要由操作系统切换的上下文,操作系统特权是修改基址寄存器的最小先决条件;对于要由管理程序切换的上下文,管理程序特权是修改该基址寄存器的最小先决条件,等等。

[0263] 在另一方面,提供了通过主机级软件(例如管理程序或虚拟机监测器)防止存储器单元(例如页)的移动的能力,该主机级软件提供了针对一个或多个配置状态寄存器(即CSR后备存储器)的存储。在一个示例中,这包括固定存储器单元并提供对CSR后备存储器的自动固定的指示。

[0264] 当指示提供针对一个或多个配置状态寄存器的存储的存储器的基址的配置状态寄存器(即基址寄存器,诸如TCBR)被写入时,根据架构规范,向当前客户的主机提供指示。在至少一个实施例中,该指示对应于异常。在一个实施例中,异常类型指示对配置状态基址寄存器的写入事件。响应于接收到配置状态基址寄存器改变的指示,至少一个主机监视软件(例如,管理程序或虚拟机监测器)执行操作以更新页固定信息。在本发明的一个方面中,更新固定信息包括记录被固定的CSR后备存储器页的地址或设置对应于该页的固定指示符。在另一方面,更新固定信息还包括通过从对应于固定的CSR后备存储器的一个或多个记录地址的池中移除针对特定配置状态基址寄存器的先前记录地址或重置固定指示符来对先前固定的CSR后备存储器页解除固定。

[0265] 根据本发明的一个方面,这些更新确保一个或多个主机级不会页调出或移动CSR后备存储器,从而使缓存的地址转换无效,或者以其它方式导致对配置状态寄存器的更新,从而导致页转换错误。在本发明的另一方面,通过提供CSR后备存储器的位置的通知,并向一个或多个主机提供更新任何缓存的转换的机会,固定信息也可以用于移动CSR后备存储器的位置。

[0266] 此外,为了支持多级客户机制,初始化指向支持配置状态寄存器的存储器(CSR后备存储器)的基址寄存器的指令还可被指定为传递地在多个主机级上引发异常,以确保适当的固定。在一个实施例中,一级主机被通知,并且该主机将在适当时引起HCALL(管理程序调用)的固定。

[0267] 通过固定存储器中配置状态寄存器的存储器后备页,当访问配置状态寄存器时避免页错误。这可能是软件不期望的,并且在一些软件中导致panic(),诸如,例如当软件检查什么指令引起陷阱并且发现该指令是引起数据页错误的未被定义为访问存储器的指令时。panic()是当非期望事件发生时操作系统中执行的调用,并且通常导致系统崩溃。

[0268] 例如,固定被用于避免循环异常(例如,当用于异常相关的配置状态寄存器的页不可用时,针对该页的页错误异常将必然发生等);并且用于确保快速响应(例如,对异常和涉及配置状态寄存器处理的其它外部事件)。

[0269] 在一个实施例中,在软件中执行固定。例如,如下所述,固定可以在半虚拟化环境中使用管理程序调用(HCALL)结合软件上下文切换来执行。

[0270] 在一个实施例中,当监管软件对上下文(例如线程上下文、或进程上下文、或逻辑分区上下文、或虚拟机上下文、或操作系统上下文等)进行初始化时,监管器分配存储器以提供用于与被初始化的上下文相对应的一个或多个配置状态寄存器的存储。

[0271] 在一个示例中,这可以通过调用提供适当的对齐和大小的存储器的分配例程来执行。根据至少一个实施例,返回的地址被存储在存储与上下文对应的信息的存储器区域中。

在一个实施例中,该存储器区域被称为"u区域"并且由变量"u"表示。在至少一个实施例中,变量u是记录、结构、类或其它复合数据类型,其中多个成员对应于将要针对上下文记录的各种属性。在至少一个实施例中,该结构包括对应于至少一个CSR后备存储器页的地址的成员(字段)。在至少一个示例中,该成员被命名为"csr_page"。

[0272] my_csr_page_pointer=分配后备页

[0273] u.csr_page=my_csr_page_pointer

[0274] 当针对半虚拟化环境中的管理程序使用例如HCALL执行CSR后备存储器的固定时,上下文切换上下文序列(例如根据图19A和19B中一者的序列)通过固定和解除固定HCALL来增强。根据本发明的该方面的上下文切换的一个实施例,执行以下步骤:

[0275] (1)根据已知技术保存先前上下文的非CSR状态,包括但不限于通用寄存器、浮点寄存器、向量寄存器等;

[0276] (2)保存处理器内配置状态寄存器(例如,基于图19A和19B中的一者的技术);

[0277] (3)固定输入CSR后备存储器页(即,被激活为配置状态寄存器存储器页的页,作为接通(switch in)(激活)下一上下文中的一部分):HCALL(PIN,next_u->csr_page),其中next_u是指向作为下一上下文被接通(激活)的上下文的u区域的指针;

[0278] (4)可选地,在至少一个实施例中,同步输出配置状态寄存器:sync_o_csr;

[0279] (5)用对应于被激活的上下文的CSR后备存储器的基地址来加载基址寄存器(在一个示例中,该CSR对应于TCBR):mtspr TCBR,next_u->csr_page;

[0280] (6)可选地,在至少一个实施例中,同步输入配置状态寄存器:sync_i_csr;

[0281] (7)解除固定输出CSR后备存储器页(即,被去激活为CSR后备存储器页的页,作为关闭(switch out)(去激活)先前上下文的一部分):HCALL(UNPIN,prev_u->csr_page),其中prev_u是指向作为先前上下文被关闭(去激活)的上下文的u区域的指针;

[0282] (8)根据已知技术,加载下一上下文的其它非CSR状态,包括但不限于通用寄存器、浮点寄存器、向量寄存器等;

[0283] (9)将控制传递到新激活的上下文,例如,使用例如rfid指令(在结合Power ISA的实现中)从操作系统传递到应用线程或进程上下文。

[0284] 本领域技术人员将理解,可以对上述步骤进行重新排序。例如,在至少一个实施例中,可以在固定操作之前执行解除固定操作。其它变化是可能的。

[0285] 在另一个实施例中,响应于加载诸如TCBR的基址寄存器来执行固定。通知事件(例如,中断)可被引发给监管器。在一个实施例中,如果存在多个监管器级,则向每个监管器级引发通知事件。响应于接收到通知事件,更新固定信息。

[0286] 在另一实施例中,将值写入到基址寄存器使得指示被固定的页的页表条目指示标志被设置在对应的PTE中。如果存在多个级别的页表条目,则可针对每个级设置页表条目指示标志。

[0287] 在至少另一个实施例中,提供了启动固定或解除固定过程的固定和解除固定指令中的至少一个。

[0288] 在又一个实施例中,固定可以由主机软件通过检查系统中活动的基址寄存器以确定哪些页被"固定"来确定,即,基址寄存器(例如,TCBR)的多个内容表示被固定的页的记录。在至少一个实施例中,在移出或页调出页之前,监管器级软件通过确定页地址是否对应

于至少一个基址寄存器中的地址来确定页是否对应于固定的页。

[0289] 在又一实施例中，主机可以接收固定的通知事件，例如，在一个示例实施例中，作为异常。基于接收到该通知，主机系统接收要固定的地址并将其存储以供将来在存储器管理期间引用。在一个实施例中，通知还包括关于被解除固定的先前地址的信息(例如，存储在基配置状态寄存器中的先前值，或者例如使用配置状态寄存器以其它方式提供的另一值)。

[0290] 参见图24描述了关于向主机提供固定通知的一个示例的进一步细节。该逻辑由处理器执行。参见图24，针对包括存储器内配置状态寄存器的存储器页(或其它存储器单元)接收新配置值(例如，客机地址)，步骤2400。在一个示例中，该新配置值存储在基址寄存器(例如，TCBR)中。然而，在其它实施例中，可以以另一种方式提供该值。

[0291] 转换存储器页的客机地址(例如，客机虚拟地址或客机实地址)，步骤2402。在一个示例中，客机地址被转换为物理实地址，并且该转换被缓存以供将来访问。变量n被设置为等于客机级，步骤2404。然后，在步骤2406，n以选择值被递减，例如1。使用与被固定的客机地址对应的主机级n虚拟地址向主机级n通知固定事件，步骤2408。此外，确定是否存在更多的主机级(例如，n大于0)，询问2410。如果存在更多的主机级，则处理继续到步骤2406。否则，处理完成。

[0292] 在一个实施例中，固定由与地址对应的页表条目中的指示符(诸如位)来指示。位指示该页被固定并由客机使用。图22中描述了该固定指示符的一个示例。

[0293] 参见图25中描述的示例转换和固定操作来描述与固定有关的进一步细节。在一个示例中，处理器执行该处理。首先，接收针对包括存储器内配置状态寄存器的存储器页的新配置值(例如，客机地址)，步骤2500。将客机地址转换成物理地址，并且缓存该转换以供将来访问，步骤2502。变量n被设置为等于客机级，而ADDRESS(地址)被设置为等于客机虚拟地址，步骤2504。此后，n按定义值递减，例如1，并且ADDRESS被设置为等于translate_to_host(转换_到_主机)(ADDRESS,n)，步骤2506。即，ADDRESS被设置为针对主机级的转换的主机地址。固定指示符(例如，位)被设置(例如，设置为1)在地址的页表条目中，步骤2508。此外，确定是否存在更多的主机级;即，n是否大于0，询问2510?如果存在更多的主机级，则处理继续到步骤2506。否则，处理结束。在这一点上，ADDRESS对应于固定页的物理地址，并且可以用于与地址转换协作。

[0294] 在一个示例中，基于转换和缓存，在所有主机级处设置指示符(例如，位)。在一个实施例中，页表走(walk)和固定指示被组合。这改进了性能，因为转换访问用于固定指示的相同页表条目。

[0295] 在一个实施例中，对另一值(例如，在配置状态寄存器中存储的先前值(地址)或者例如使用配置寄存器以其它方式提供的另一值)执行解除固定。

[0296] 参照图26描述了与移动和解除固定操作相关的处理的一个示例。在一个示例中，处理器执行该处理。首先，接收对地址解除固定的请求，步骤2600。该请求包括要被解除固定的客机虚拟地址。此外，n被设置为等于客机级，而ADDRESS(地址)被设置为等于要被解除固定的客机虚拟地址，步骤2602。接下来，n按定义值递减，例如1，并且ADDRESS被设置为等于translate_to_host(转换_到_主机)(ADDRESS,n)，步骤2604。即，ADDRESS被设置为主机级的被转换的主机地址。用于该地址的页表条目中的固定指示符(例如，位)被重置(例如，

设置为0)，步骤2606。此后，确定是否存在更多的主机级(例如，n大于0)，步骤2608。如果存在更多主机级，则处理继续到步骤2604。否则，处理结束。此时，ADDRESS对应于解除固定的页的物理地址。

[0297] 如本文描述，基于确定要解除固定的存储器单元，例如自动提供通知。通知可以通过设置指示符、发出中断、提供异常等来进行。许多变化是可能的。

[0298] 在另一方面，经由半虚拟化的固定调用提供了高效的固定管理。期望不必在每次安装页时都对页进行固定和解除固定。另一方面，还期望限制固定的页的数量，以便不会不必要地分割主机的页缓存，并限制其页分配自由度。因此，引入了固定HCALL(管理程序调用)，其中客机指定将由主机解除固定的页。管理程序可以指示要被解除固定的页是否被解除固定，从而如果管理程序具有可用的资源，则给予客机不必针对每个页调用固定请求的灵活度。

[0299] 在一个实施例中，该调用包括更新指向CSR存储器后备页的基指针或地址(基址)。此外，在一个实施例中，客机指定其是否想要将页保持为固定的。

[0300] 在另一实施例中，管理程序可以通过回调操作系统来请求在管理程序运行到低资源情形时先前已留给操作系统的固定的页的返回。在一个实施例中，操作系统指定其想要解除固定的一个或多个页作为对回调的响应。

[0301] 根据一个或多个方面，如参照图27所述，单个调用(例如一个HCALL)用于例如由在处理器上执行的主机执行固定和解除固定操作。如所描绘的，在一个示例中，响应于一个管理程序调用2704，执行固定操作2700和解除固定操作2702。在一个示例中，在步骤2710，对第一地址(例如，第一基地址)执行解除固定操作以解除固定旧页，并且在步骤2720，对第二地址(例如，第二基地址)执行固定操作以固定新页。使用一个调用而不是多个调用，节省了处理时间。通过组合的固定和解除固定调用，指定要固定的新页(例如，CSR后备存储器页)以及指定要解除固定的先前页，来减少解除固定和固定调用的次数。

[0302] 在一个实施例中，如果管理程序控制台管理约束允许，则操作系统可以请求在调用中指定的要解除固定的地址不被解除固定。返回关于该地址是固定还是解除固定的响应。然后，管理程序仍可使用回调来请求操作系统使一个或多个被固定的页被解除固定。

[0303] 在一个示例中，操作系统保持多于活动的存储器内配置状态寄存器页所需的页数。在进一步的示例中，操作系统固定持有存储器内配置状态寄存器的所有页，无论是否是活动的。这消除了未来固定的需要。然而，这可能导致过多的固定的页的数量和系统低效。因此，在一个实施例中，操作系统提供回调功能，其中在太多页(或大于所选数量的数量)被固定在系统中以供存储器内配置状态寄存器使用时，管理程序可以调用操作系统来解除分配被固定的页。

[0304] 参考以下示例描述与解除固定/固定有关的进一步细节，其中解除固定/固定在上下文切换中被执行。具体地，以下示例描述了上下文固定的一个示例，其中客机(OS)请求主机(HV)切换固定，并且进一步可选地，保留固定。

[0305] 在一个实施例中，当监管软件初始化上下文(例如线程上下文、或进程上下文、或逻辑分区上下文、或虚拟机上下文、或操作系统上下文等)时，监管器分配存储器来为对应于被初始化的上下文的一个或多个配置状态寄存器提供存储。

[0306] 在一个示例中，这可以通过调用提供适当的对齐和大小的存储器的分配例程来执

行。根据至少一个实施例，返回的地址被存储在存储与上下文相对应的信息的存储器区域中。在一个实施例中，该存储器区域被称为"u区域"并且由变量"u"表示。在至少一个实施例中，变量u是记录、结构、类或其它复合数据类型，其中多个成员对应于针对上下文要记录的各种属性。在至少一个实施例中，该结构包括对应于至少一个CSR后备存储器页的地址的成员(字段)。在至少一个示例中，该成员被命名为"csr_page"。

[0307] my_csr_page_pointer＝allocate backing page(分配后备页)

[0308] u.csr_page＝my_csr_page_pointer

[0309] 当针对半虚拟化环境中的管理程序使用例如HCALL执行CSR后备存储器的固定时，上下文切换上下文序列(例如根据图19A和19B中一者的序列)通过固定和解除固定HCALL来增强。根据本发明的该方面的上下文切换的一个实施例，执行以下步骤：

[0310] (1)根据已知技术保存先前上下文的非CSR状态，包括但不限于通用寄存器、浮点寄存器、向量寄存器等；

[0311] (2)保存处理器内配置状态寄存器(例如，基于图19A和19B中的一者的技术)；

[0312] (3)可选地，在至少一个实施例中，同步输出配置状态寄存器:sync_o_csr；

[0313] (4)如果否(next_u->csr_page_pinned)

[0314] 找到页以解除固定 victim＝select_TCBR_for_unpin()；

[0315] 可期望保留? retain＝retain_desirable_p(victim)；

[0316] 给出获取固定页 lost_victim＝HCALL(PIN_give_to_get，

[0317] next_u->csr_page,victim,retain)

[0318] 受害者丢失? if(lost_victim)mark_unpinned(victim)；

[0319] (5)可选地，在至少一个实施例中，同步输入配置状态寄存器:sync_i_csr；

[0320] (6)根据已知技术，加载下一上下文的其它非CSR状态，包括但不限于通用寄存器、浮点寄存器、向量寄存器等；

[0321] (7)将控制传递到新激活的上下文，例如，使用例如rfid指令(在结合Power ISA的实现中)从操作系统传递到应用线程或进程上下文。

[0322] 参照图28描述了执行固定/解除固定操作的一个示例的进一步细节。经由例如单个管理程序调用以及指定是否正在进行旧保留OA作为固定的存储器的请求的保留指示符(或其的指示)，来接收用于固定新地址NA并解除固定旧地址OA的请求，步骤2800。作为示例，新地址指示CSR存储器后备页。地址NA被固定，步骤2810，并且确定保留指示符是否指定用于将地址OA保留为固定在存储器中的请求，询问2820。如果存在保留OA的固定的请求，则基于所指示的策略(例如，跨多个虚拟机的资源分配)和用于固定的可用存储器来确定是否准许固定页的保持，询问2822。如果请求被准许，询问2824，则做出解除固定操作将不被执行的指示，步骤2826。否则，在步骤2830，对地址OA解除固定，并且在步骤2832，提供执行解除固定的指示。

[0323] 返回到询问2820，如果保留指示符指定该固定将不被保留，则处理继续到步骤2830。

[0324] 如上所述，一个调用可以用于解除固定一个地址、固定另一个地址和/或请求要被解除固定的地址实际上未被解除固定。通过使用一个调用，促进了处理并且提高了性能。

[0325] 如本文描述，所选择的配置状态寄存器被存储在存储器中，而不是处理器中。通过

将寄存器存储在存储器中，可以实现某些好处和优化，包括与数据损坏检测和校正相关联的益处和优化。

[0326] 在一个示例中，存储器后备状态用于增强弹性并解决单事件翻转(single event upset,SEU)或软错误。单事件翻转是由电离辐射效应引入的状态变化。随着CMOS(互补金属氧化物半导体)特征尺寸的缩小，用于改变位的电荷量Q_{CRIT}也随之缩小，因为针对每个位存储了更少的电荷。为了补救单事件翻转的影响，应用数据保护。这包括例如使用寄存器的奇偶校验或纠错码(ECC)保护来分别检测和修复损坏的状态寄存器值。使用纠错码或奇偶校验保护，因为当可以将跨区域的设计分摊到许多寄存器上时，对于寄存器文件来说修复是合理地可负担的。对于处理器内配置寄存器，这通常是不可负担的，因为必须为每个寄存器设计单独的保护和恢复。

[0327] 然而，根据本发明的一个方面，配置状态寄存器被存储在存储器中，其中配置状态寄存器用根据本发明的一个方面的一个或多个高级保护机制(包括但不限于奇偶校验位和纠错码(ECC))来保护。

[0328] 在一个方面，处理器内配置状态寄存器也通过使用SEU弹性系统存储器层级来保护。在一个实施例中，利用检测SEU引起的损坏的技术来保护处理器内配置状态寄存器。各种检测技术可以与本发明的方面结合使用。在一个示例中，损坏检测机制对应于针对处理器内配置状态寄存器的数据奇偶校验保护的使用。在另一实施例中，可通过在没有对配置寄存器的写入的情况下测试寄存器值变化的标记来检测SEU引发的损坏。此外，该处理器内寄存器也被存储在存储器中，以确保处理器内配置寄存器的ECC保护的副本是可用的。在一个实施例中，响应于由此保护的处理器内配置寄存器的更新，还将更新的副本存储到存储器内副本。当处理器识别奇偶校验错误时，从ECC保护的存储器内副本取得该值，并且更新处理器内寄存器。

[0329] 此外，在一个实施例中，诸如指令地址、数据地址和内容断点寄存器的高使用率值被存储在后备存储器中，并且当检测到单事件翻转时可以被恢复。恢复包括奇偶校验保护，或者经由硬件重新加载路径或者通过执行机器检查，并且使机器检查处理程序重新加载这些寄存器。因此，根据本发明的一个方面，通过将配置状态寄存器存储在系统存储器中并使用例如ECC来保护单事件翻转，从而保护配置寄存器状态。

[0330] 下面描述了与使用用于配置状态寄存器的纠错码有关的进一步细节。特别地，参见图29A-29C描述了针对数据写入使用纠错码的示例，并且参见图30A-30C描述了针对数据读取使用纠错码的示例。

[0331] 首先参见图29A，在该示例中，数据写入未被保护。在此示例中，接收处理器内配置状态寄存器的值，步骤2900，并且将该值写入到实施配置状态寄存器的锁存器，步骤2902。该数据写入对于单事件翻转和其它类型的错误是不受保护的。

[0332] 相对地，参照图29B，在步骤2920，接收处理器内配置状态寄存器的值，并且在步骤2922，计算错误保护或纠错码。在步骤2924，所接收的值连同保护或纠错码被写入配置状态寄存器(例如，实现配置状态寄存器的锁存器)。

[0333] 此外，参见图29C，描述了用于存储器内配置状态寄存器的数据写入的一个实施例。在该示例中，接收存储器内配置状态寄存器的值，步骤2952，并且确定配置状态寄存器被存储在其中的系统存储器地址，步骤2954。然后，将该值存储到受保护的系统存储器地

址,因为它是受益于错误保护的存储器的一部分,步骤2956。

[0334] 在一个方面,存储包括例如计算针对所接收的存储器内配置状态寄存器值的纠错码,以及将所计算的纠错码与所接收的值一起存储。如已知的,纠错码将一个或多个奇偶校验位添加到表示值的数据位(例如,每数据位的一个或多个子集的一个或多个奇偶校验位),并且使用这些奇偶校验位来确定任何错误。如果在数据中存在错误,则奇偶校验位指示在数据位中存在错误的位置,从而允许进行校正(例如,将以二进制表示的一个或多个数据位改变为另一个值)。

[0335] 除了上述之外,参见图30A-30C描述了对处理器内配置状态寄存器和存储器内配置状态寄存器执行数据读取的一个示例。

[0336] 参见图30A,描述了其中不提供保护的处理器内配置状态寄存器的数据读取的一个示例。在该示例中,该值被提供给处理器逻辑,步骤3000。

[0337] 相对地,参见图30B描述了与从其中提供保护的处理器内配置状态寄存器读取值相关联的处理的一个示例。在此实例中,从锁存器接收值,步骤3020,并且检查校正码,步骤3022。如果没有检测到损坏,询问3024,则将该值提供给处理器逻辑,步骤3030。然而,返回到询问3024,如果检测到损坏,则计算纠正值,步骤3026,并且将纠正值写回到锁存器,步骤3028。此外,该值被提供给处理器逻辑,步骤3030。

[0338] 另外,参见图30C描述了使用存储器内配置状态寄存器的数据读取。在该示例中,接收配置状态寄存器编号,步骤3050。确定配置状态寄存器的系统存储器地址,步骤3052。从受保护的系统存储器读取该值,步骤3054,并且将该值提供给处理器逻辑,步骤3056。

[0339] 在一个方面,读取包括使用纠错码确定数据(例如,值)的损坏是否已经发生。如果检测到损坏,则可采取一个或多个动作,包括但不限于执行恢复。恢复可以包括使用纠错码来计算用于损坏值的纠正值。还存在其它示例。

[0340] 通过使用存储器内配置状态寄存器,获得了错误保护好处,避免了昂贵的附加步骤和将这种保护添加到实现处理器内配置状态寄存器的锁存器的延迟。

[0341] 作为示例,基于写入存储器内配置状态寄存器来提供纠错。可以基于接收要存储在存储器中的配置状态寄存器的值来生成错误检测和纠正码。基于接收到对存储配置状态寄存器的存储器的读取请求,错误检测和纠正码可用于检测损坏是否已发生。基于读取存储器内配置状态寄存器并检测损坏已经发生,来校正损坏。

[0342] 本文详细描述了与提供存储器内配置状态寄存器有关的方面。通过在存储器内提供配置状态寄存器,来促进处理,并且可以增强性能。可以实现某些改进和优化。

[0343] 存储器内配置状态寄存器可用于指令处理以及其它操作序列中。例如,可以基于从外部设备接收中断信号来接收异常。在处理异常的过程中,可访问一个或多个配置状态寄存器,诸如SRR0和SRR1。当这些寄存器是存储器内配置状态寄存器时,中断处理序列被扩展以包括加载和/或存储操作。其它示例和/或变化是可能的。

[0344] 本发明的一个或多个方面不可分地依赖于计算机技术,并便于计算机内的处理,从而改进其性能。参见图31A-31B描述了与本发明的一个或多个方面相关的便于计算环境内的处理的一个实施例的进一步细节。

[0345] 参见图31A,获取针对其分配了存储器的存储器内配置状态寄存器的标识(3100)。基于该标识来确定到存储器中的偏移,其中存储器内配置状态寄存器被存储在该偏移处

(3102)。偏移是基于存储器内配置状态寄存器的功能亲和度而被分配给存储器内配置状态寄存器的(3104)。至少使用偏移来访问存储器内配置状态寄存器(3106)。

[0346] 作为示例,偏移将存储器内配置状态寄存器置于与具有相同的功能亲和度的另一存储器内配置状态寄存器相同的缓存线中(3108)。在一个示例中,基于存储器内配置状态寄存器和另一存储器内配置状态寄存器两者在特定操作中被使用,存储器内配置状态寄存器和另一存储器内配置状态寄存器具有相同的功能亲和度(3110)。

[0347] 作为另一示例,偏移将存储器内配置状态寄存器置于与具有相同的功能亲和度的另一存储器内配置状态寄存器相邻的缓存线中(3112)。

[0348] 在一个示例中,参见图31B,偏移是在存储器的特定单元内的索引位置(3114)。此外,在一个示例中,提供存储器的特定单元的版本指示(3116)。

[0349] 在一个方面,该标识包括存储器内配置状态寄存器的寄存器编号(3118)。

[0350] 此外,在一个实施例中,确定偏移包括在数据结构中执行查找(3120)。在另一实施例中,确定偏移包括使用计算来确定偏移(3122)。

[0351] 在一个方面,将偏移返回给请求者,以便请求者访问存储器内配置状态寄存器(3124)。

[0352] 其它变化和实施例是可能的。

[0353] 其它类型的计算环境也可结合和使用本发明的一个或多个方面,包括但不限于仿真环境,仿真环境的示例参考图32A进行描述。在该示例中,计算环境20例如包括经由例如一个或多个总线28和/或其它连接彼此耦接的本机中央处理单元(CPU)22、存储器24、以及一个或多个输入/输出设备和/或接口26。作为示例,计算环境20可包括由纽约州阿蒙克市国际商业机器公司提供的PowerPC处理器或pSeries服务器;和/或基于由国际商业机器公司、英特尔公司或其它公司提供的架构的其它机器。

[0354] 本机中央处理单元22包括一个或多个本机寄存器30,诸如在环境内处理期间所使用的一个或多个通用寄存器和/或一个或多个专用寄存器。此类寄存器包括表示任何特定时间点的环境的状态的信息。

[0355] 此外,本机中央处理单元22执行存储在存储器24中的指令和代码。在一个特定示例中,中央处理单元执行存储在存储器24中的仿真器代码32。该代码使得在一个架构中配置的计算环境能够仿真另一架构。举例而言,仿真器器码32允许基于除z/架构之外的架构的机器(诸如,PowerPC处理器、pSeries服务器、或其它服务器或处理器)仿真z/架构并执行基于z/结构研发的软件和指令。

[0356] 参考图32B描述了与仿真器代码32有关的进一步细节。存储在存储器24中的访客指令40包括被研发为在除本机CPU 22的架构外的架构中执行的软件指令(例如,与机器指令相关)。例如,访客指令40可能已被设计为在z/结构处理器上执行,但是替代地,在本机CPU 22(其可以是例如英特尔处理器)上进行仿真。在一个示例中,仿真器代码32包括指令提取例程42以从存储器24获得一个或多个访客指令40,并且可选地为所获得的指令提供局部缓冲。仿真器代码322还包括指令转译例程44以确定已获得的访客指令的类型并将访客指令转译成一个或多个对应的本机指令46。该转译包括例如识别要由访客指令执行的功能并选择本机指令执行该功能。

[0357] 此外,仿真器代码32包括仿真控制例程48以使本机指令被执行。仿真控制例程48

可使本机CPU 22执行仿真一个或多个先前获得的访客指令的本机指令的例程，并且在该执行结束时，将控制返回到指令提取例程以仿真下一访客指令或访客指令组的获取。本机指令46的执行可包括将数据从存储器24加载到寄存器中，将数据从寄存器存储回存储器，或者执行如由转译例程确定的某些类型的算术或逻辑运算。

[0358] 例如，每个例程以软件实现，该软件存储在存储器中并由本机中央处理单元22执行。在其它示例中，一个或多个例程或操作以固件、硬件、软件或它们的一些组合来实现。仿真处理器的寄存器可使用本机CPU的寄存器30或通过使用存储器24中的位置来仿真。在实施例中，访客指令40、本机指令46和仿真器代码32可存在于同一存储器中或者可分配在不同的存储器设备中。

[0359] 如本文中所使用的，固件包括例如处理器的微代码或毫代码(Millicode)。例如，固件包括用于实现较高级机器代码的硬件级指令和/或数据结构。在一个实施例中，固件包括例如专有代码，该专有代码通常被传送作为包括针对基础硬件的受信任软件或微代码并且控制操作系统对系统硬件的访问的微代码。

[0360] 所获得、转换和执行的客户指令40可以是例如本文描述的指令中的一者。具有一种架构(例如，z/架构)的指令从存储器中被提取、转译并表示为具有另一架构(例如，PowerPC、pSeries、英特尔等)的一系列本机指令46。此类本机指令然后被执行。

[0361] 一个或多个方面可以涉及云计算。

[0362] 首先应当理解，尽管本公开包括关于云计算的详细描述，但其中记载的技术方案的实现却不限于云计算环境。相反，本发明的实施例能够结合现在已知或以后开发的任何其它类型的计算环境来实现。

[0363] 云计算是一种服务交付模型，用于实现对可配置计算资源(例如，网络、网络带宽、服务器、处理、存储器、存储、应用、虚拟机和服务)的共享池的方便的按需网络访问，该可配置计算资源可以以最小的管理努力或与服务的提供者的交互来快速供应和释放。这种云模式可以包括至少五个特征、至少三个服务模型和至少四个部署模型。

[0364] 特征如下：

[0365] 按需自助服务：云消费者可以单方面地自动地根据需要提供计算能力，诸如服务器时间和网络存储，而不需要与服务的提供者进行人工交互。

[0366] 广域网接入：能力在网络上可用，并且通过促进由异构的薄或厚客户端平台(例如，移动电话、膝上型计算机和PDA)使用的标准机制来访问。

[0367] 资源池化：供应商的计算资源被池化以使用多租户模型来服务多个消费者，其中不同的物理和虚拟资源根据需求被动态地分配和重新分配。存在位置无关的意义，因为消费者通常不控制或不知道所提供的资源的确切位置，但是能够在较高抽象级别(例如国家、州或数据中心)指定位置。

[0368] 快速弹性：在一些情况下，可以快速且弹性地提供快速向外扩展的能力和快速向内扩展的能力。对于消费者，可用于提供的能力通常看起来不受限制，并且可以在任何时间以任何数量购买。

[0369] 测量服务：云系统通过利用在适合于服务类型(例如，存储、处理、带宽和活动用户账户)的某一抽象级别的计量能力来自动地控制和优化资源使用。可以监视、控制和报告资源使用，从而为所利用服务的提供者和消费者两者提供透明度。

[0370] 服务模型如下：

[0371] 软件即服务(SaaS)：提供给消费者的能力是使用在云基础设施上运行的提供者的应用。应用可通过诸如网络浏览器(例如，基于web的电子邮件)等瘦客户机接口从各种客户机设备访问。消费者不管理或控制包括网络、服务器、操作系统、存储、或甚至个别应用能力的底层云基础设施，可能的例外是有限的用户专用应用配置设置。

[0372] 平台即服务(PaaS)：提供给消费者的能力是将消费者创建或获取的应用部署到云基础设施上，该应用是使用由提供商支持的编程语言和工具创建的。消费者不管理或控制包括网络、服务器、操作系统或存储的底层云基础设施，但具有对部署的应用和可能的应用托管环境配置的控制。

[0373] 基础设施即服务(IaaS)：提供给消费者的能力是提供处理、存储、网络和消费者能够部署和运行任意软件的其它基本计算资源，该软件可以包括操作系统和应用。消费者不管理或控制底层云基础设施，但具有对操作系统、存储、部署的应用的控制，以及可能对选择的联网组件(例如，主机防火墙)的有限控制。

[0374] 部署模型如下：

[0375] 私有云：云基础设施仅为组织操作。它可以由组织或第三方管理，并且可以存在于该组织内部或外部。

[0376] 社区云：云基础设施由若干组织共享，并且支持具有共享关注(例如，任务、安全要求、策略和合规性考虑)的特定共同体。它可以由组织或第三方管理，并且可以存在于该共同体内部或外部。

[0377] 公有云：云基础设施可用于一般公众或大型工业群体，并且由销售云服务的组织拥有。

[0378] 混合云：云基础设施是两个或更多云(私有、共同体或公共)的组合，这些云保持唯一实体，但是通过使能数据和应用可移植性(例如，用于云之间的负载平衡的云突发传输)的标准化或私有技术绑定在一起。

[0379] 云计算环境是面向服务的，特点集中在无状态、低耦合、模块性和语义互操作性。在云计算的核心是包括互连节点的网络的基础设施。

[0380] 现在参考图33，描绘了示意性的云计算环境50。如图所示，云计算环境50包括云的消费者使用本地计算设备可以与其通信的一个或多个云计算节点10，本地计算设备例如是个人数字助理(PDA)或蜂窝电话54A，台式计算机54B，膝上型计算机54C和/或汽车计算机系统54N。节点10可以彼此通信。它们可以在一个或多个网络中物理地或虚拟地分组(未示出)，例如如上所述的私有云，共同体云，公共云或混合云，或其组合。这样，云的消费者无需维护本地计算设备上的资源就能够允许云计算环境50提供基础设施即服务、平台即服务和/或软件即服务。应该理解，图33中所示的计算设备54A-N的类型仅仅是示意性的，而计算节点10和云计算环境50可以(例如，使用网络浏览器)通过任何类型的网络和/或网络可寻址连接与任何类型的计算设备通信。

[0381] 现在参见图34，示出了由云计算环境50(图33)提供的一组功能抽象层。应当预先理解，图34中所示的组件、层和功能仅旨在说明，并且本发明的实施例不限于此。如所描绘的，提供了以下层和相应的功能：

[0382] 硬件和软件层60包括硬件和软件组件。硬件组件的示例包括主机61；基于RISC(精

简指令集计算机)体系结构的服务器62;服务器63;刀片服务器64;存储设备65;网络和网络组件66。在一些实施例中,软件组件包括网络应用服务器软件67和数据库软件68。

[0383] 虚拟层70提供抽象层,从该抽象层可以提供以下虚拟实体的示例:虚拟服务器71;虚拟存储72;虚拟网络73(包括虚拟私有网络);虚拟应用和操作系统74;和虚拟客户端75。

[0384] 在一个示例中,管理层80可以提供下面描述的功能。资源供应功能81提供用于在云计算环境内执行任务的计算资源和其它资源的动态获取。计量和定价功能82在云计算环境内对资源的使用进行成本跟踪,并且提供用于消费这些资源的帐单或发票。在一个示例中,这些资源可以包括应用软件许可。安全功能为云的消费者和任务提供身份认证,以及为数据和其它资源提供保护。用户门户功能83为消费者和系统管理员提供对云计算环境的访问。服务水平管理功能84提供云计算资源的分配和管理,以满足所需的服务水平。服务水平协议(SLA)计划和履行功能85为根据SLA预测的对云计算资源未来需求提供预先安排和供应。

[0385] 工作负载层90提供可以利用云计算环境的功能的示例。可以从该层提供的工作负载和功能的示例包括:地图绘制与导航91;软件开发和生命周期管理92;虚拟教室的教学提供93;数据分析处理94;交易处理95;和内容表处理96。

[0386] 在任何可能的技术细节结合层面,本发明可以是系统、方法和/或计算机程序产品。计算机程序产品可以包括计算机可读存储介质,其上载有用于使处理器实现本发明的各个方面的计算机可读程序指令。

[0387] 计算机可读存储介质可以是可以保持和存储由指令执行设备使用的指令的有形设备。计算机可读存储介质例如可以是——但不限于——电存储设备、磁存储设备、光存储设备、电磁存储设备、半导体存储设备或者上述的任意合适的组合。计算机可读存储介质的更具体的示例(非穷举的列表)包括:便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、静态随机存取存储器(SRAM)、便携式压缩盘只读存储器(CD-ROM)、数字多功能盘(DVD)、记忆棒、软盘、机械编码设备、例如其上存储有指令的打孔卡或凹槽内凸起结构、以及上述的任意合适的组合。这里所使用的计算机可读存储介质不被解释为瞬时信号本身,诸如无线电波或者其他自由传播的电磁波、通过波导或其他传输媒介传播的电磁波(例如,通过光纤电缆的光脉冲)、或者通过电线传输的电信号。

[0388] 这里所描述的计算机可读程序指令可以从计算机可读存储介质下载到各个计算/处理设备,或者通过网络、例如因特网、局域网、广域网和/或无线网下载到外部计算机或外部存储设备。网络可以包括铜传输电缆、光纤传输、无线传输、路由器、防火墙、交换机、网关计算机和/或边缘服务器。每个计算/处理设备中的网络适配卡或者网络接口从网络接收计算机可读程序指令,并转发该计算机可读程序指令,以供存储在各个计算/处理设备中的计算机可读存储介质中。

[0389] 用于执行本发明操作的计算机可读程序指令可以是汇编指令、指令集架构(ISA)指令、机器指令、机器相关指令、微代码、固件指令、状态设置数据、集成电路配置数据或者以一种或多种编程语言的任意组合编写的源代码或目标代码,所述编程语言包括面向对象的编程语言—诸如Smalltalk、C++等,以及过程式编程语言—诸如“C”语言或类似的编程语言。计算机可读程序指令可以完全地在用户计算机上执行、部分地在用户计算机上执行、作

为一个独立的软件包执行、部分在用户计算机上部分在远程计算机上执行、或者完全在远程计算机或服务器上执行。在涉及远程计算机的情形中，远程计算机可以通过任意种类的网络—包括局域网(LAN)或广域网(WAN)—连接到用户计算机，或者，可以连接到外部计算机(例如利用因特网服务提供商来通过因特网连接)。在一些实施例中，通过利用计算机可读程序指令的状态信息来个性化定制电子电路，例如可编程逻辑电路、现场可编程门阵列(FPGA)或可编程逻辑阵列(PLA)，该电子电路可以执行计算机可读程序指令，从而实现本发明的各个方面。

[0390] 这里参照根据本发明实施例的方法、装置(系统)和计算机程序产品的流程图和/或框图描述了本发明的各个方面。应当理解，流程图和/或框图的每个方框以及流程图和/或框图中各方框的组合，都可以由计算机可读程序指令实现。

[0391] 这些计算机可读程序指令可以提供给通用计算机、专用计算机或其它可编程数据处理装置的处理器，从而生产出一种机器，使得这些指令在通过计算机或其它可编程数据处理装置的处理器执行时，产生了实现流程图和/或框图中的一个或多个方框中规定的功能/动作的装置。也可以把这些计算机可读程序指令存储在计算机可读存储介质中，这些指令使得计算机、可编程数据处理装置和/或其他设备以特定方式工作，从而，存储有指令的计算机可读介质则包括一个制造品，其包括实现流程图和/或框图中的一个或多个方框中规定的功能/动作的各个方面的指令。

[0392] 也可以把计算机可读程序指令加载到计算机、其它可编程数据处理装置、或其它设备上，使得在计算机、其它可编程数据处理装置或其它设备上执行一系列操作步骤，以产生计算机实现的过程，从而使得在计算机、其它可编程数据处理装置、或其它设备上执行的指令实现流程图和/或框图中的一个或多个方框中规定的功能/动作。

[0393] 附图中的流程图和框图显示了根据本发明的多个实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上，流程图或框图中的每个方框可以代表一个模块、程序段或指令的一部分，所述模块、程序段或指令的一部分包括一个或多个用于实现规定的逻辑功能的可执行指令。在有些作为替换的实现中，方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如，两个连续的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意的是，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以用执行规定的功能或动作的专用的基于硬件的系统来实现，或者可以用专用硬件与计算机指令的组合来实现。

[0394] 除了上述之外，可以由提供客户环境管理的服务提供商提供、给予、部署、管理、服务一个或多个方面。例如，服务提供商可以创建、维护、支持计算机代码和/或为一个或多个客户执行一个或多个方面的计算机基础设施。作为回报，服务提供商可以例如根据订阅和/或费用协议从客户接收付款。附加地或替代地，服务提供商可以从向一个或多个第三方销售广告内容来接收付款。

[0395] 在一方面，可以部署应用以执行一个或多个实施例。作为一个示例，应用的部署包括提供可操作以执行一个或多个实施例的计算机基础结构。

[0396] 作为另一方面，可以部署计算基础设施，包括将计算机可读代码集成到计算系统中，其中与计算系统结合的代码能够执行一个或多个实施例。

[0397] 作为又一方面，可以提供一种用于集成计算基础设施的过程，包括将计算机可读

代码集成到计算机系统中。该计算机系统包括计算机可读介质，其中该计算机介质包括一个或多个实施例。与计算机系统结合的代码能够执行一个或多个实施例。

[0398] 尽管以上描述了各种实施例，但这些仅是示例。例如，具有其它体系结构的计算环境可用于合并和使用一个或多个实施例。此外，可以使用不同的指令或操作。另外，可以使用不同的寄存器和/或可以指定其它类型的指示(除了寄存器编号外)。许多变化是可能的。

[0399] 此外，其它类型的计算环境可以受益并被使用。作为示例，适用于存储和/或执行程序代码的数据处理系统是可用的，其包括通过系统总线直接或间接耦接到存储器元件的至少两个处理器。存储器元件包括例如在程序代码的实际执行期间使用的局部存储器，大容量存储和缓存存储器，该缓存存储器提供至少一些程序代码的临时存储，以便减少执行期间必须从大容量存储重新取回代码的次数。

[0400] 输入/输出或I/O设备(包括但不限于键盘，显示器，指示设备，DASD，磁带，CD，DVD，拇指驱动器和其它存储介质等)可以直接耦接到系统或通过介入I/O控制器而耦接到系统。网络适配器还可以耦接到系统，以使数据处理系统能够通过介入私有或公共网络而耦接到其它数据处理系统或远程打印机或存储设备。调制解调器，电缆调制解调器和以太网卡只是可用类型的网络适配器中的一小部分。

[0401] 本文使用的术语仅出于描述特定实施例的目的，并不意图限制本发明。如这里所使用的，单数形式“一”，“一个”和“该”旨在也包括复数形式，除非上下文另有明确说明。将进一步理解，当在本说明书中使用时，术语“包括”和/或“包含”指定所述特征、整数、步骤、操作、元素和/或组件的存在，但不排除存在或者添加一个或多个其它特征、整数、步骤、操作、元素、组件和/或它们的组合。

[0402] 以下权利要求中的所有装置或步骤加功能元件的相应结构、材料、动作和等同物(如果有的话)旨在包括如所具体要求保护的用于结合其它要求保护的元件来执行功能的任何结构、材料或动作。已经出于说明和描述的目的给出了对一个或多个实施例的描述，但是并不旨在穷举或限制于所公开的形式。许多修改和变化对于本领域普通技术人员来说是显而易见的。选择和描述实施例是为了最好地解释各个方面和实际应用，并且使本领域普通技术人员能够理解具有各种修改的各种实施例适合于预期的特定用途。

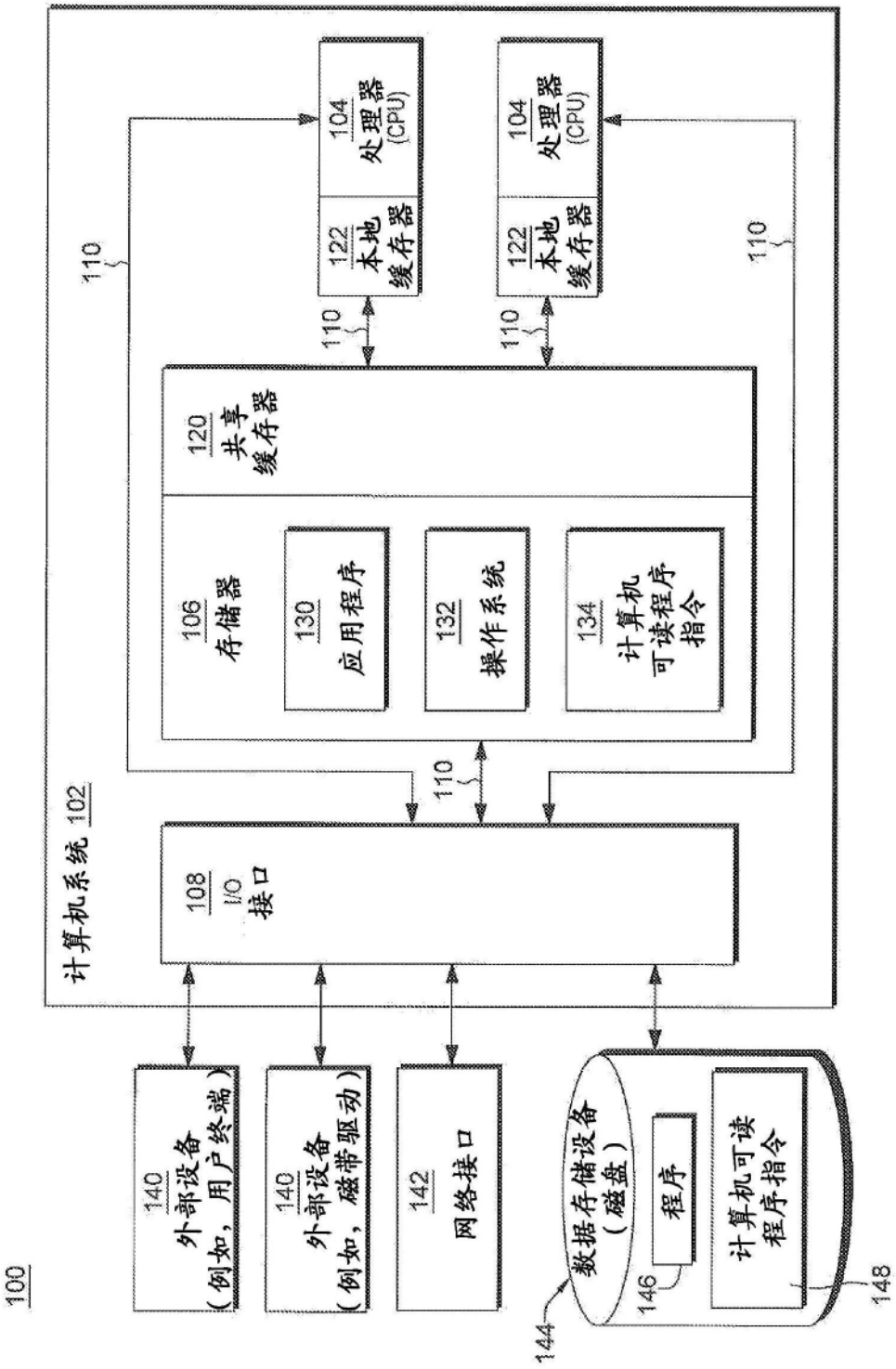

100
计算机系统 102
104 处理器 (CPU)
122 本地缓存器
104 处理器 (CPU)
122 本地缓存器
110
120 共享缓存器
106 存储器
130 应用程序
132 操作系统
134 计算机可读程序指令
108 I/O 接口
140 外部设备 (例如，用户终端)
140 外部设备 (例如，磁带驱动)
142 网络接口
数据存储设备 (磁盘)
146 程序
计算机可读程序指令
144
148

图1A

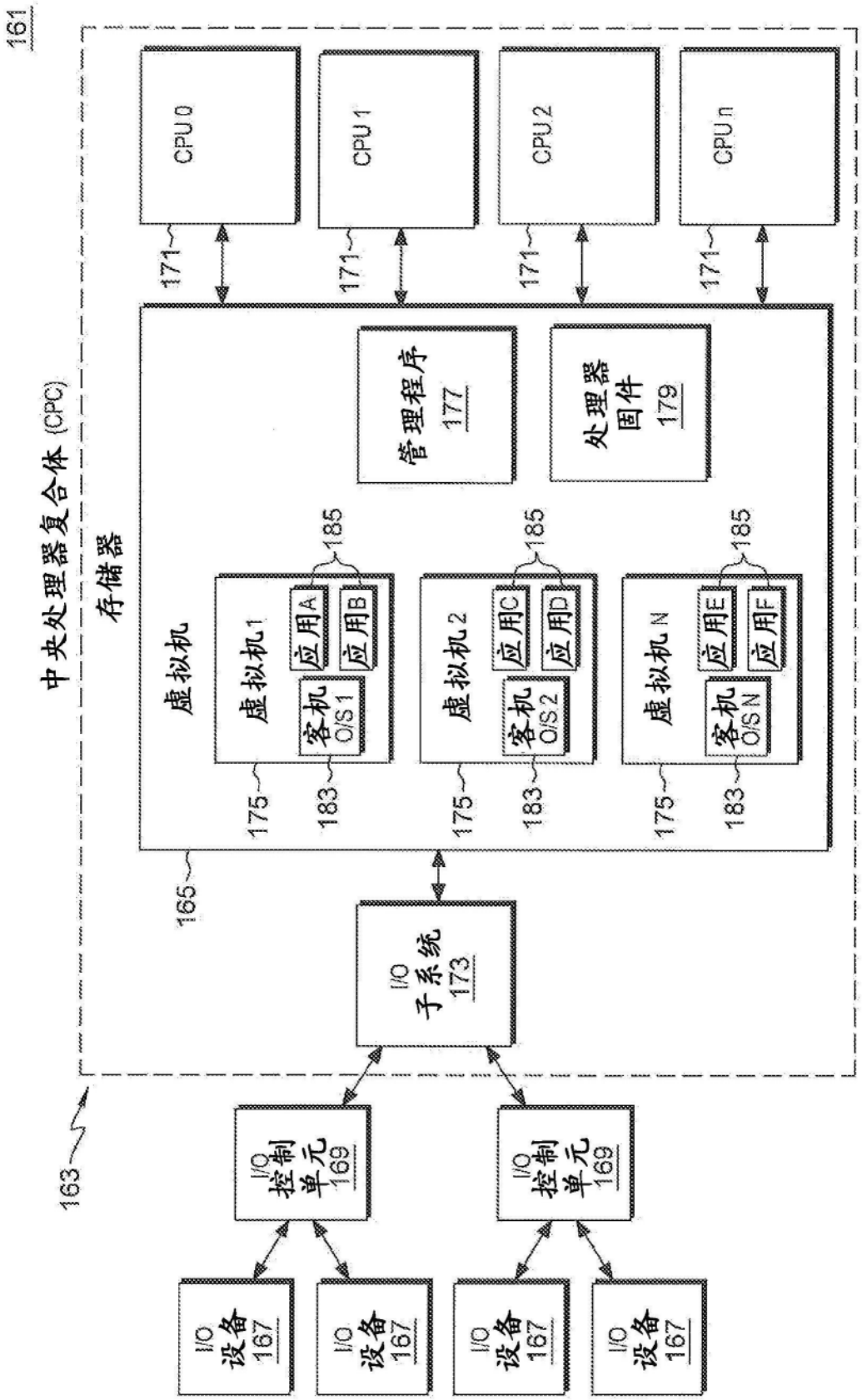

161
中央处理器复合体 (CPC)
存储器
CPU 0
CPU 1
CPU 2
CPU n
171
虚拟机
虚拟机 1
应用A
应用B
客机 O/S 1
虚拟机 2
应用C
应用D
客机 O/S 2
虚拟机 N
应用E
应用F
客机 O/S N
185
175
183
管理程序 177
处理器固件 179
165
I/O 子系统 173
163
I/O 控制单元 169
I/O 设备 167

图1B

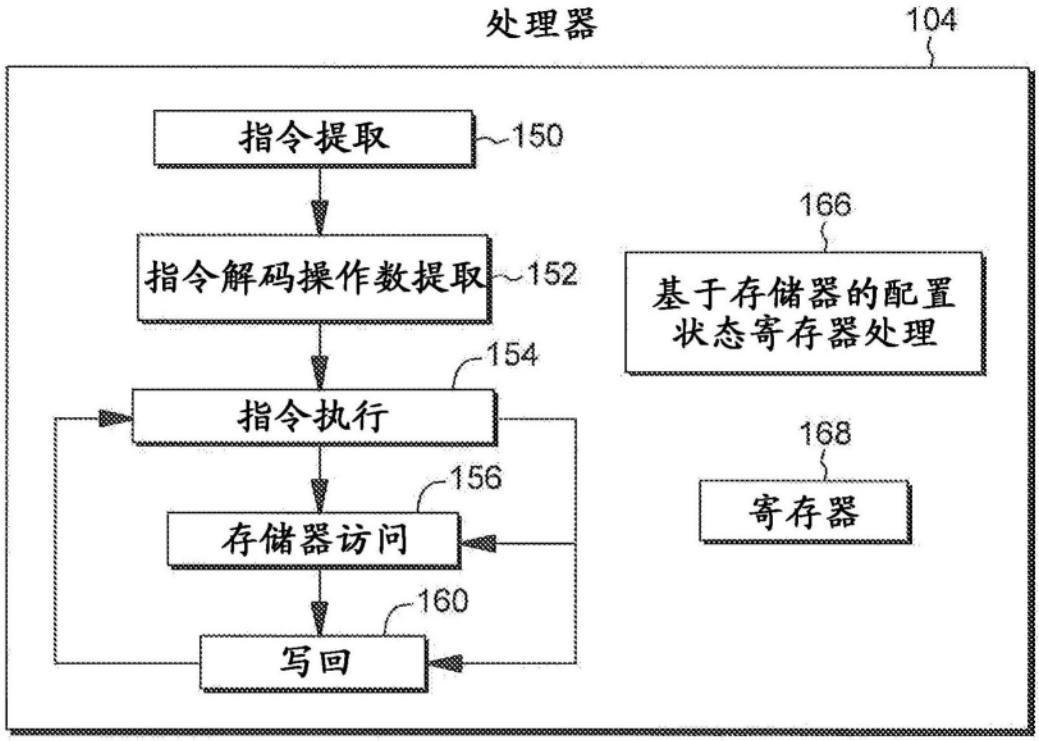

处理器
104
指令提取
150
指令解码操作数提取
152
指令执行
154
存储器访问
156
写回
160
166
基于存储器的配置状态寄存器处理
168
寄存器

图1C

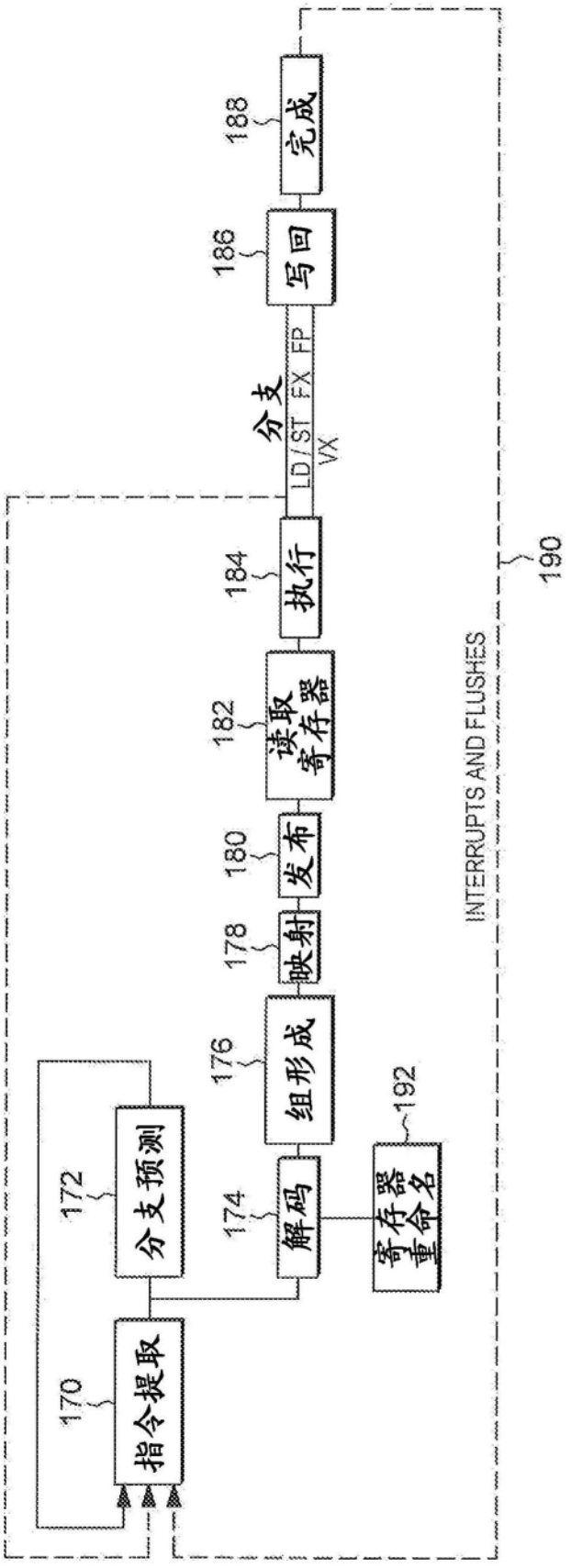

170
指令提取
172
分支预测
174
解码
176
组形成
178
映射
180
发布
182
读取寄存器
184
执行
分支
LD/ST FX FP
VX
186
写回
188
完成
192
寄存器命名
INTERRUPTS AND FLUSHES
190

图1D

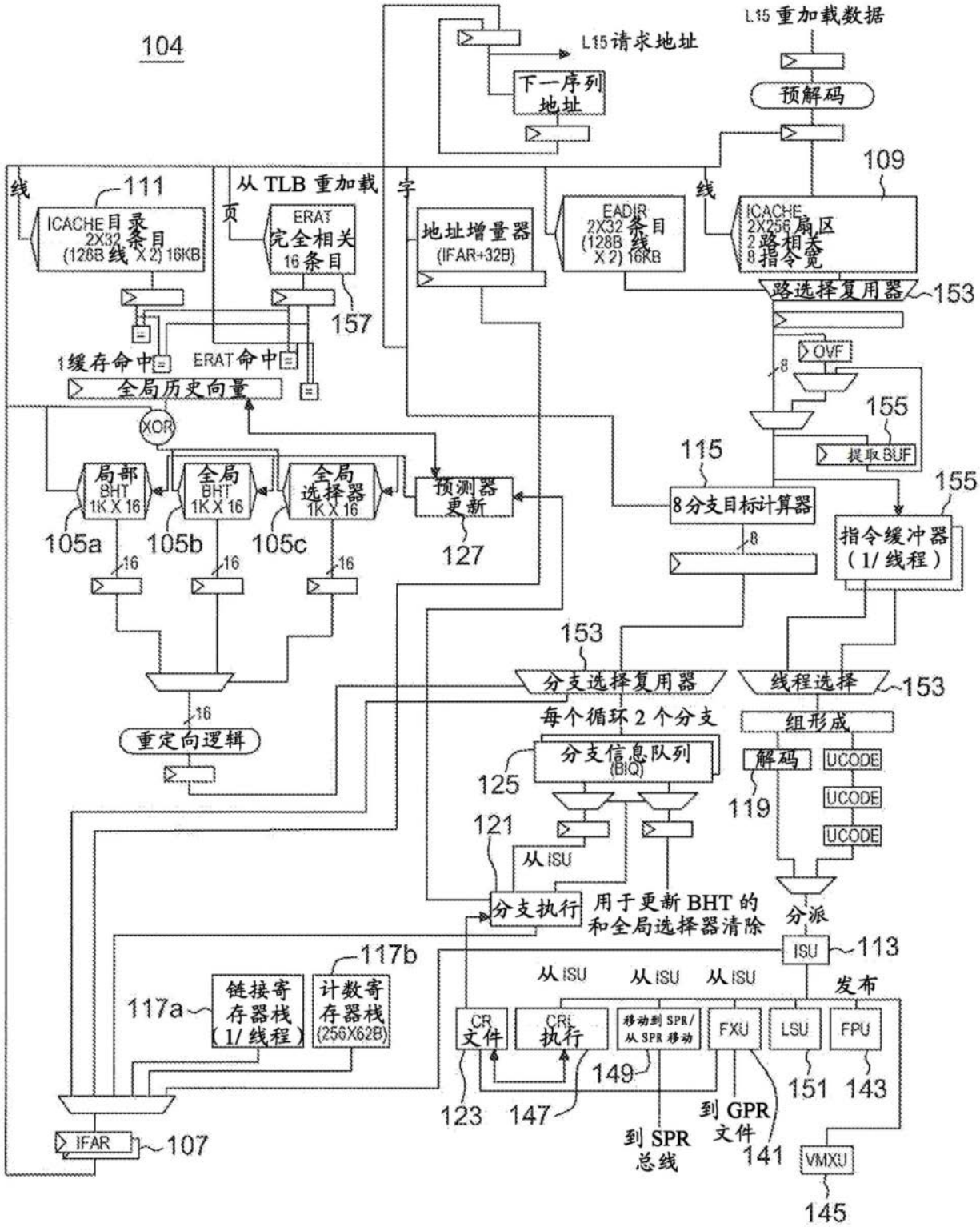

104
L15请求地址
L15重加载数据
下一序列地址
预解码
109
111
从TLB重加载
线
字
页
ICACHE目录 2X32条目 (128B线X2) 16KB
ERAT 完全相关 16条目
地址增量器 (IFAR+32B)
EADIR 2X32条目 (128B线X2) 16KB
ICACHE 2X256扇区 2路相关 8指令宽
路选择复用器
153
157
1缓存命中
ERAT命中
全局历史向量
XOR
OVF
8
155
提取BUF
115
局部 BHT 1K X 16
全局 BHT 1K X 16
全局选择器 1K X 16
预测器更新
8分支目标计算器
指令缓冲器 (1/线程)
105a
105b
105c
127
16
分支选择复用器
线程选择
每个循环2个分支
组形成
重定向逻辑
分支信息队列 (BIQ)
解码
UCODE
125
119
121
从ISU
分支执行
用于更新BHT的和全局选择器清除
分派
ISU
113
117b
117a
链接寄存器栈 (1/线程)
计数寄存器栈 (256X62B)
发布
CR文件
CR执行
移动到SPR/从SPR移动
FXU
LSU
FPU
149
151
143
123
147
到SPR总线
到GPR文件
141
IFAR
107
VMXU
145

图1E

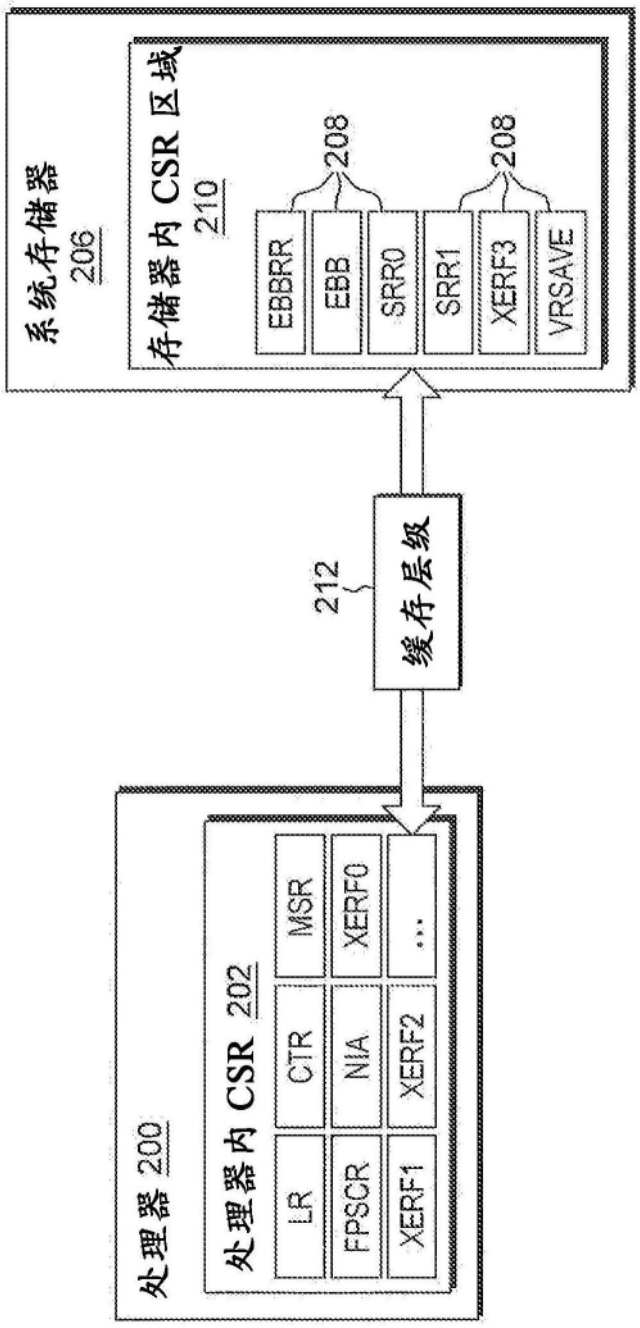

系统存储器
206
存储器内 CSR 区域
210
208
208
EBBRR
EBB
SRR0
SRR1
XERF3
VRSAVE
212
缓存层级
处理器 200
处理器内 CSR 202
LR
CTR
MSR
FPSCR
NIA
XERF0
XERF1
XERF2
...

图2

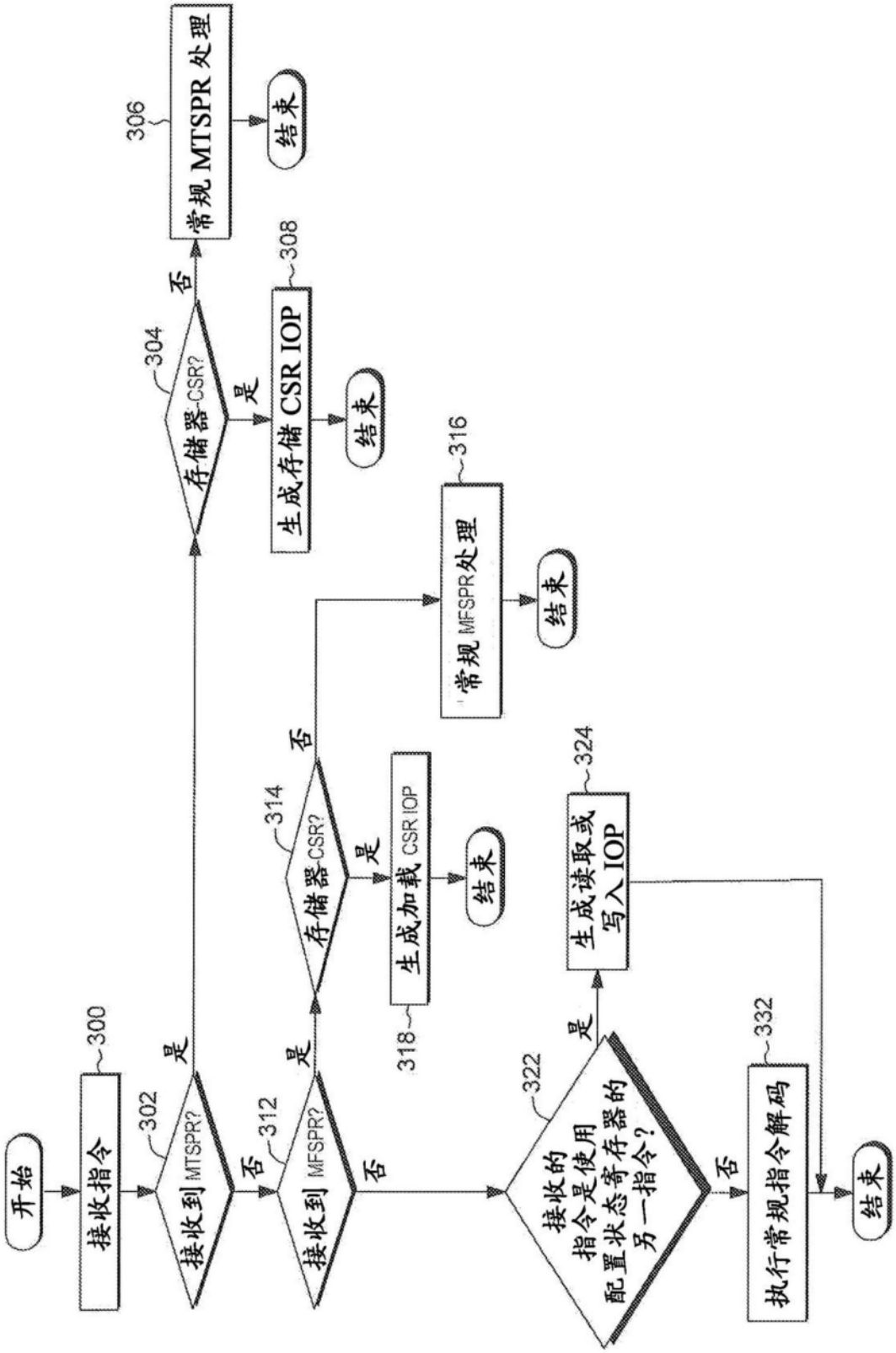

开始
接收指令
300
接收到 MTSPR?
302
是
否
306
常规 MTSPR 处理
结束
304
存储器-CSR?
是
否
308
生成存储 CSR IOP
结束
接收到 MFSPR?
312
是
否
314
存储器-CSR?
是
否
318
生成加载 CSR IOP
结束
316
常规 MFSPR处理
结束
322
接收的指令是使用配置状态寄存器的另一指令?
是
否
324
生成读取或写入 IOP
332
执行常规指令解码
结束

图3

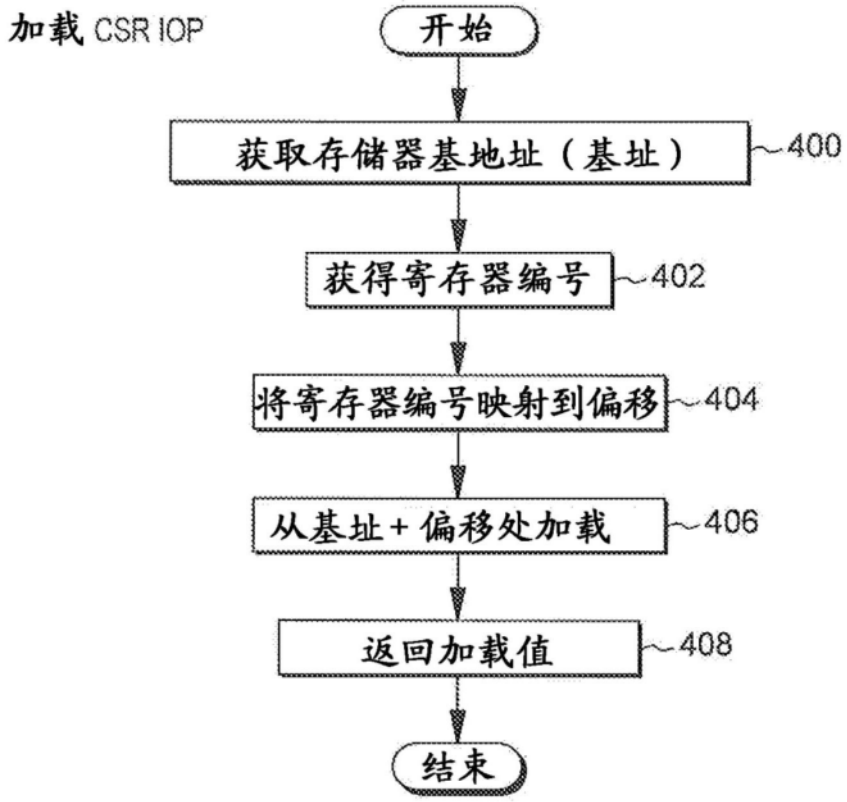

加载 CSR IOP
开始
获取存储器基地址（基址）
400
获得寄存器编号
402
将寄存器编号映射到偏移
404
从基址+偏移处加载
406
返回加载值
408
结束

图4

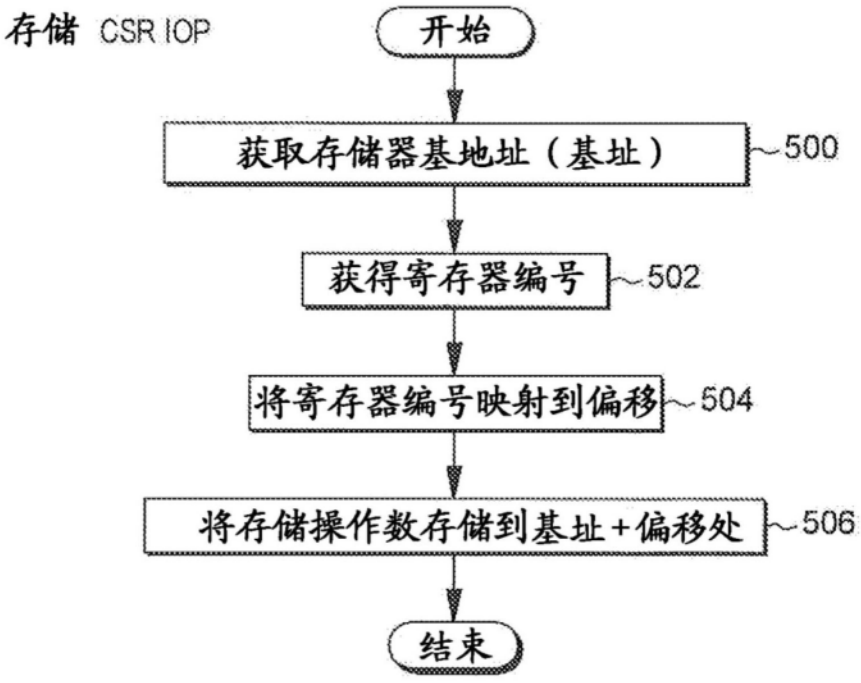

存储 CSR IOP
开始
获取存储器基地址（基址）
500
获得寄存器编号
502
将寄存器编号映射到偏移
504
将存储操作数存储到基址+偏移处
506
结束

图5

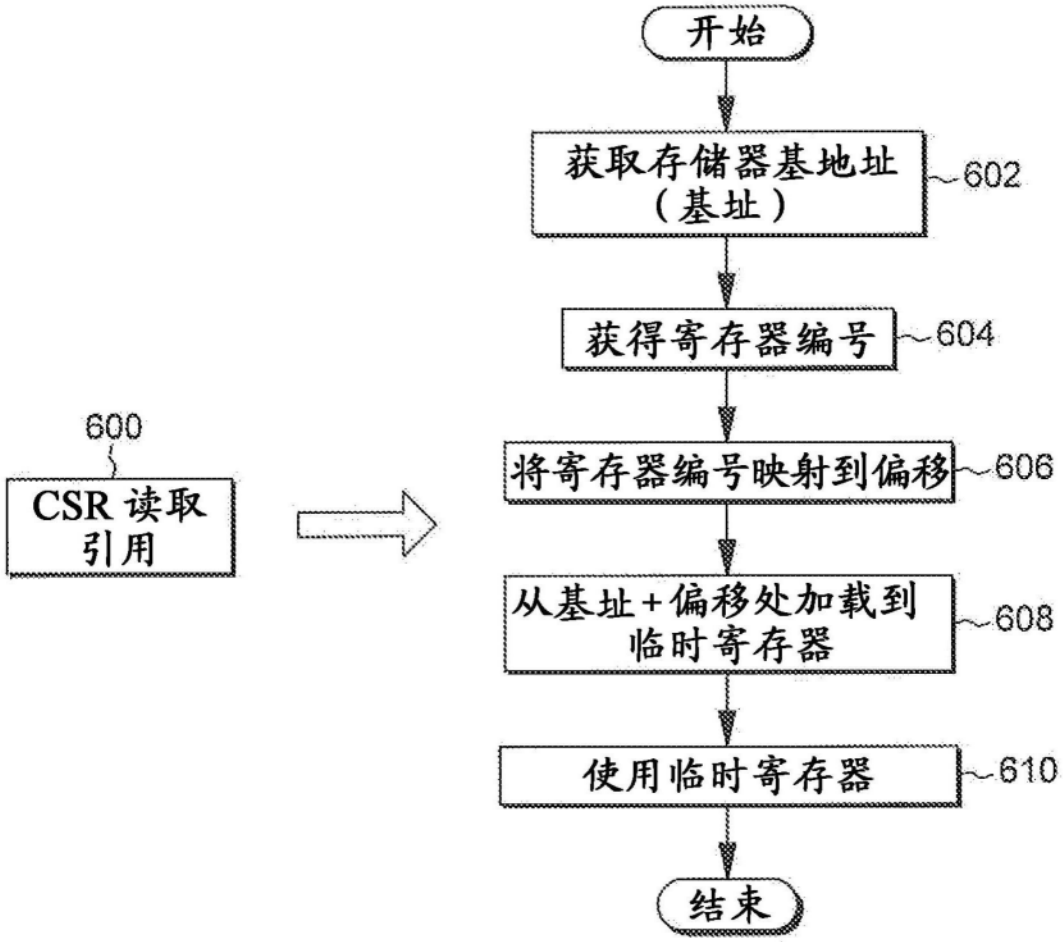

600
CSR 读取
引用
开始
获取存储器基地址
（基址）
602
获得寄存器编号
604
将寄存器编号映射到偏移
606
从基址+偏移处加载到
临时寄存器
608
使用临时寄存器
610
结束

图6

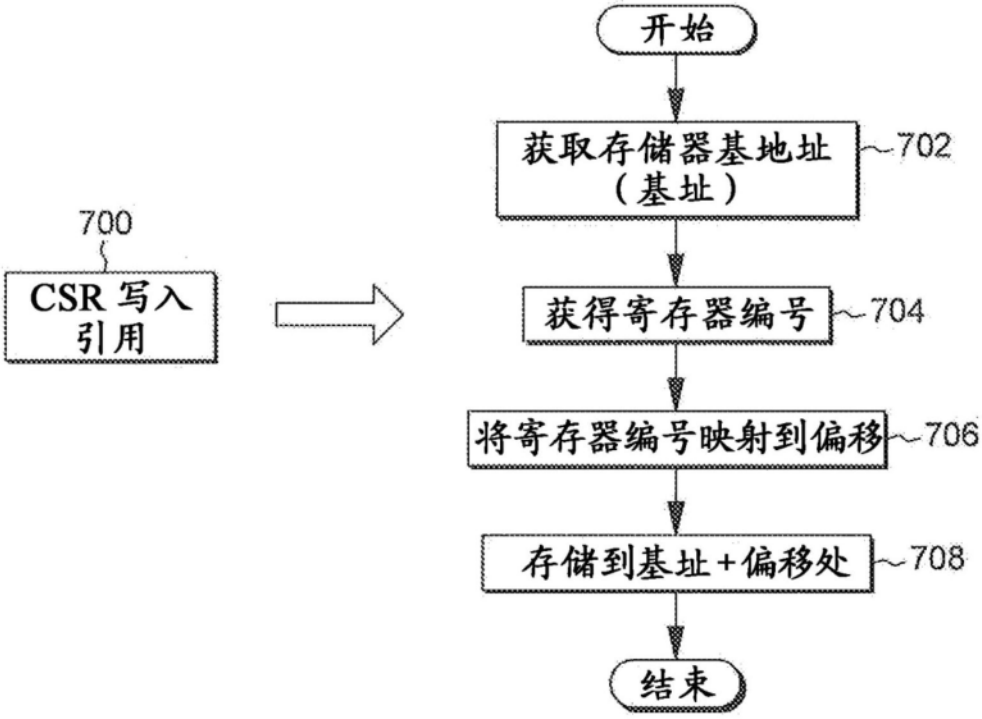

700
CSR 写入引用
开始
获取存储器基地址（基址）
702
获得寄存器编号
704
将寄存器编号映射到偏移
706
存储到基址+偏移处
708
结束

图7

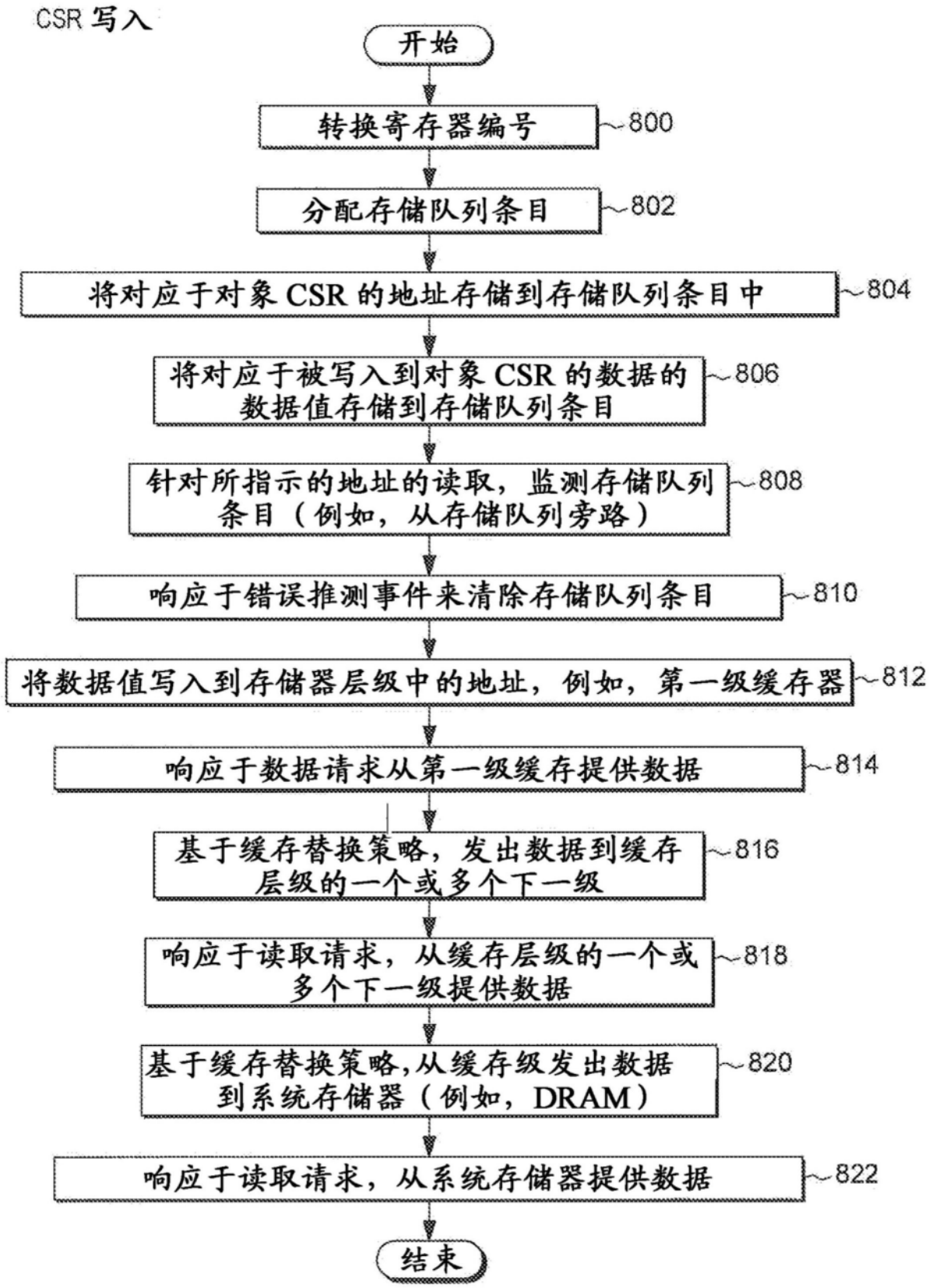

CSR 写入
开始
转换寄存器编号
800
分配存储队列条目
802
将对应于对象 CSR 的地址存储到存储队列条目中
804
将对应于被写入到对象 CSR 的数据的数据值存储到存储队列条目
806
针对所指示的地址的读取，监测存储队列条目（例如，从存储队列旁路）
808
响应于错误推测事件来清除存储队列条目
810
将数据值写入到存储器层级中的地址，例如，第一级缓存器
812
响应于数据请求从第一级缓存提供数据
814
基于缓存替换策略，发出数据到缓存层级的一个或多个下一级
816
响应于读取请求，从缓存层级的一个或多个下一级提供数据
818
基于缓存替换策略，从缓存级发出数据到系统存储器（例如，DRAM）
820
响应于读取请求，从系统存储器提供数据
822
结束

图8

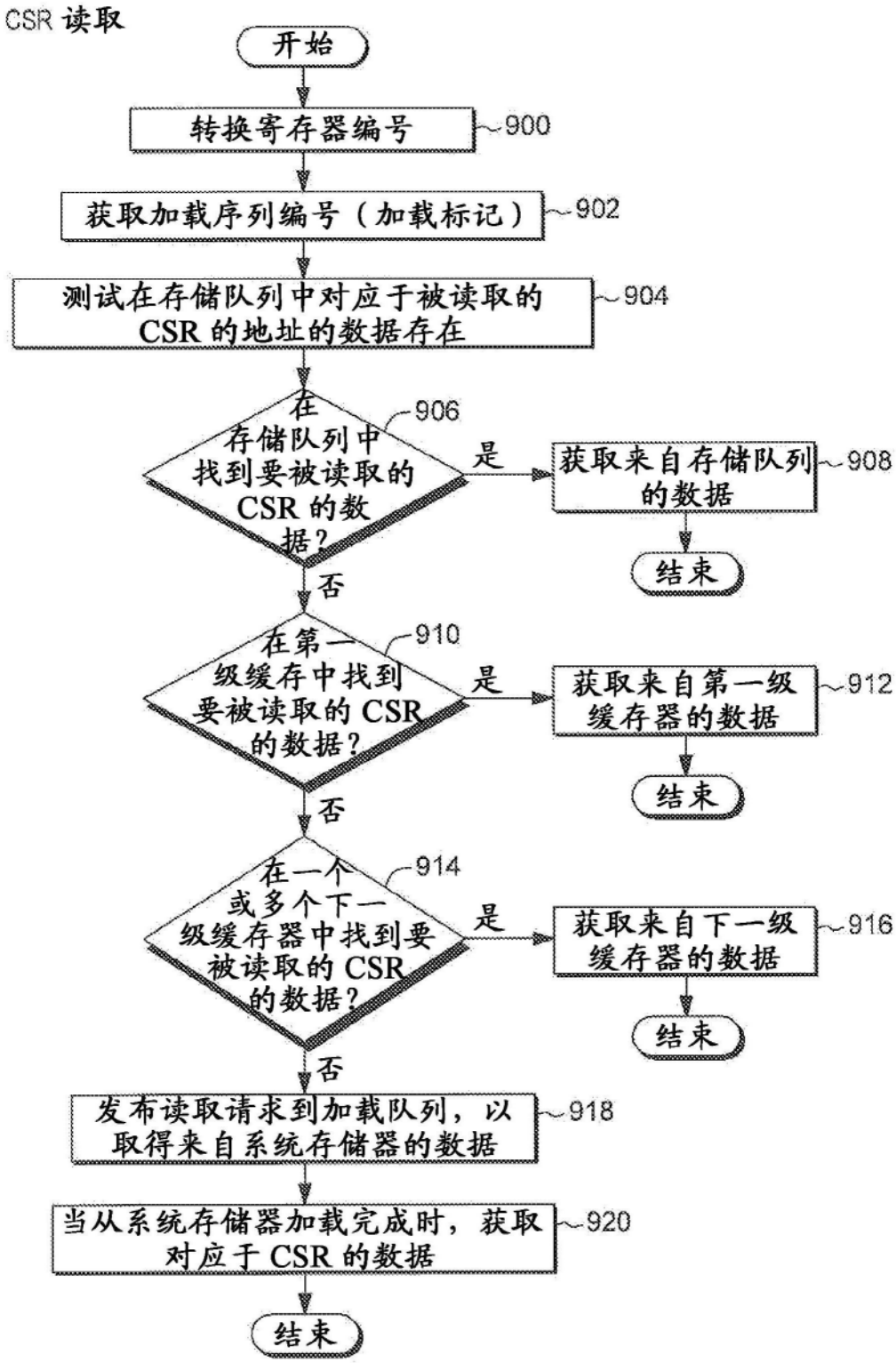

CSR 读取
开始
转换寄存器编号
900
获取加载序列编号（加载标记）
902
测试在存储队列中对应于被读取的CSR 的地址的数据存在
904
在存储队列中找到要被读取的CSR 的数据？
906
是
获取来自存储队列的数据
908
结束
否
在第一级缓存中找到要被读取的 CSR 的数据？
910
是
获取来自第一级缓存器的数据
912
结束
否
在一个或多个下一级缓存器中找到要被读取的 CSR 的数据？
914
是
获取来自下一级缓存器的数据
916
结束
否
发布读取请求到加载队列，以取得来自系统存储器的数据
918
当从系统存储器加载完成时，获取对应于 CSR 的数据
920
结束

图9

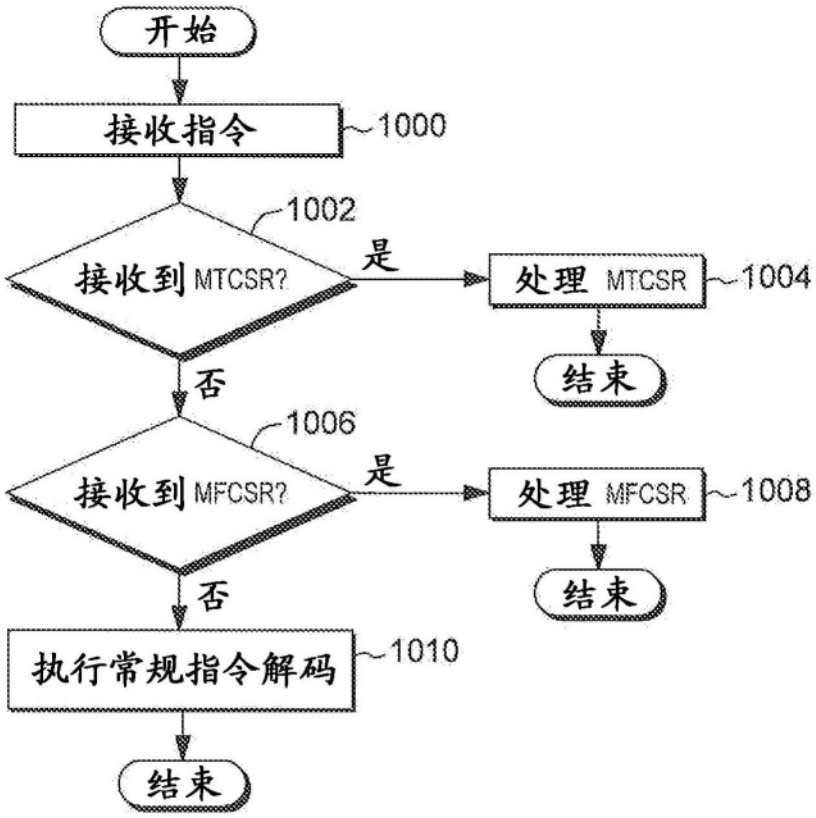

开始
接收指令
1000
1002
接收到 MTCSR?
是
处理 MTCSR
1004
结束
否
1006
接收到 MFCSR?
是
处理 MFCSR
1008
结束
否
执行常规指令解码
1010
结束

图10

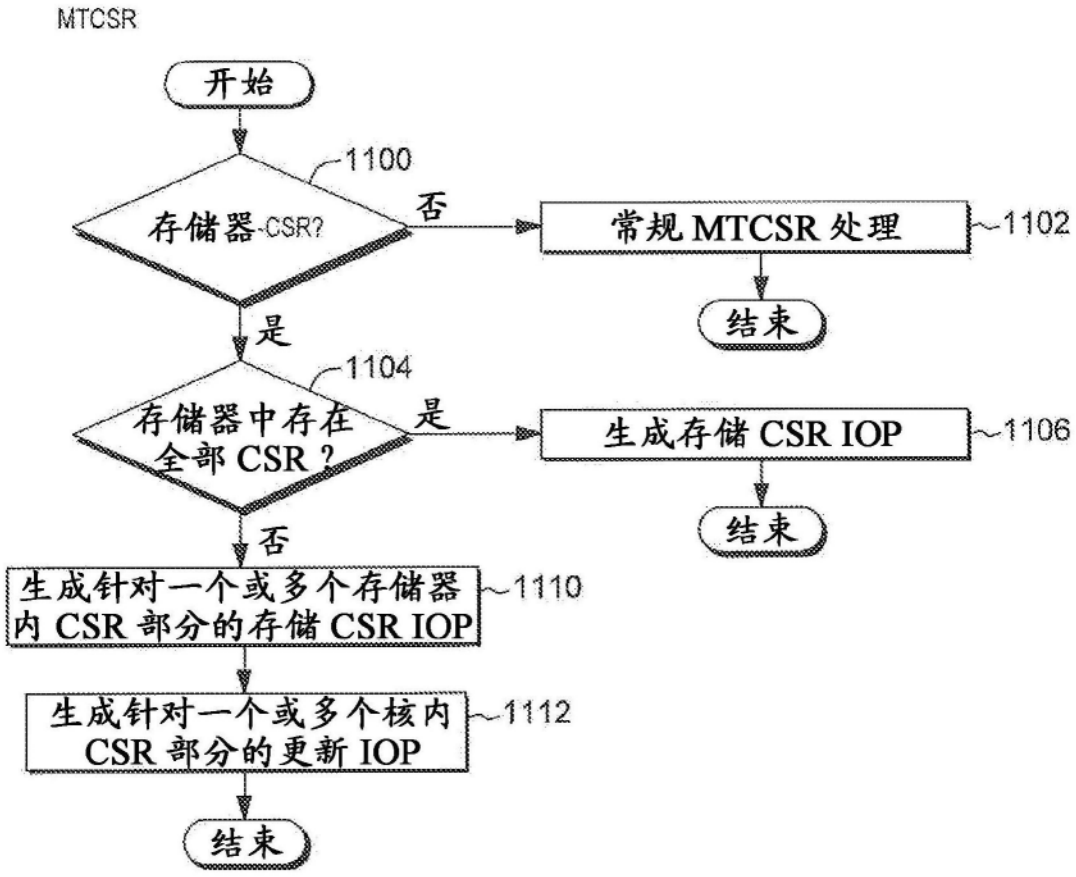

MTCSR
开始
1100
存储器-CSR?
否
常规 MTCSR 处理
1102
结束
是
1104
存储器中存在全部 CSR ?
是
生成存储 CSR IOP
1106
结束
否
生成针对一个或多个存储器内 CSR 部分的存储 CSR IOP
1110
生成针对一个或多个核内 CSR 部分的更新 IOP
1112
结束

图11

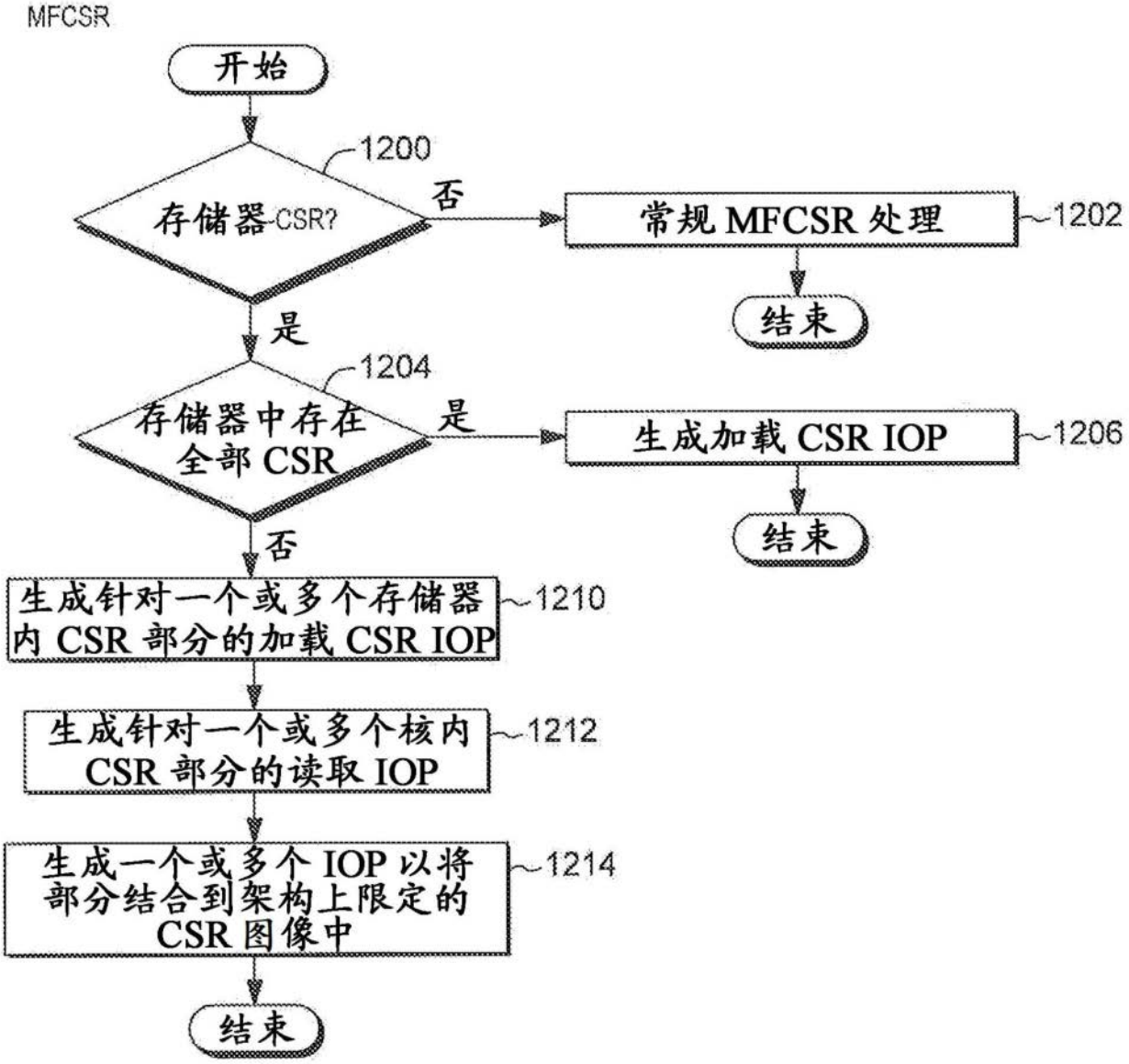

MFCSR
开始
1200
存储器-CSR?
否
常规 MFCSR 处理
1202
结束
是
1204
存储器中存在全部 CSR
是
生成加载 CSR IOP
1206
结束
否
生成针对一个或多个存储器内 CSR 部分的加载 CSR IOP
1210
生成针对一个或多个核内 CSR 部分的读取 IOP
1212
生成一个或多个 IOP 以将部分结合到架构上限定的 CSR 图像中
1214
结束

图12

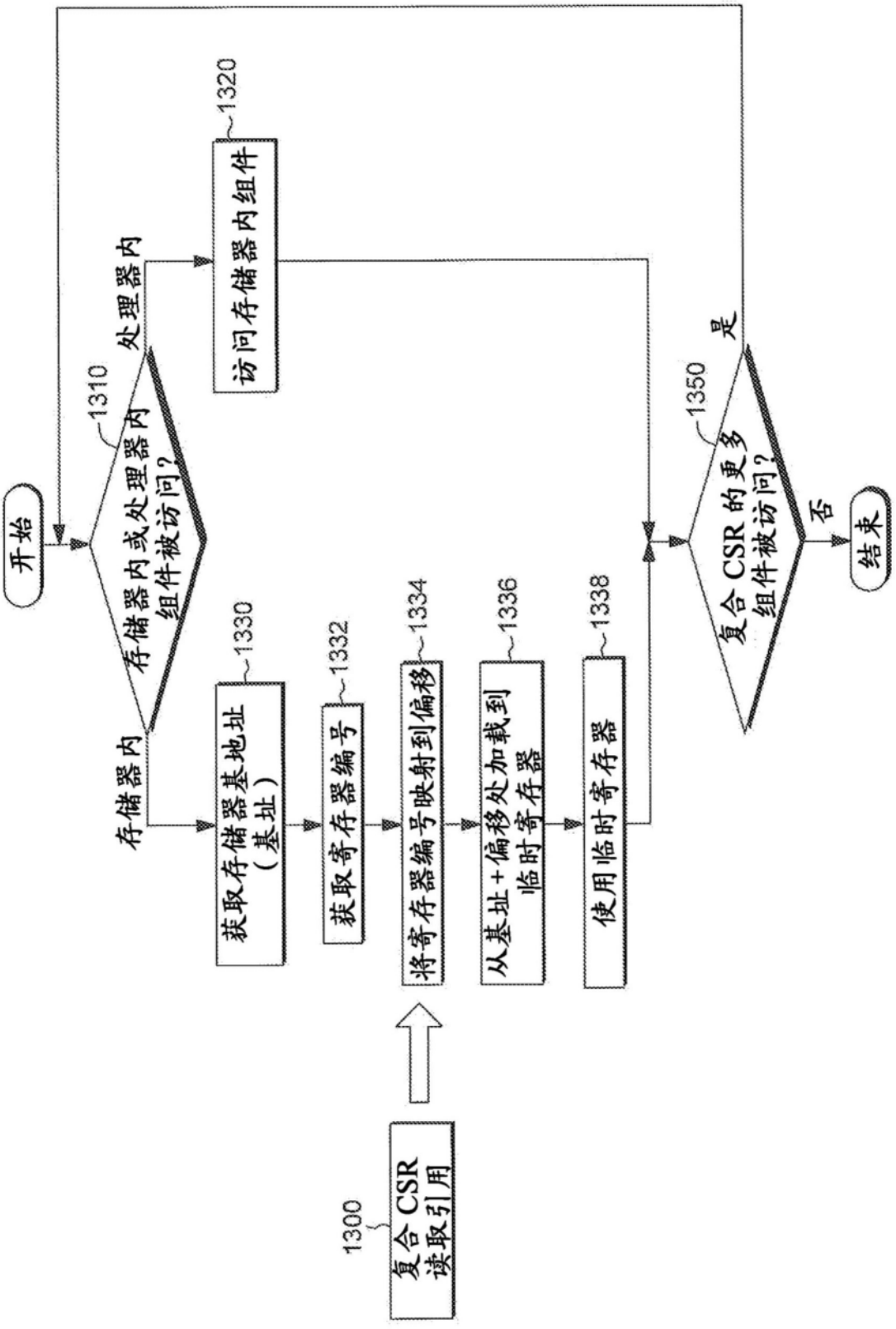

开始
1310
存储器内或处理器内组件被访问?
存储器内
处理器内
1320
访问存储器内组件
1330
获取存储器基地址(基址)
1332
获取寄存器编号
1334
将寄存器编号映射到偏移
1336
从基址+偏移处加载到临时寄存器
1338
使用临时寄存器
1350
复合CSR的更多组件被访问?
是
否
结束
1300
复合CSR读取引用

图13A

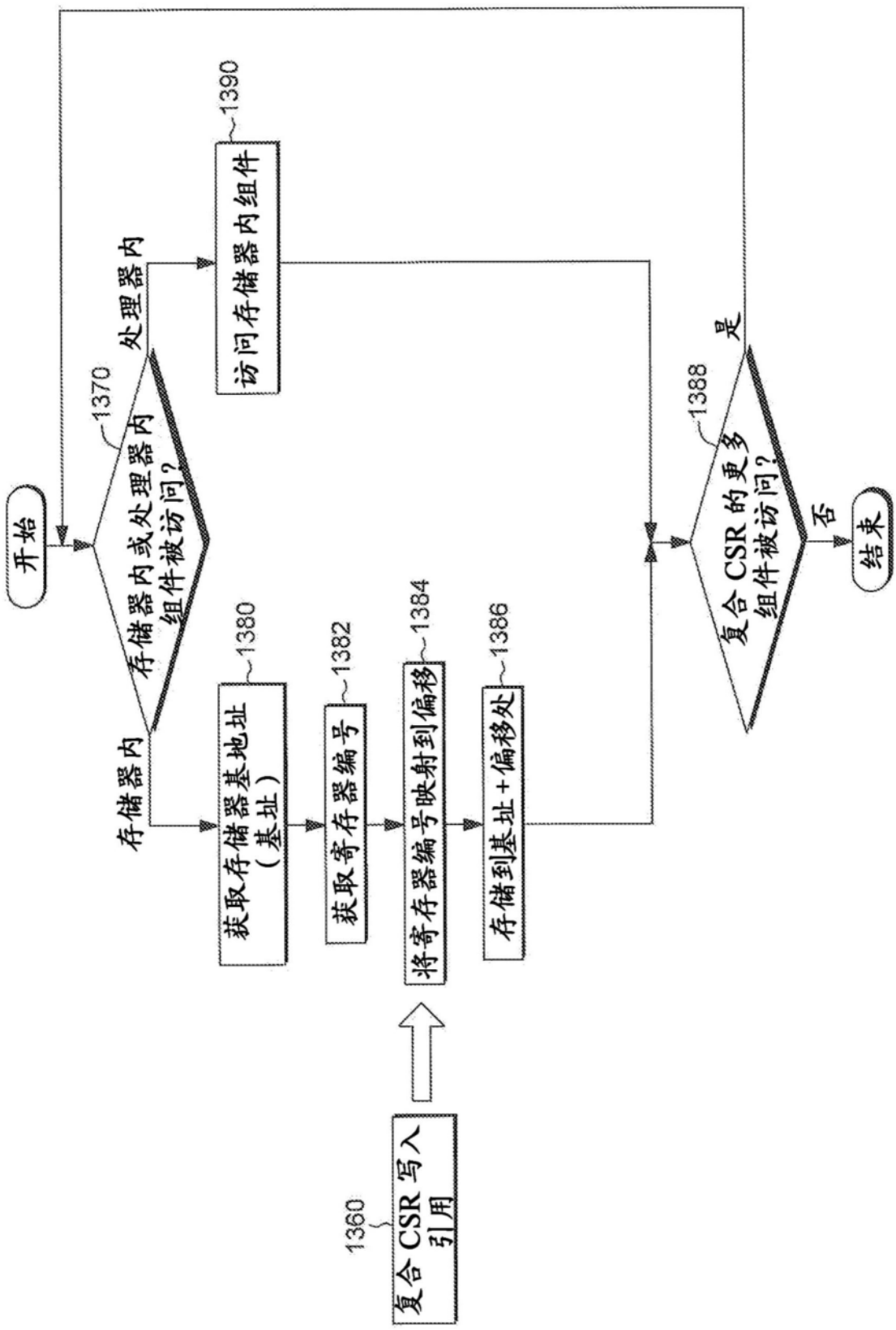
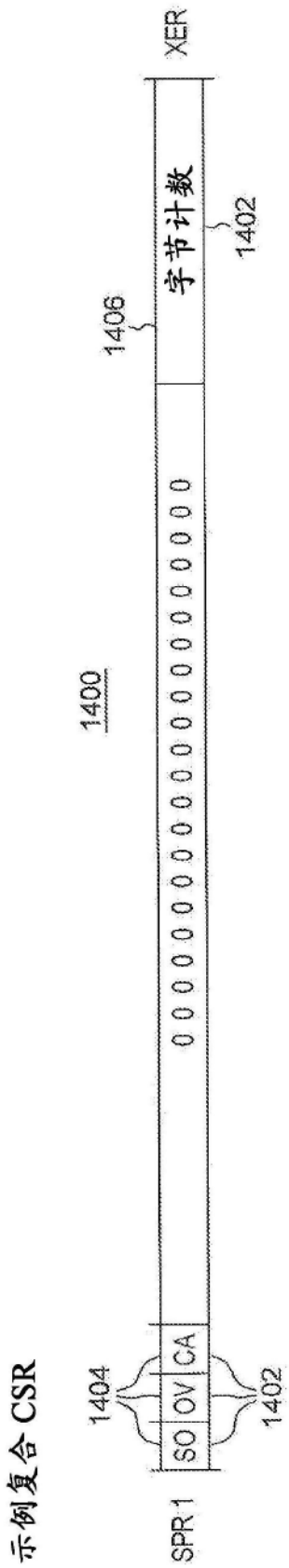

开始
存储器内或处理器内组件被访问?
1370
处理器内
存储器内
访问存储器内组件
1390
获取存储器基地址(基址)
1380
获取寄存器编号
1382
将寄存器编号映射到偏移
1384
存储到基址+偏移处
1386
复合CSR的更多组件被访问?
1388
是
否
结束
复合CSR写入引用
1360

图13B


XER
字节计数
1402
1406
1400
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1404
SO
OV
CA
1402
SPR 1

示例复合 CSR

图14

1500

	-
13	AMR
	-
128	TFHAR4
129	TFIAR4
130	TEXASR4
131	TEXASRU4
	-
136	CTRL
	-
256	VRSAVE
	-
259	SPRG3
	-
268	TB2
269	TBU2
	-
768	SIER
769	MMCR2
770	MMCRA
771	PMC1
772	PMC2
773	PMC3
774	PMC4
775	PMC5
776	PMC6

图15A

1500

	-
779	MMCR0
780	SIAR
781	SDAR
782	MMCR1
	-
800	BESCRS
801	BESCRSU
802	BESCRR
803	BESCRRU
804	EBBHR
805	EBBRR
806	BESCR
	-
808	reserved3
809	reserved3
810	reserved3
811	reserved3
	-
813	LMRR
814	LMSER
815	TAR
	-
896	PPR10
	-
898	PPR32

图15B

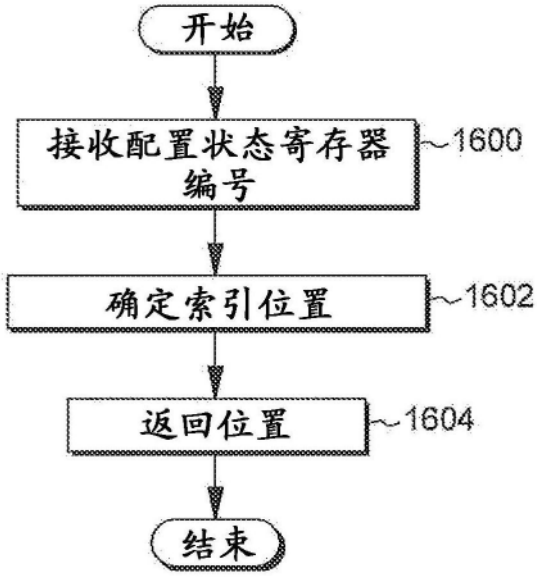


图16

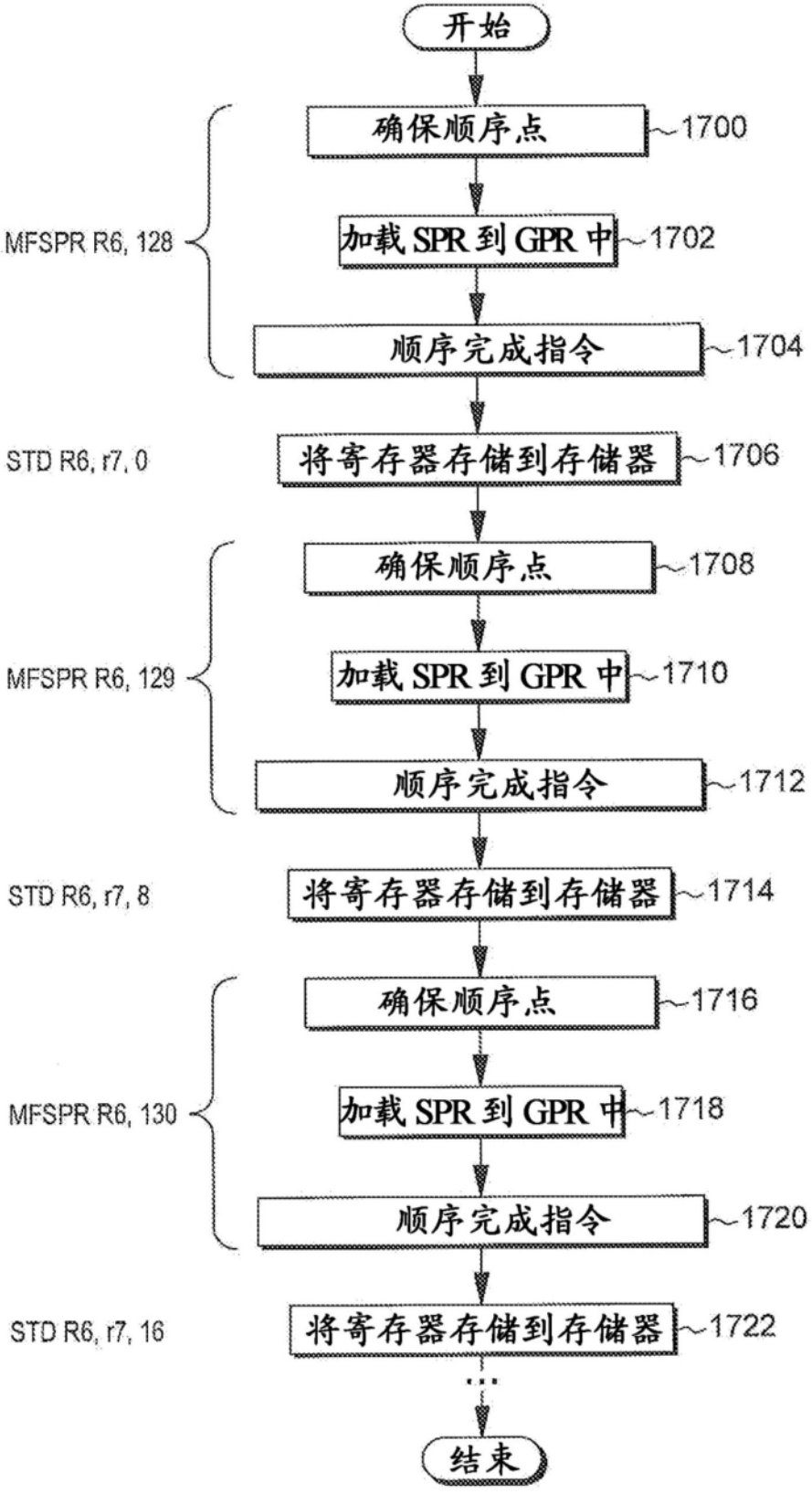

开始
确保顺序点
1700
MFSPR R6, 128
加载 SPR 到 GPR 中
1702
顺序完成指令
1704
STD R6, r7, 0
将寄存器存储到存储器
1706
确保顺序点
1708
MFSPR R6, 129
加载 SPR 到 GPR 中
1710
顺序完成指令
1712
STD R6, r7, 8
将寄存器存储到存储器
1714
确保顺序点
1716
MFSPR R6, 130
加载 SPR 到 GPR 中
1718
顺序完成指令
1720
STD R6, r7, 16
将寄存器存储到存储器
1722
...
结束

图17A

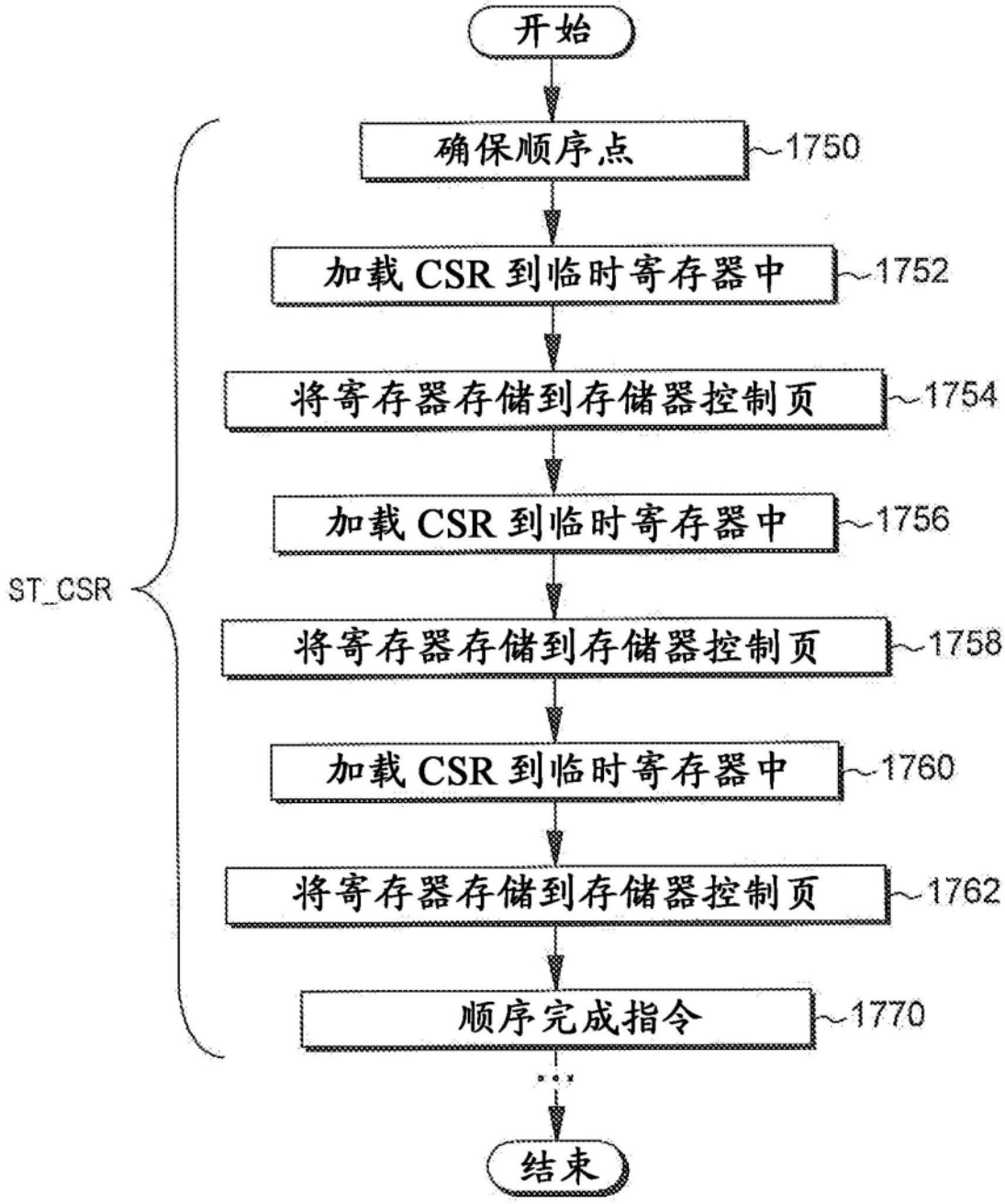

开始
确保顺序点
1750
加载 CSR 到临时寄存器中
1752
将寄存器存储到存储器控制页
1754
加载 CSR 到临时寄存器中
1756
ST_CSR
将寄存器存储到存储器控制页
1758
加载 CSR 到临时寄存器中
1760
将寄存器存储到存储器控制页
1762
顺序完成指令
1770
结束

图17B

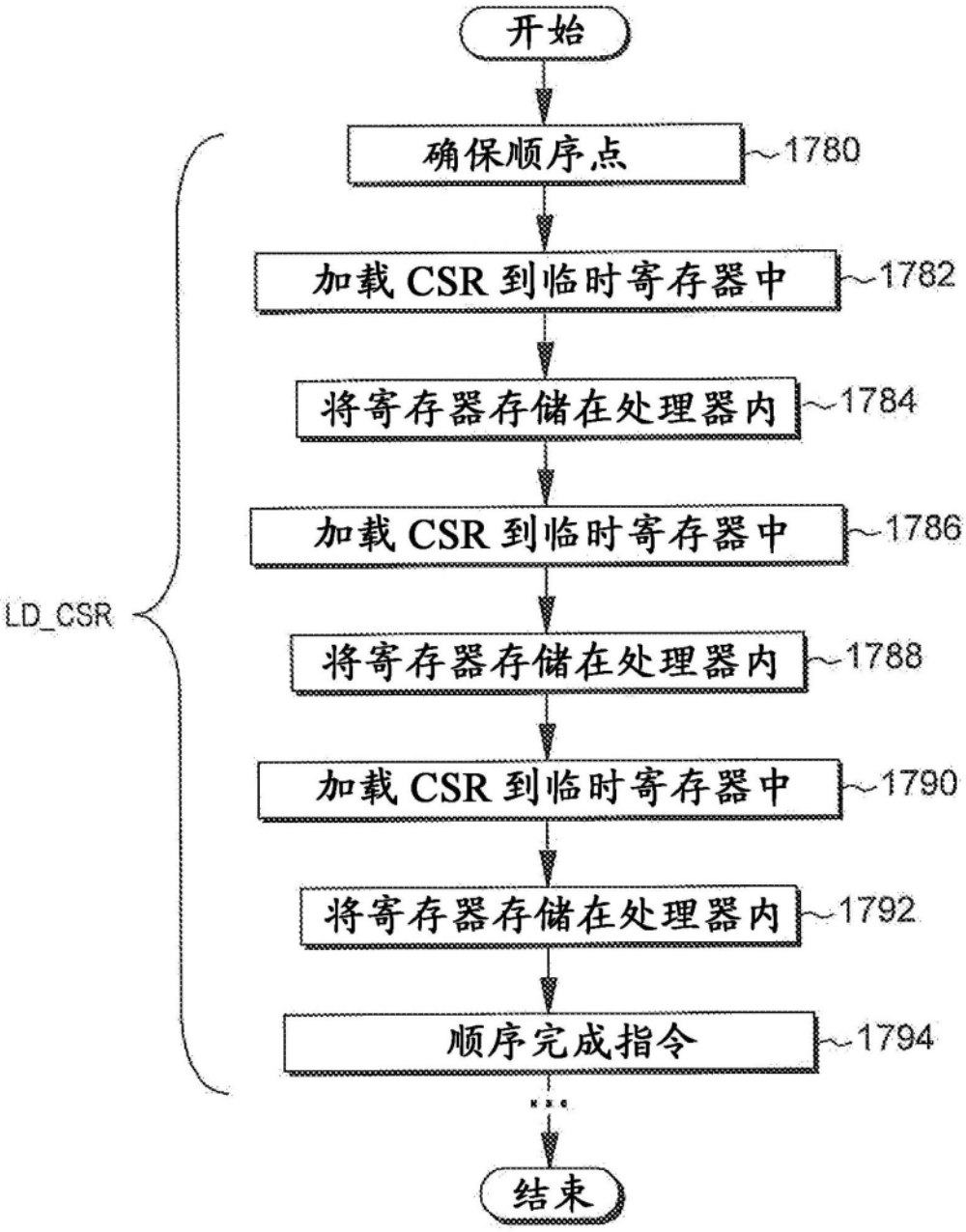

开始
确保顺序点
1780
加载 CSR 到临时寄存器中
1782
将寄存器存储在处理器内
1784
加载 CSR 到临时寄存器中
1786
将寄存器存储在处理器内
1788
加载 CSR 到临时寄存器中
1790
将寄存器存储在处理器内
1792
顺序完成指令
1794
结束
LD_CSR

图17C

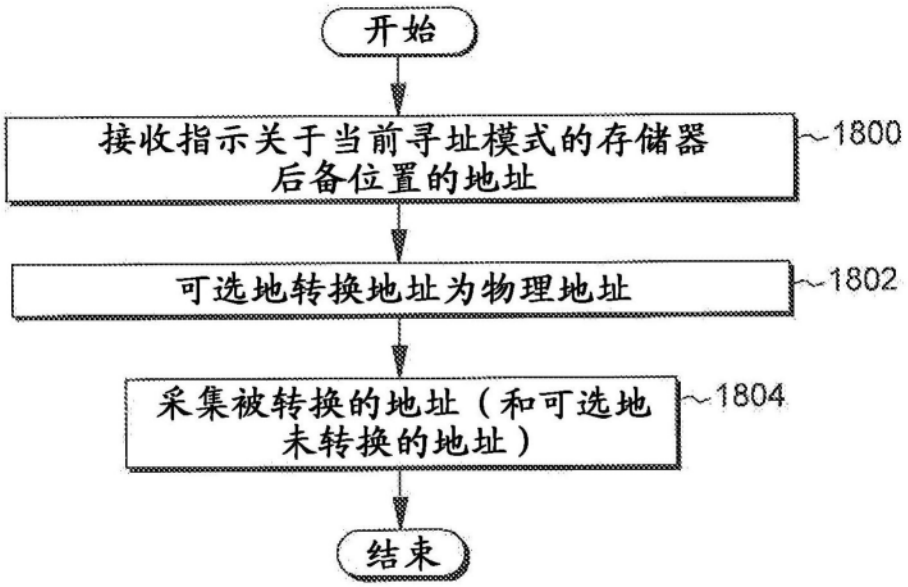

开始
接收指示关于当前寻址模式的存储器后备位置的地址
1800
可选地转换地址为物理地址
1802
采集被转换的地址（和可选地未转换的地址）
1804
结束

图18A

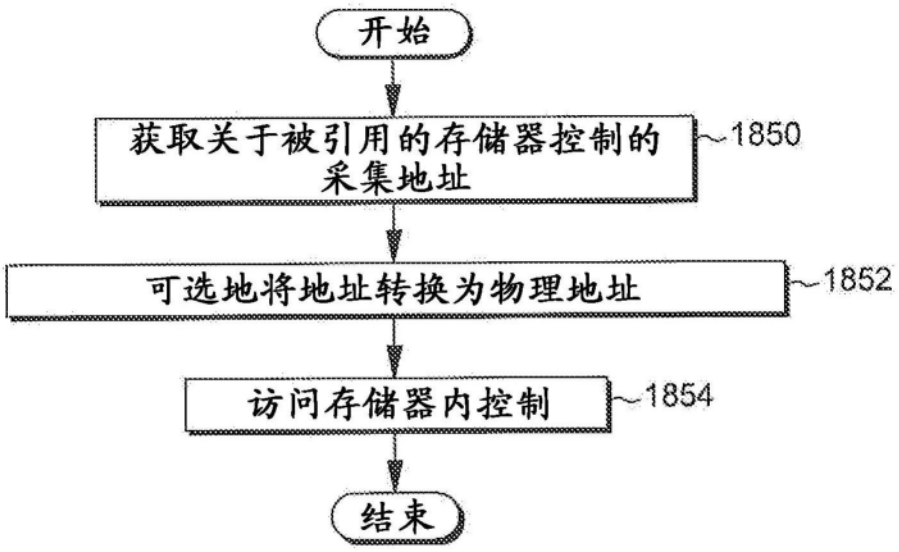

开始
获取关于被引用的存储器控制的采集地址
1850
可选地将地址转换为物理地址
1852
访问存储器内控制
1854
结束

图18B

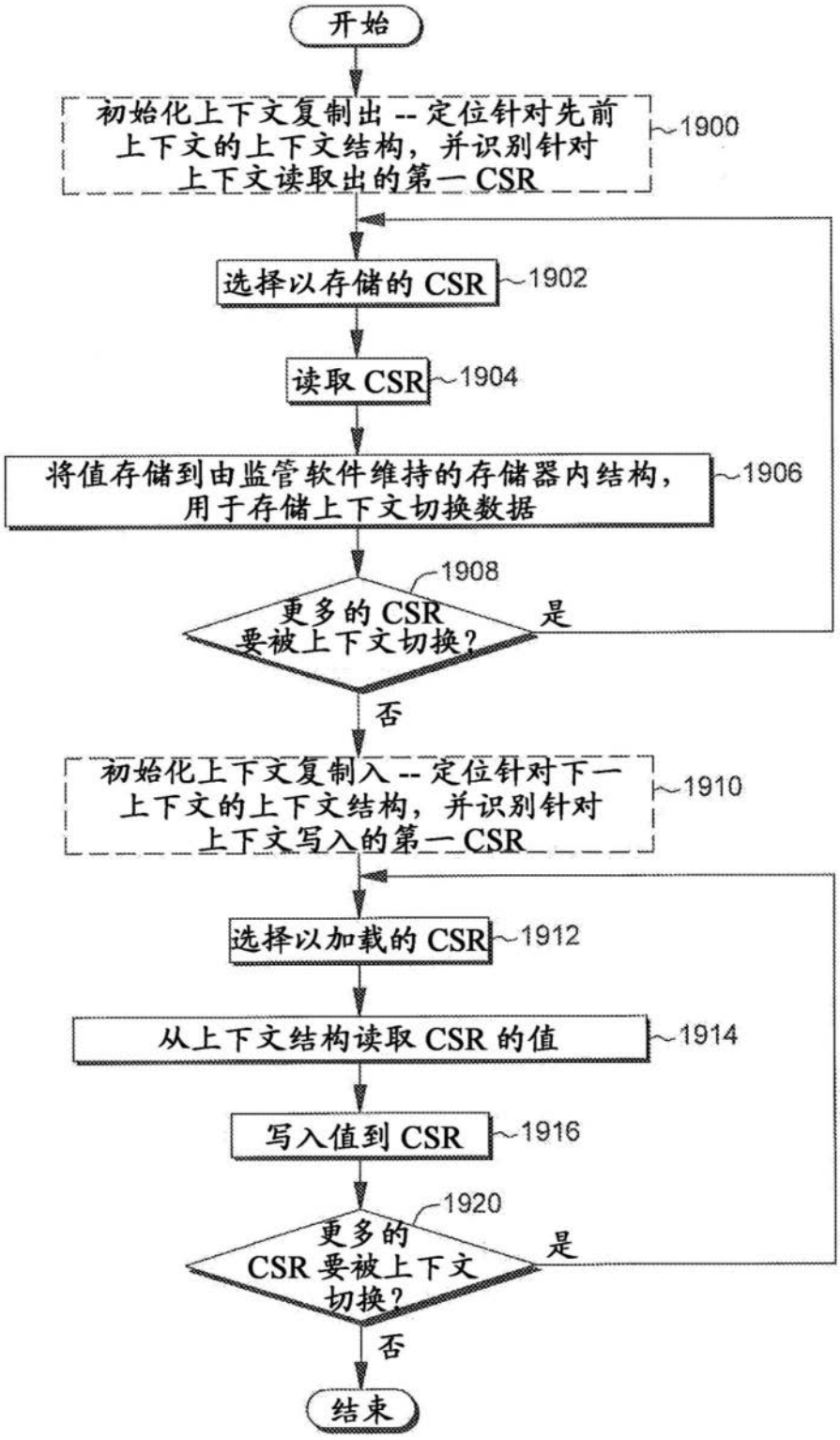

开始
初始化上下文复制出 -- 定位针对先前上下文的上下文结构，并识别针对上下文读取出的第一 CSR
1900
选择以存储的 CSR
1902
读取 CSR
1904
将值存储到由监管软件维持的存储器内结构，用于存储上下文切换数据
1906
1908
更多的 CSR 要被上下文切换？
是
否
初始化上下文复制入 -- 定位针对下一上下文的上下文结构，并识别针对上下文写入的第一 CSR
1910
选择以加载的 CSR
1912
从上下文结构读取 CSR 的值
1914
写入值到 CSR
1916
1920
更多的 CSR 要被上下文切换？
是
否
结束

图19A

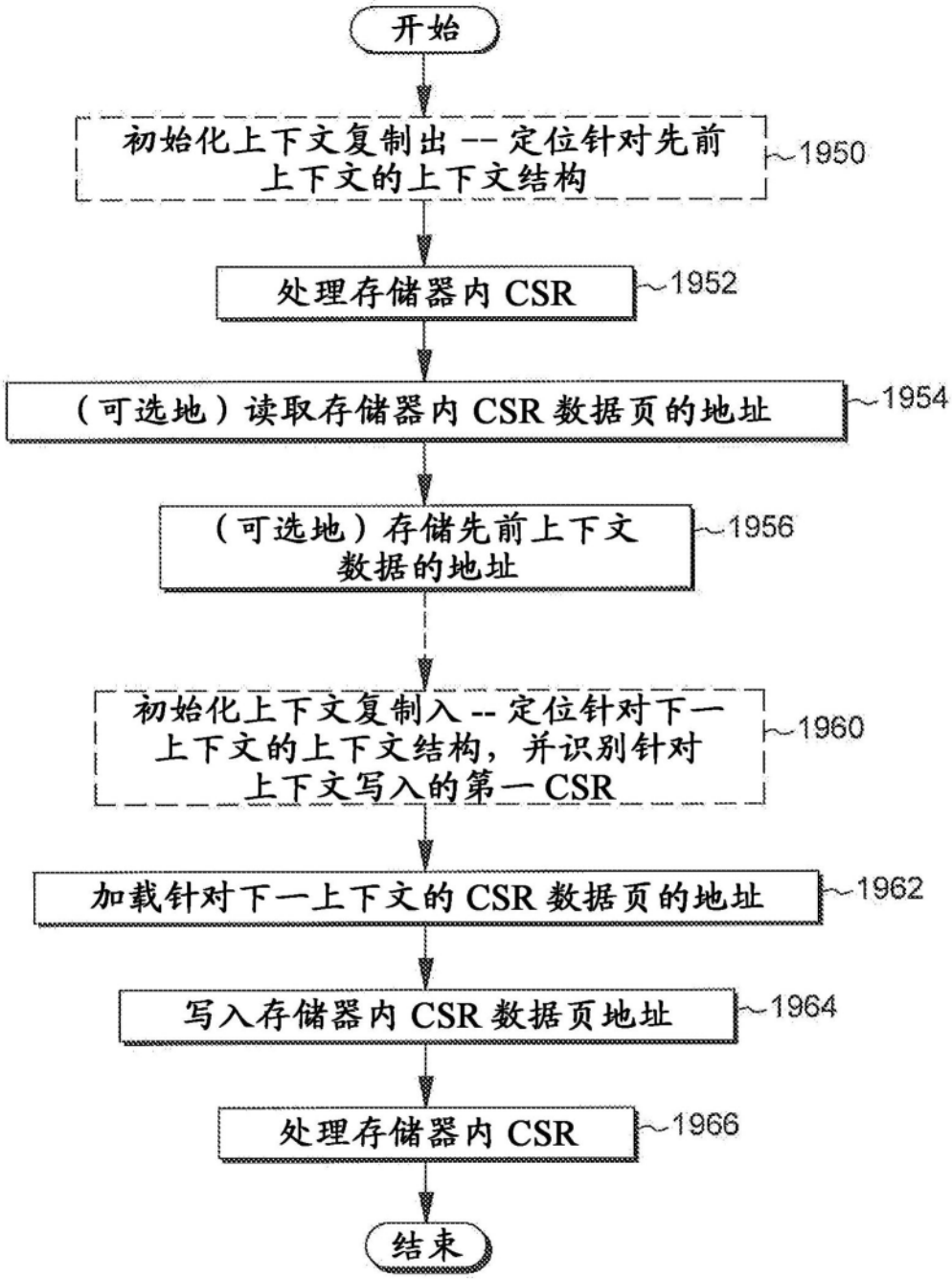

开始
初始化上下文复制出 -- 定位针对先前上下文的上下文结构
1950
处理存储器内 CSR
1952
（可选地）读取存储器内 CSR 数据页的地址
1954
（可选地）存储先前上下文数据的地址
1956
初始化上下文复制入 -- 定位针对下一上下文的上下文结构，并识别针对上下文写入的第一 CSR
1960
加载针对下一上下文的 CSR 数据页的地址
1962
写入存储器内 CSR 数据页地址
1964
处理存储器内 CSR
1966
结束

图19B

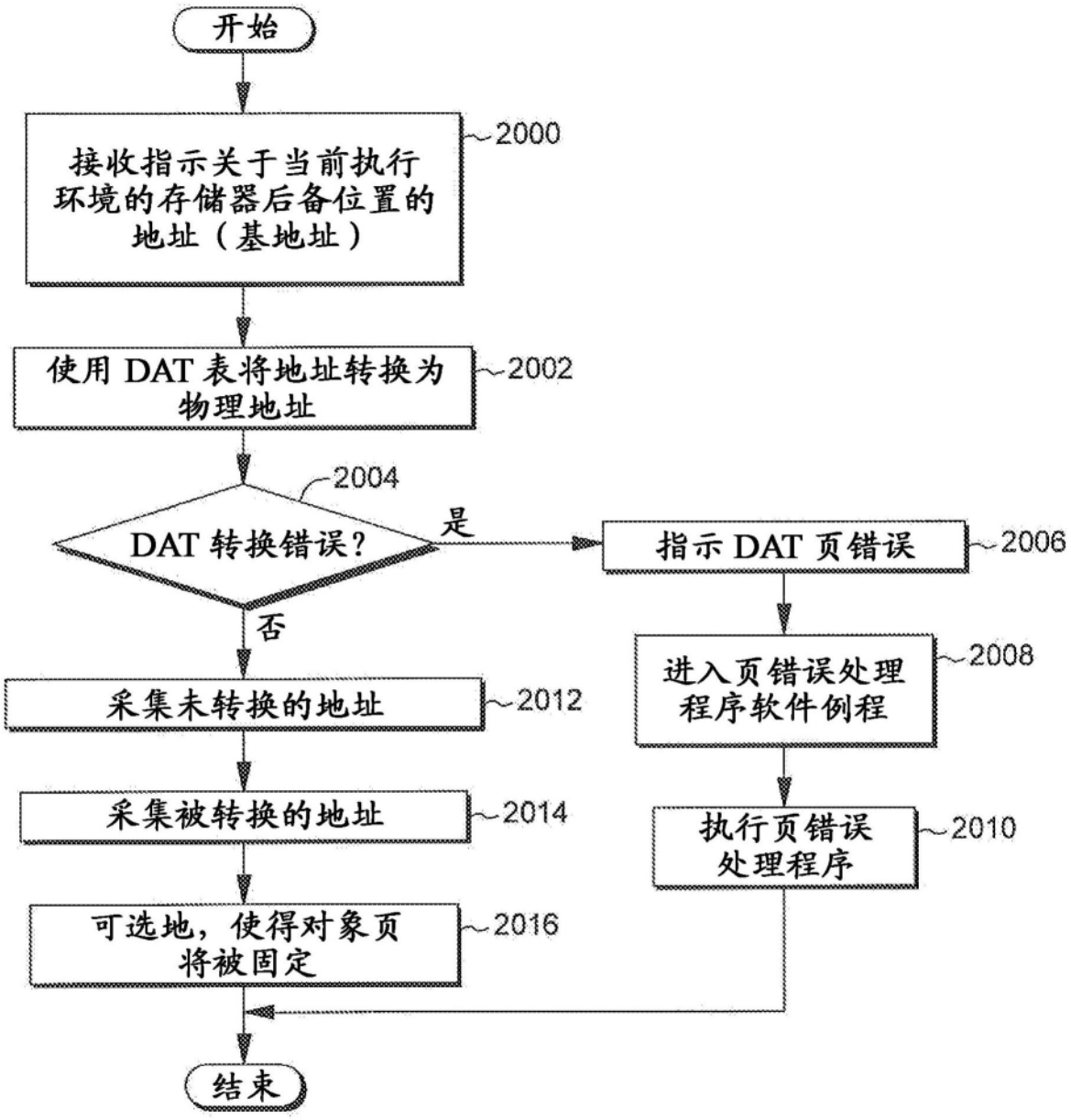

开始
接收指示关于当前执行环境的存储器后备位置的地址（基地址）
2000
使用 DAT 表将地址转换为物理地址
2002
2004
DAT 转换错误？
是
否
指示 DAT 页错误
2006
进入页错误处理程序软件例程
2008
执行页错误处理程序
2010
采集未转换的地址
2012
采集被转换的地址
2014
可选地，使得对象页将被固定
2016
结束

图20

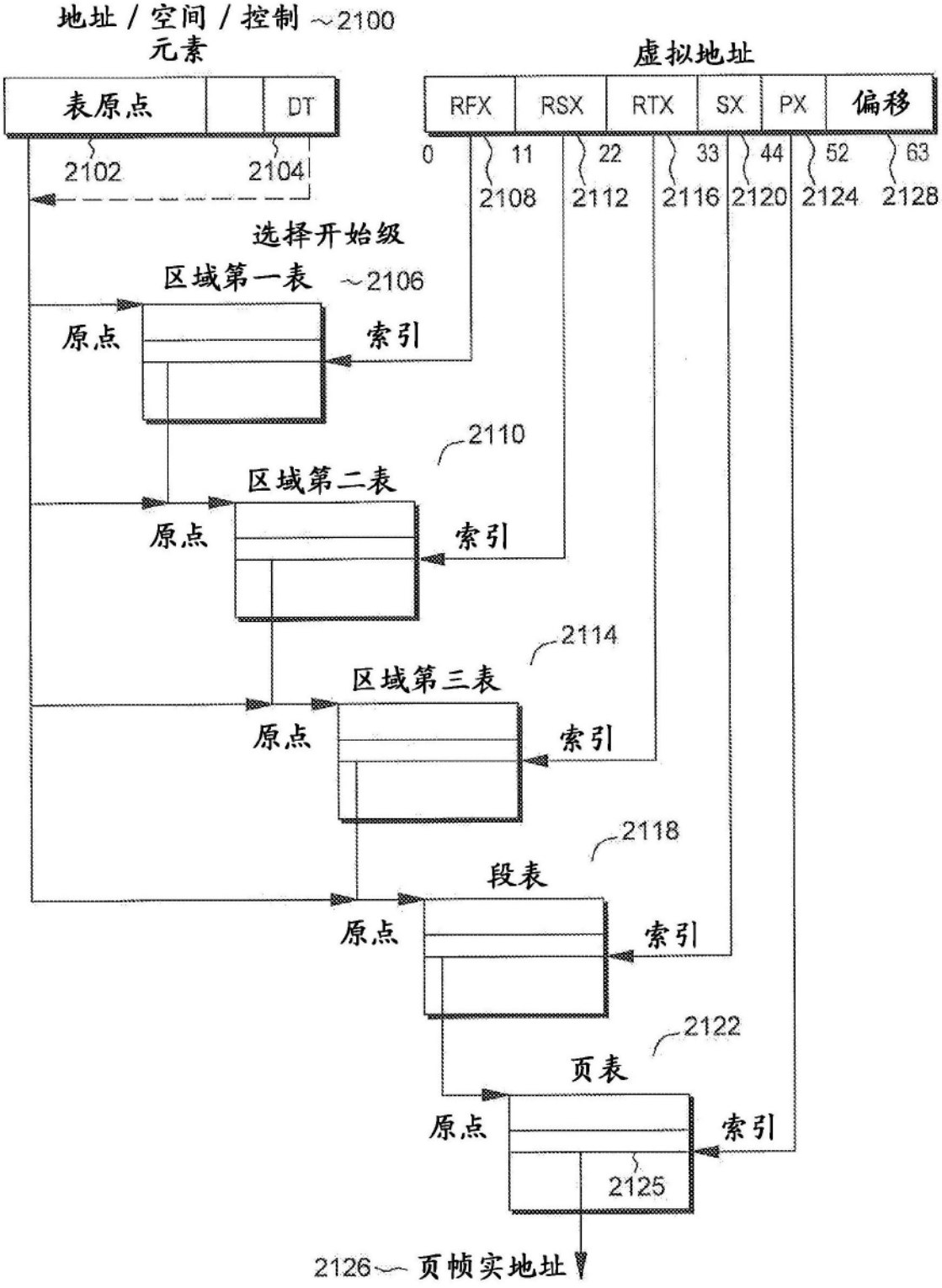

地址 / 空间 / 控制元素
2100
表原点
DT
2102
2104
虚拟地址
RFX
RSX
RTX
SX
PX
偏移
0
11
22
33
44
52
63
2108
2112
2116
2120
2124
2128
选择开始级
区域第一表
2106
原点
索引
2110
区域第二表
原点
索引
2114
区域第三表
原点
索引
2118
段表
原点
索引
2122
页表
原点
索引
2125
2126
页帧实地址

图21A

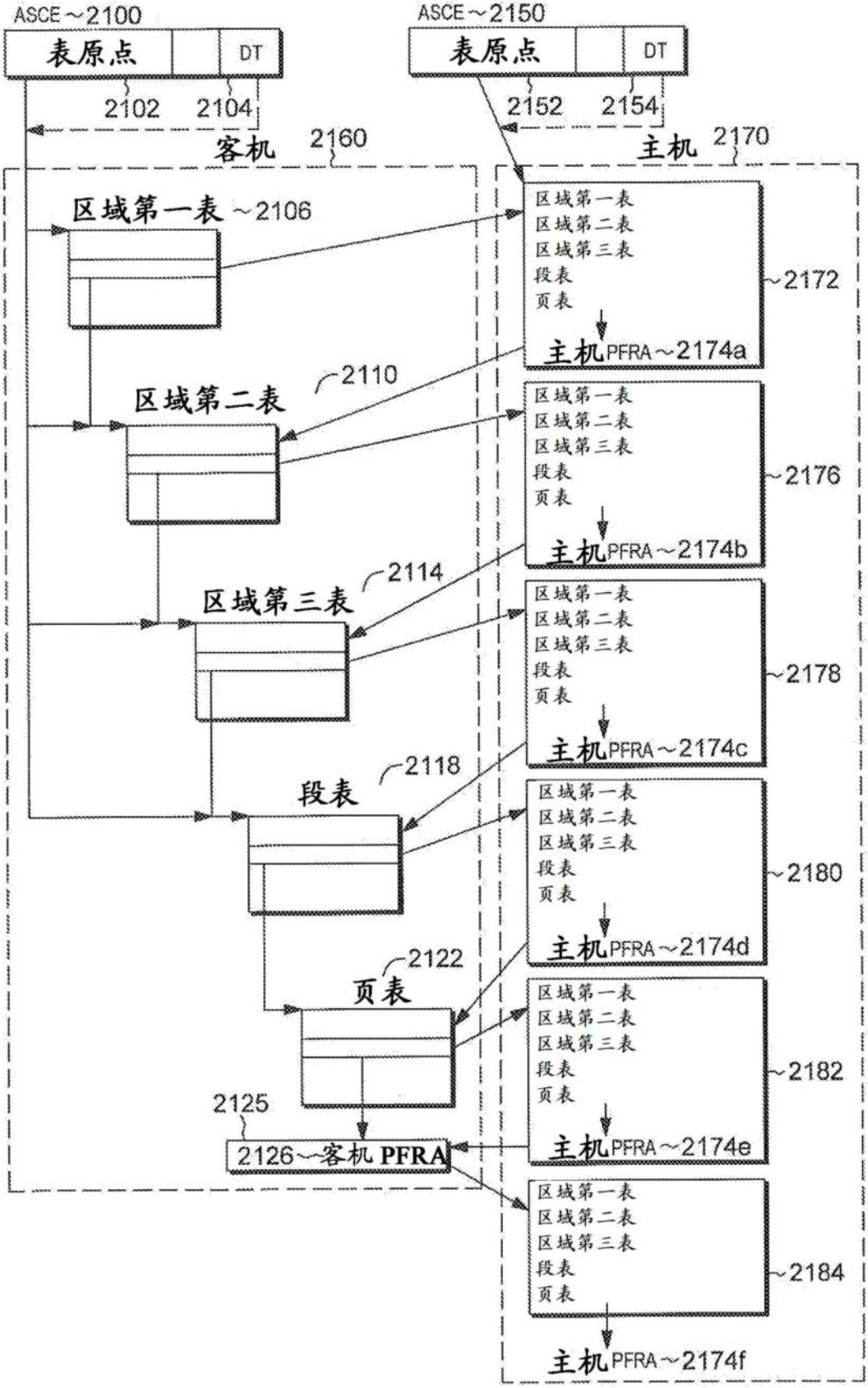

ASCE~2100
表原点
DT
2102
2104
客机
2160
区域第一表 ~2106
区域第二表
2110
区域第三表
2114
段表
2118
页表
2122
2125
2126~客机 PFRA
ASCE~2150
表原点
DT
2152
2154
主机
2170
区域第一表
区域第二表
区域第三表
段表
页表
~2172
主机PFRA~2174a
~2176
主机PFRA~2174b
~2178
主机PFRA~2174c
~2180
主机PFRA~2174d
~2182
主机PFRA~2174e
~2184
主机PFRA~2174f

图21B

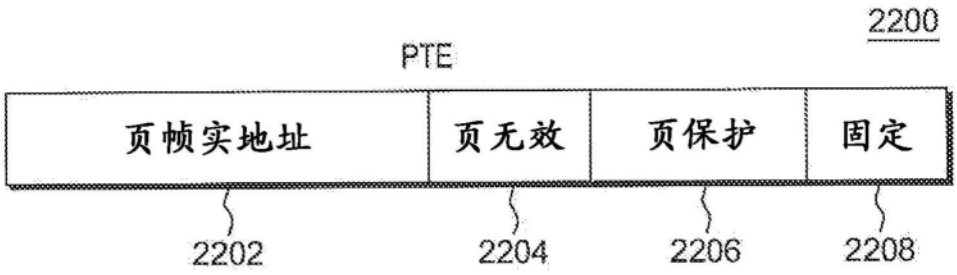


图22

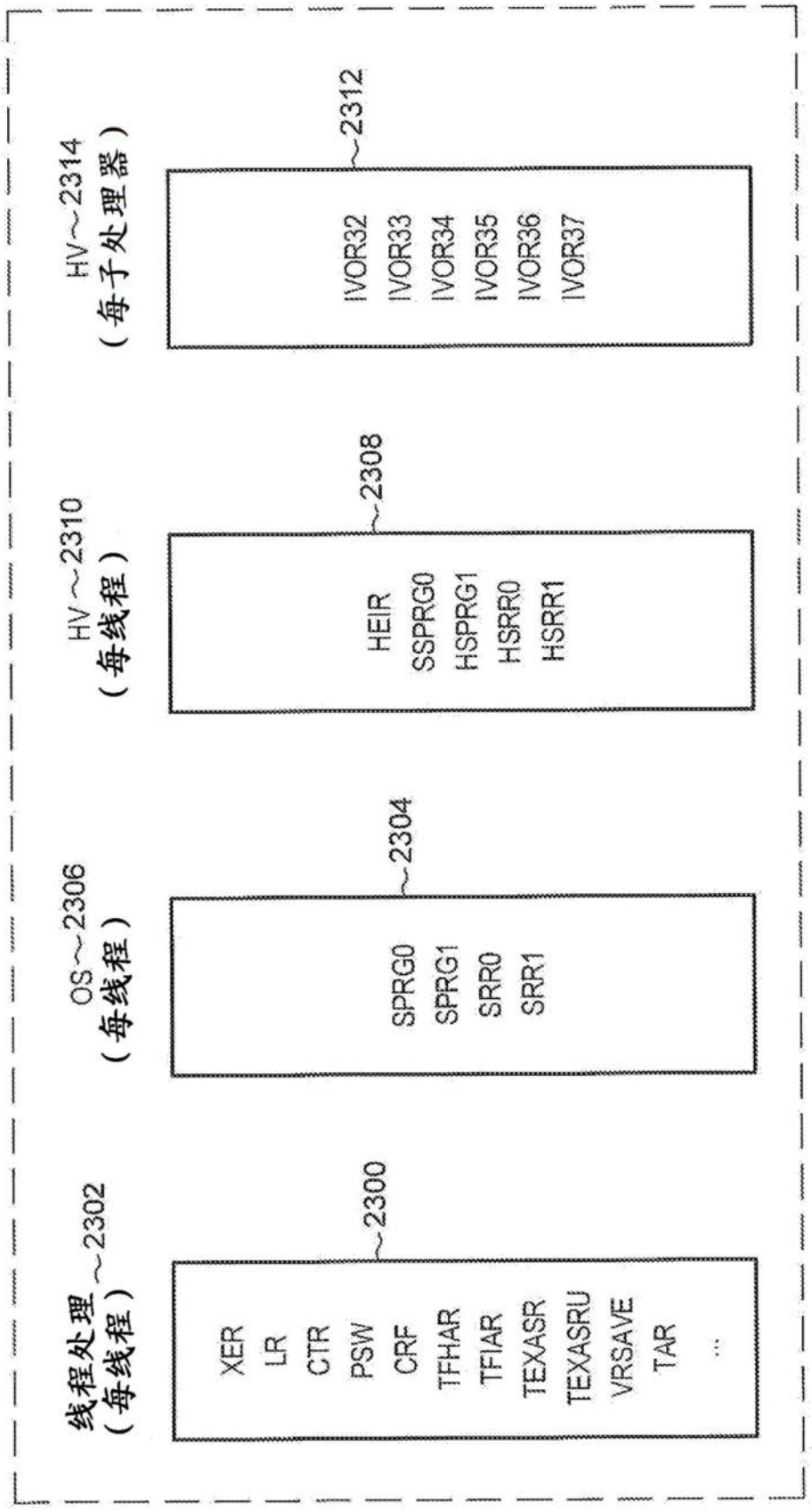

线程处理（每线程）
~2302
~2300
XER
LR
CTR
PSW
CRF
TFHAR
TFIAR
TEXASR
TEXASRU
VRSAVE
TAR
...
OS~2306（每线程）
~2304
SPRG0
SPRG1
SRR0
SRR1
HV~2310（每线程）
~2308
HEIR
SSPRG0
HSPRG1
HSRR0
HSRR1
HV~2314（每子处理器）
~2312
IVOR32
IVOR33
IVOR34
IVOR35
IVOR36
IVOR37

图23

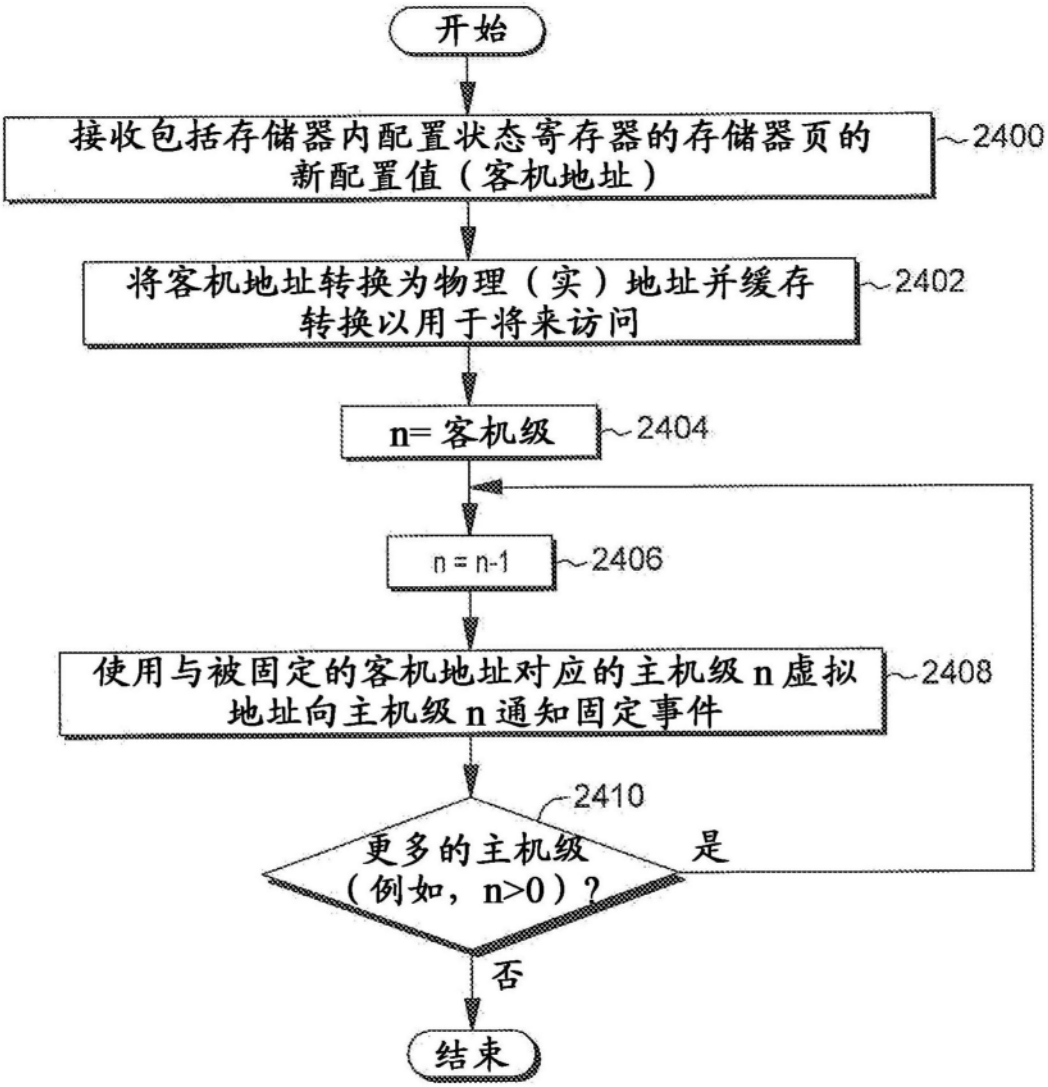

开始
接收包括存储器内配置状态寄存器的存储器页的新配置值（客机地址）
2400
将客机地址转换为物理（实）地址并缓存转换以用于将来访问
2402
n= 客机级
2404
n = n-1
2406
使用与被固定的客机地址对应的主机级 n 虚拟地址向主机级 n 通知固定事件
2408
2410
更多的主机级（例如，n>0）？
是
否
结束

图24

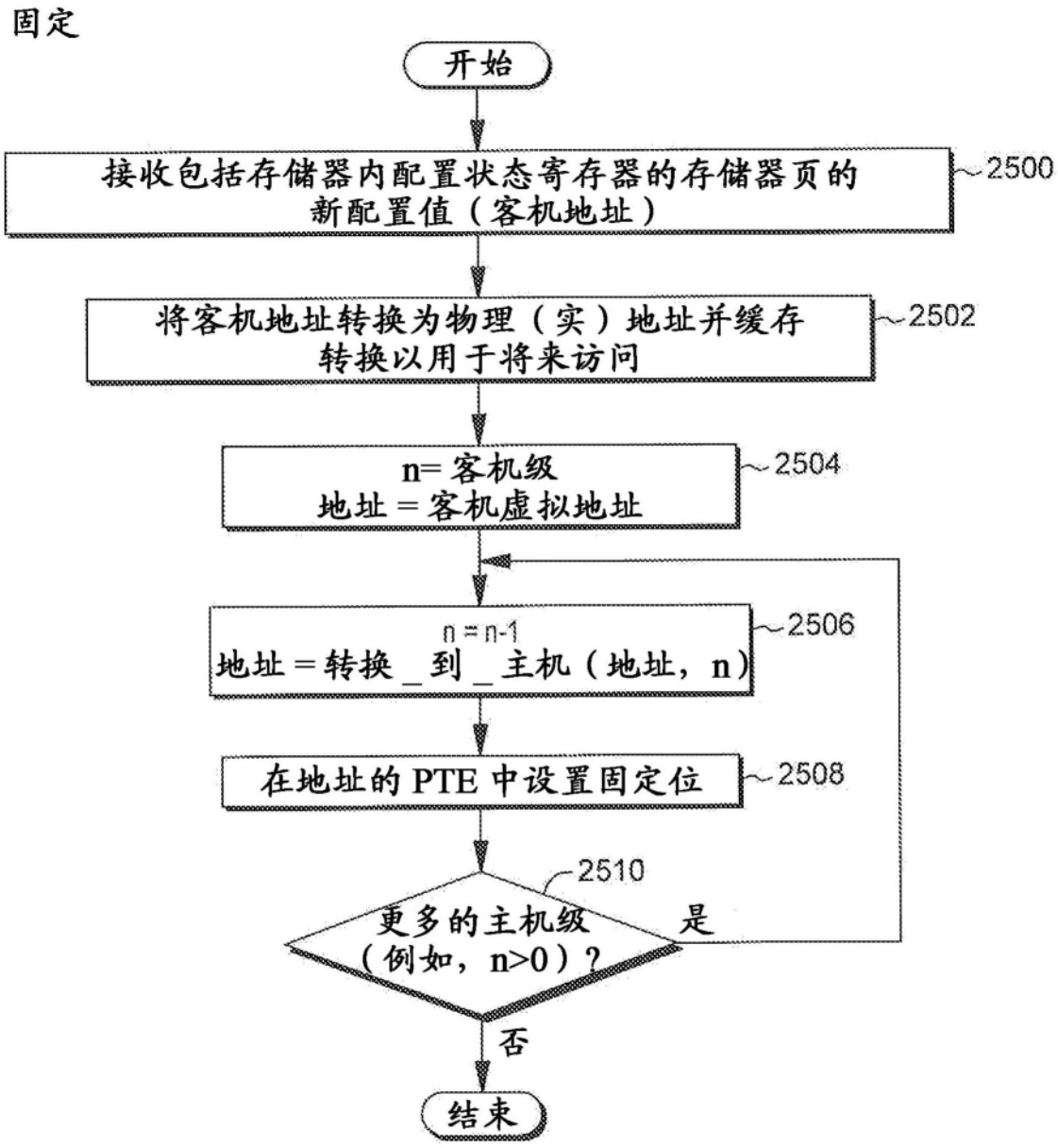

固定
开始
接收包括存储器内配置状态寄存器的存储器页的新配置值（客机地址）
2500
将客机地址转换为物理（实）地址并缓存转换以用于将来访问
2502
n= 客机级
地址 = 客机虚拟地址
2504
n = n-1
地址 = 转换 _ 到 _ 主机（地址，n）
2506
在地址的 PTE 中设置固定位
2508
2510
更多的主机级（例如，n>0）？
是
否
结束

图25

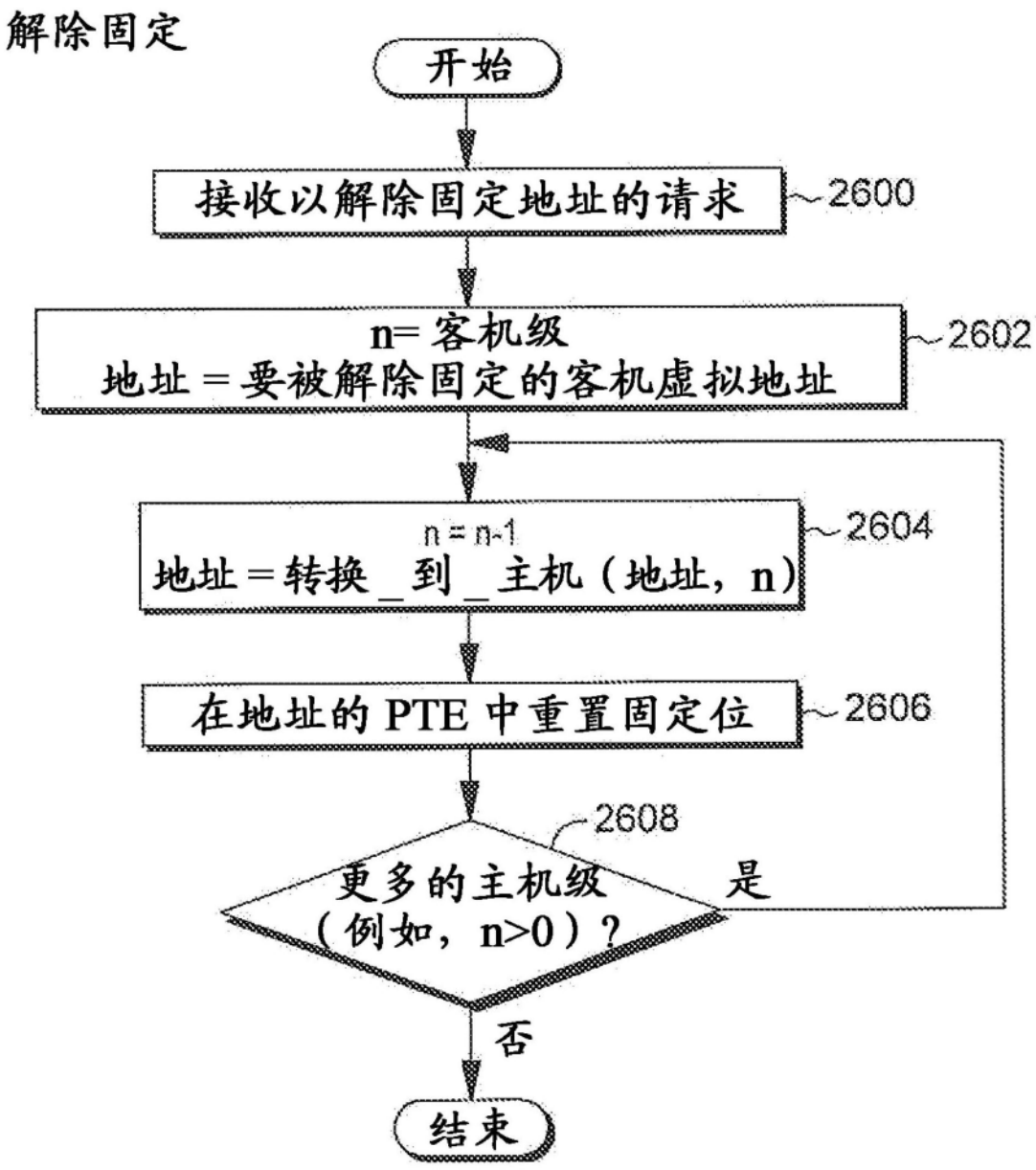

解除固定
开始
接收以解除固定地址的请求
2600
n= 客机级
地址 = 要被解除固定的客机虚拟地址
2602
n = n-1
地址 = 转换_到_主机（地址，n）
2604
在地址的 PTE 中重置固定位
2606
2608
更多的主机级
（例如，n>0）？
是
否
结束

图26

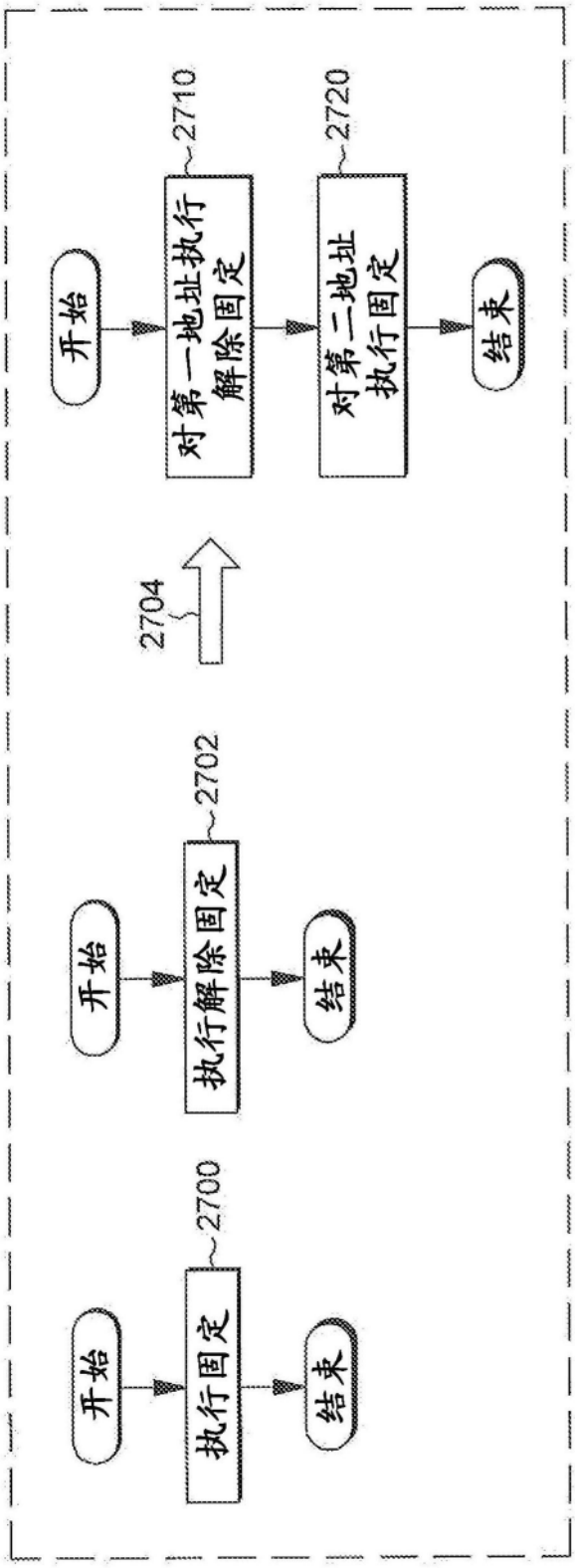

开始
执行固定
2700
结束
开始
执行解除固定
2702
结束
2704
开始
对第一地址执行解除固定
2710
对第二地址执行固定
2720
结束

图27

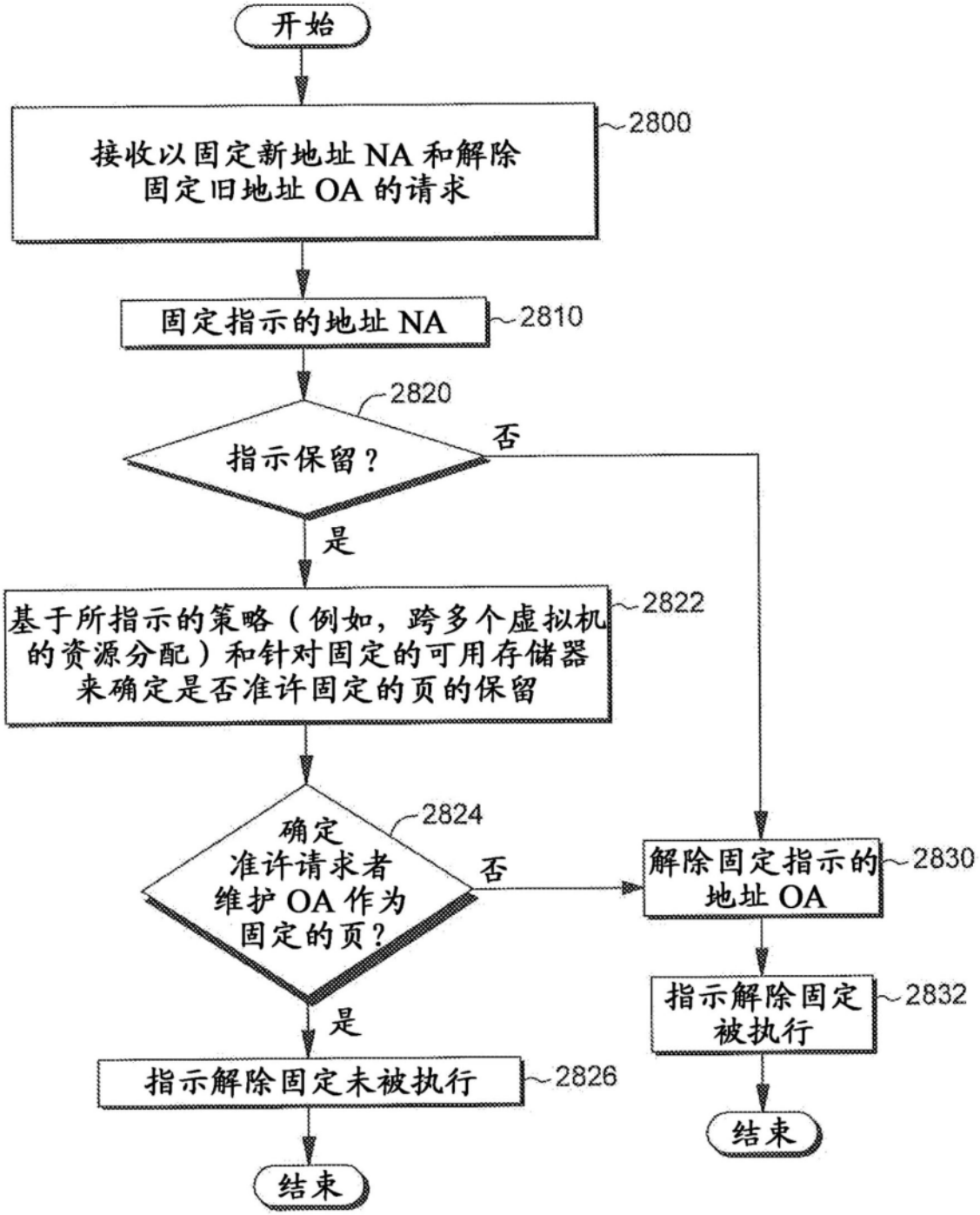

开始
接收以固定新地址 NA 和解除固定旧地址 OA 的请求
2800
固定指示的地址 NA
2810
2820
指示保留？
否
是
基于所指示的策略（例如，跨多个虚拟机的资源分配）和针对固定的可用存储器来确定是否准许固定的页的保留
2822
确定准许请求者维护 OA 作为固定的页？
2824
否
是
解除固定指示的地址 OA
2830
指示解除固定被执行
2832
指示解除固定未被执行
2826
结束
结束

图28

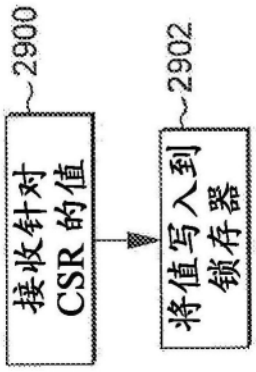


图29A

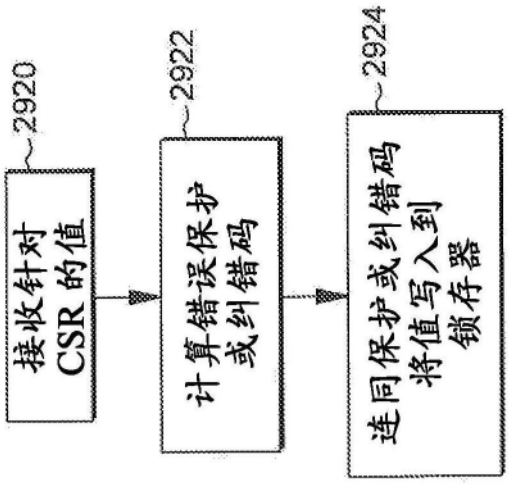


图29B

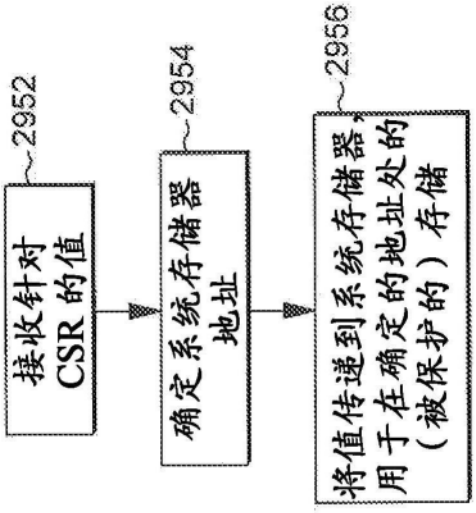


图29C

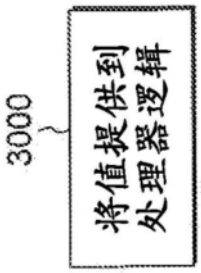

3000
将值提供到处理器逻辑

图30A

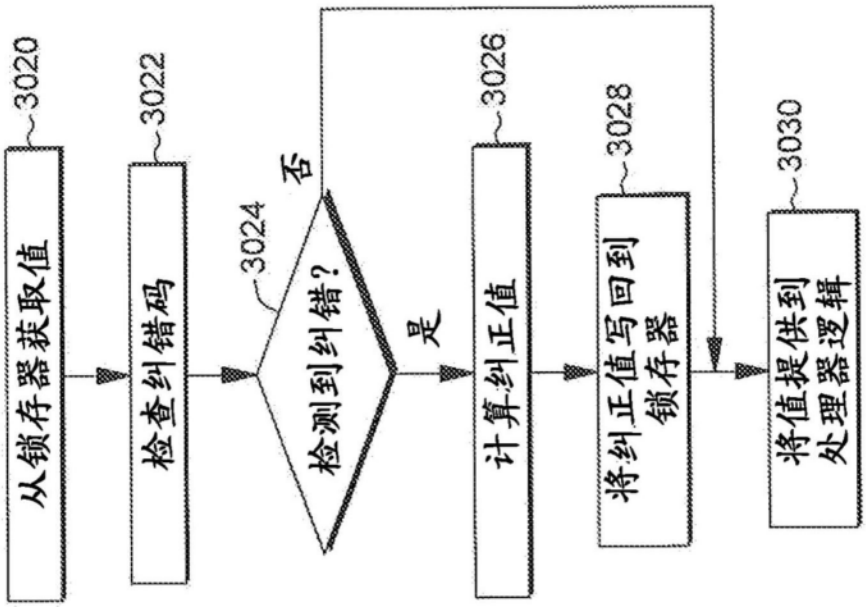

3020
从锁存器获取值
3022
检查纠错码
3024
检测到纠错?
否
是
3026
计算纠正值
3028
将纠正值写回到锁存器
3030
将值提供到处理器逻辑

图30B

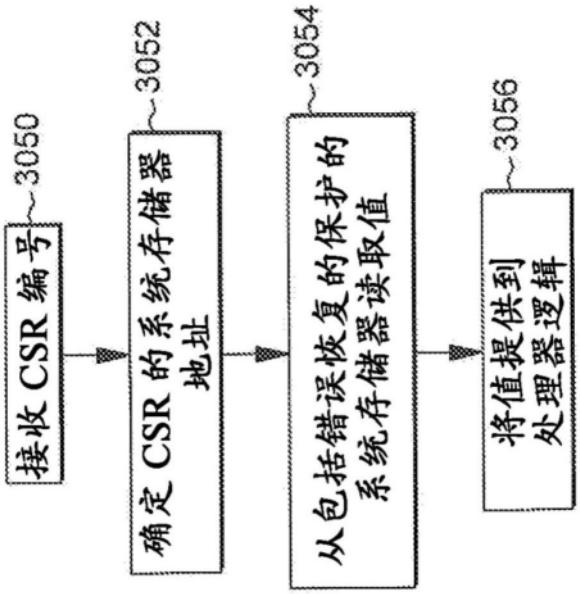

3050
接收 CSR 编号
3052
确定 CSR 的系统存储器地址
3054
从包括错误恢复的保护的系统存储器读取值
3056
将值提供到处理器逻辑

图30C

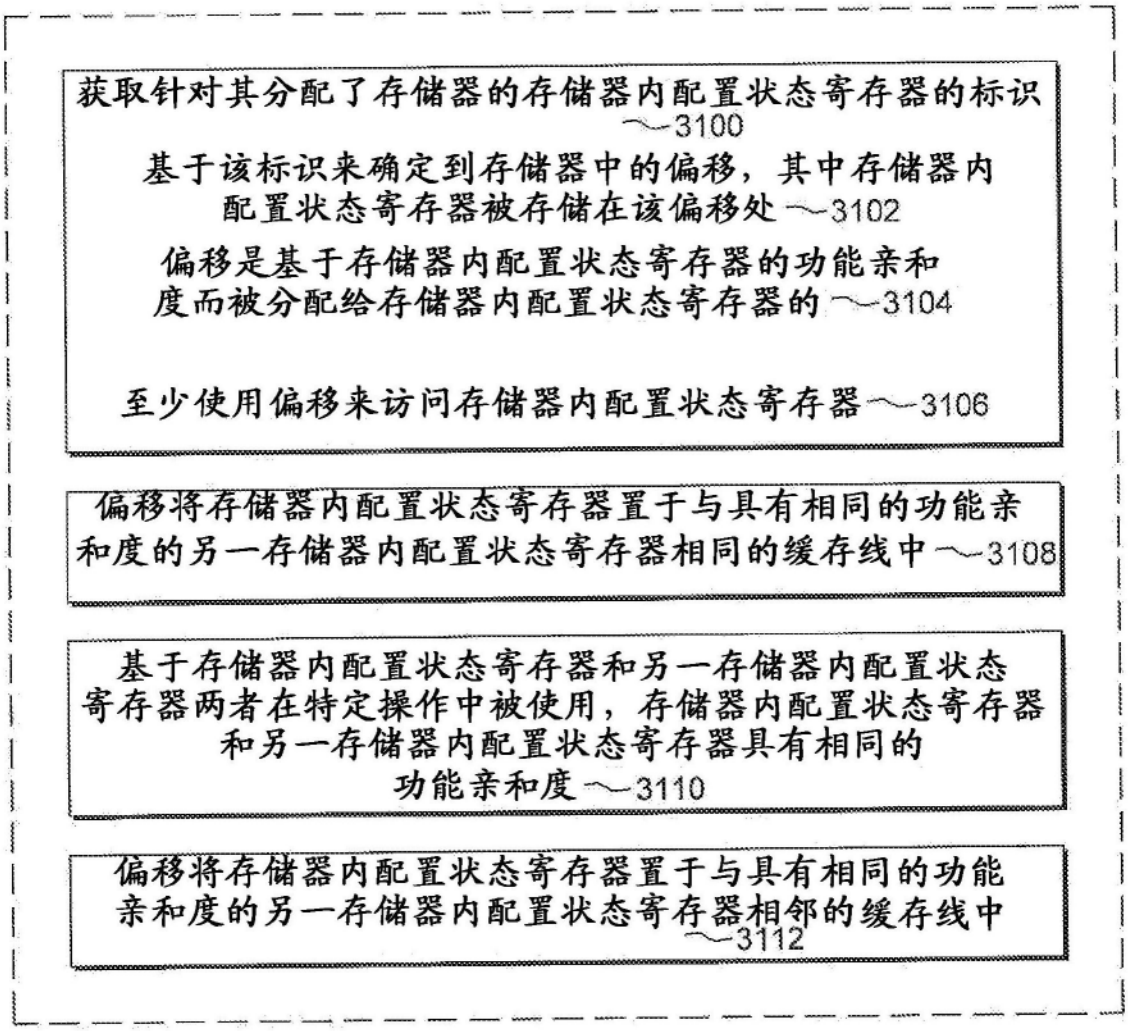

获取针对其分配了存储器的存储器内配置状态寄存器的标识 ~3100
基于该标识来确定到存储器中的偏移，其中存储器内配置状态寄存器被存储在该偏移处 ~3102
偏移是基于存储器内配置状态寄存器的功能亲和度而被分配给存储器内配置状态寄存器的 ~3104
至少使用偏移来访问存储器内配置状态寄存器 ~3106
偏移将存储器内配置状态寄存器置于与具有相同的功能亲和度的另一存储器内配置状态寄存器相同的缓存线中 ~3108
基于存储器内配置状态寄存器和另一存储器内配置状态寄存器两者在特定操作中被使用，存储器内配置状态寄存器和另一存储器内配置状态寄存器具有相同的功能亲和度 ~3110
偏移将存储器内配置状态寄存器置于与具有相同的功能亲和度的另一存储器内配置状态寄存器相邻的缓存线中 ~3112

图31A

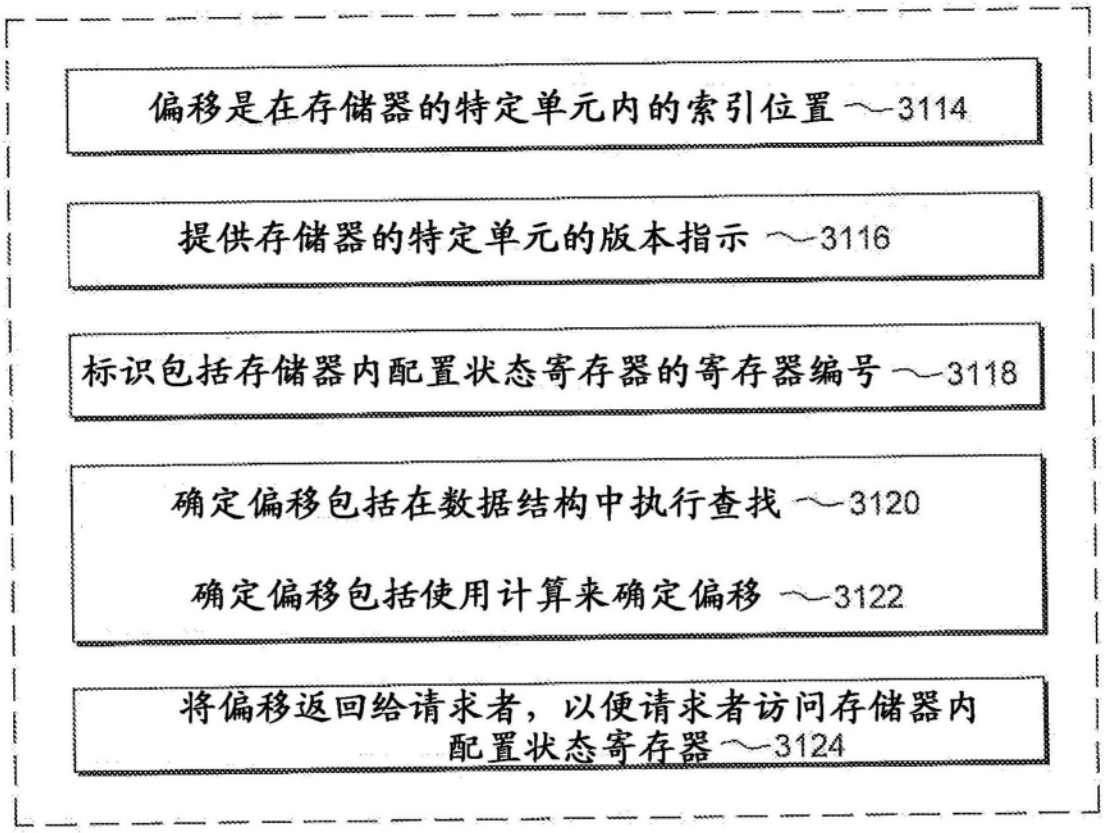

偏移是在存储器的特定单元内的索引位置 ~3114
提供存储器的特定单元的版本指示 ~3116
标识包括存储器内配置状态寄存器的寄存器编号 ~3118
确定偏移包括在数据结构中执行查找 ~3120
确定偏移包括使用计算来确定偏移 ~3122
将偏移返回给请求者，以便请求者访问存储器内配置状态寄存器 ~3124

图31B

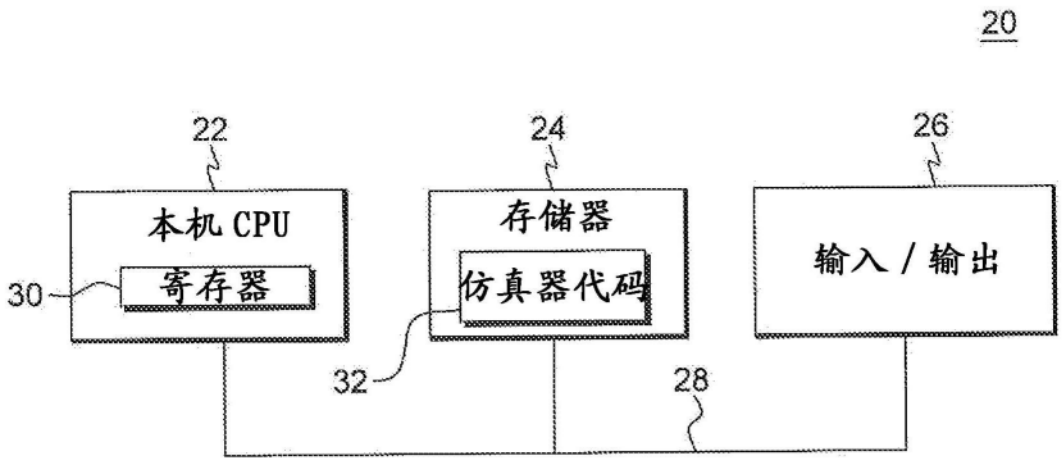

20
22
24
26
本机 CPU
寄存器
存储器
仿真器代码
输入 / 输出
30
32
28

图32A

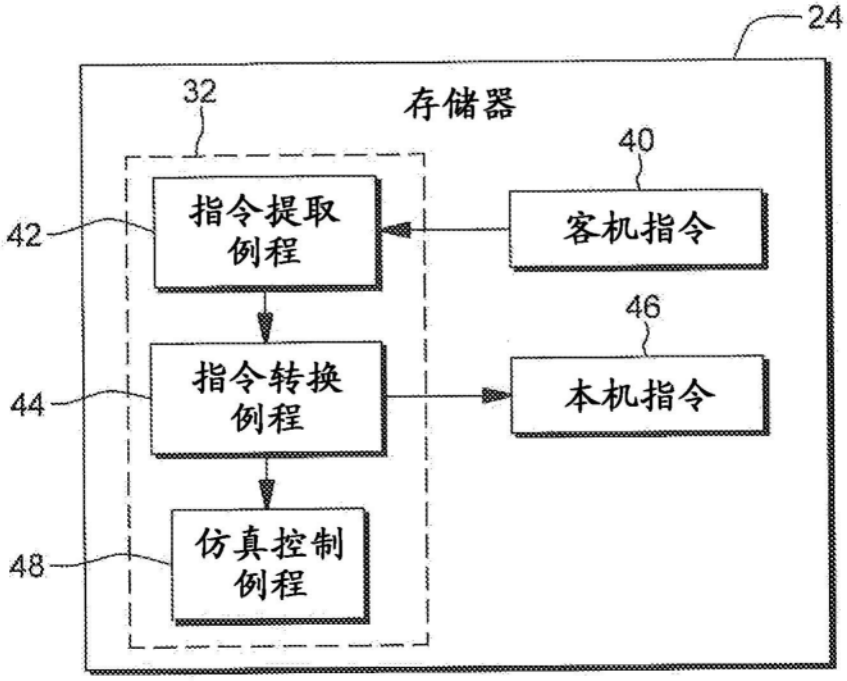


图32B

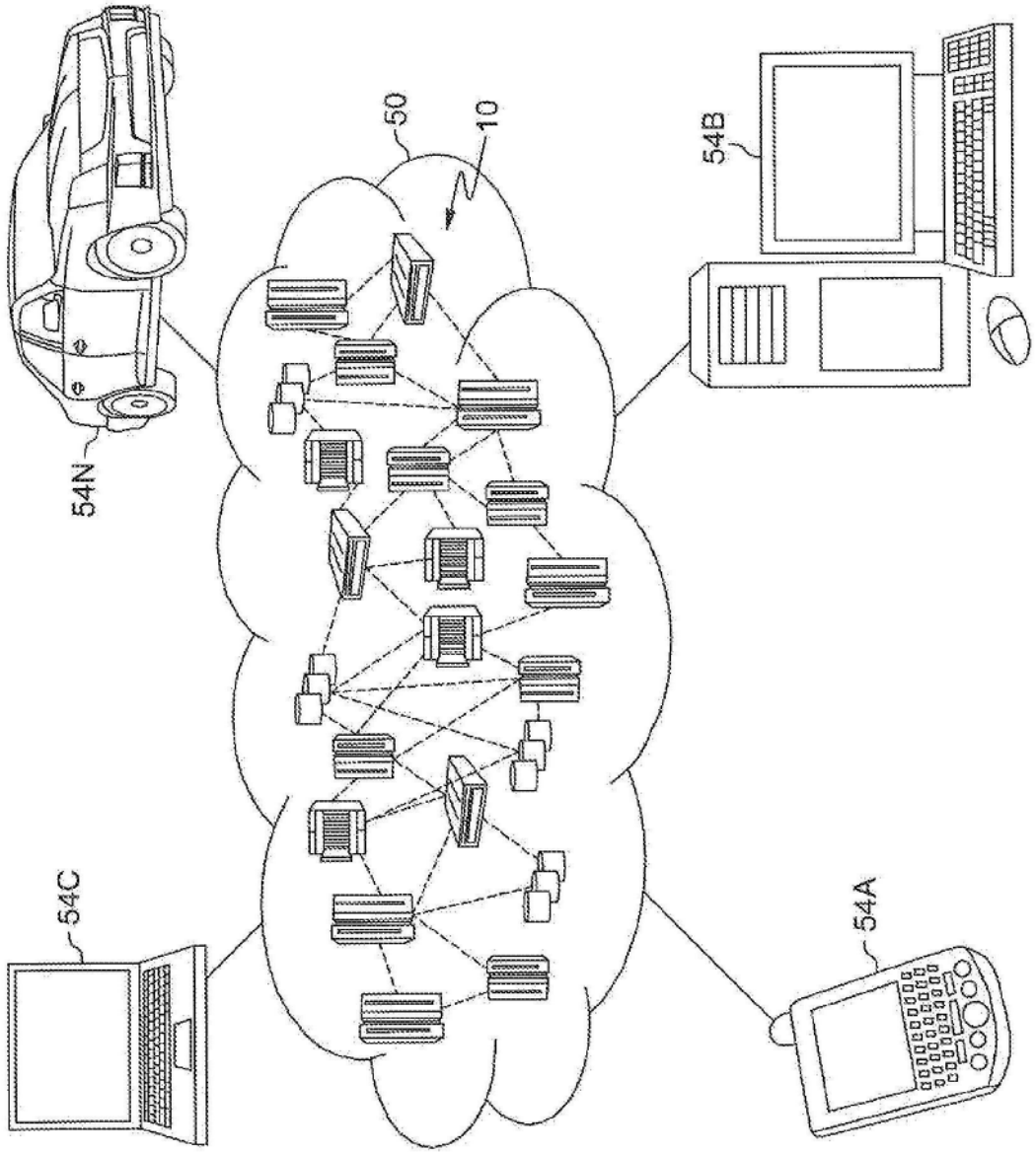


图33

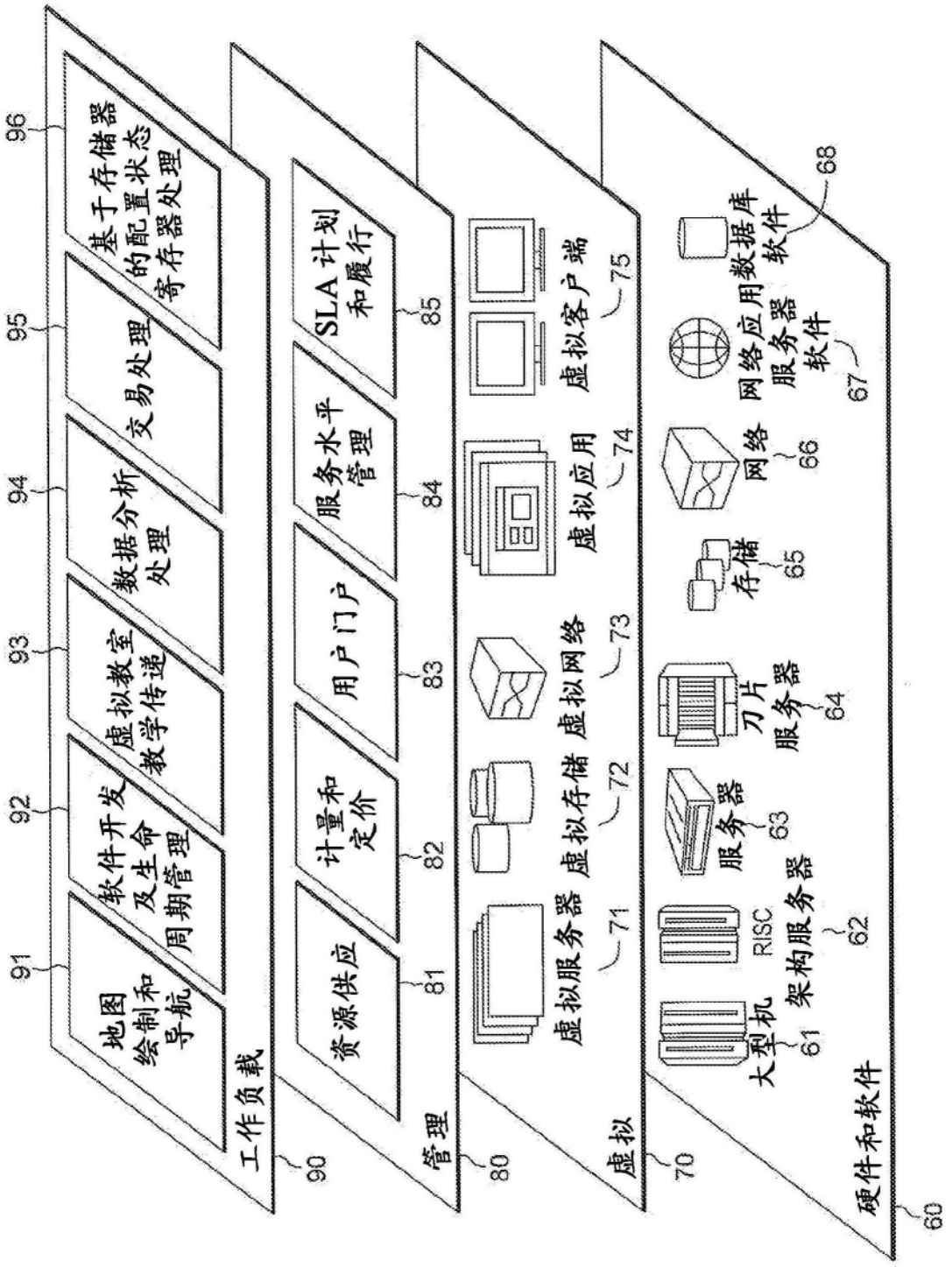

硬件和软件
60
虚拟
70
管理
80
工作负载
90
大型机
61
RISC
架构服务器
62
虚拟服务器
71
虚拟存储
72
虚拟网络
73
虚拟应用
74
虚拟客户端
75
服务器
63
刀片
服务器
64
存储
65
网络
66
网络应用
服务器
软件
67
数据库
软件
68
资源供应
81
计量和
定价
82
用户门户
83
服务水平
管理
84
SLA 计划
和履行
85
地图
绘制和
导航
91
软件开发
及生命
周期管理
92
虚拟教室
教学传递
93
数据分析
处理
94
交易处理
95
基于存储器
的配置状态
寄存器处理
96

图34