(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 1993683 B
(45)授权公告日 2010.09.01

(21)申请号 200580025726.X

(22)申请日 2005.07.14

(30)优先权数据
10/903,704 2004.07.30 US

(85)PCT申请进入国家阶段日
2007.01.29

(86)PCT申请的申请数据
PCT/US2005/024997 2005.07.14

(87)PCT申请的公布数据
WO2006/019914 EN 2006.02.23

(73)专利权人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 J·布兰特 S·K·蒙达尔
R·乌利希 G·尼格 R·乔治

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 陈斌

(51)Int.Cl.
G06F 12/10(2006.01)
G06F 9/455(2006.01)
G06F 12/14(2006.01)

(56)对比文件
US 6047354 A,2000.04.04, 说明书第22栏第25行-第23栏第25行.
CN 1506847 A,2004.06.23, 全文.
US 5123101 A,1992.06.16, 全文.
US 2003/0217250 A1,2003.11.20, 全文.
US 6625720 B1,2003.09.23, 全文.
US 2002/0082824 A1,2002.06.27, 说明书第32、42-44、52段.
EP 1296333 A2,2003.03.26, 说明书第31、32段,图2.
US 5319760 A,1994.06.07, 说明书第1栏第45行-第2栏第15行,第10栏第34行-第12栏第3行,图1-11.
CN 1436332 A,2003.08.13, 全文.

审查员 张文

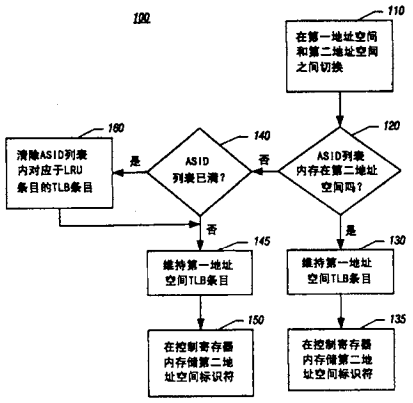
权利要求书 2 页 说明书 7 页 附图 6 页

(54)发明名称

体系结构事件期间维持处理器资源

(57)摘要

在本发明一实施例中,一种方法包括在第一地址空间和第二地址空间之间切换,确定在地址空间列表内是否存在该第二地址空间;并且在切换之后维持翻译缓冲器内第一地址空间的条目。这样,可降低与上下文切换相关联的额外开销。



CN 1993683 B

1. 一种用于在一体系结构事件期间维持一处理器资源的方法，包括：

在第一地址空间和第二地址空间之间切换；

确定在地址空间标识符列表内是否存在所述第二地址空间，所述地址空间标识符列表被存储在处理器的便笺式存储器中；并且

如果所述第二地址空间存在在所述地址空间标识符列表内，则在所述切换之后维持翻译缓冲器内所述第一地址空间的条目。

2. 如权利要求 1 所述的方法，其特征在于，还包括从所述地址空间标识符列表中获取对应于所述第二地址空间的地址空间标识符，并且将所述地址空间标识符存储在控制寄存器内。

3. 如权利要求 1 所述的方法，其特征在于，还包括如果在所述地址空间标识符列表内不存在所述第二地址空间，则清除所述翻译缓冲器内选定地址空间的条目。

4. 如权利要求 1 所述的方法，其特征在于，还包括确定对受保护位置的存储是否引起了所述切换。

5. 如权利要求 4 所述的方法，其特征在于，还包括如果所述存储引起了所述切换，则从所述翻译缓冲器清除对应于所述受保护位置的条目。

6. 如权利要求 5 所述的方法，其特征在于，清除对应于所述受保护位置的所述条目包括使用无效条目指令使所述翻译缓冲器内的线性地址无效。

7. 如权利要求 1 所述的方法，其特征在于，所述第一地址空间包括客户软件，而所述第二地址空间包括虚拟机监视器。

8. 如权利要求 7 所述的方法，其特征在于，还包括在虚拟机退出操作时维持所述条目。

9. 如权利要求 1 所述的方法，其特征在于，所述第一地址空间包括虚拟机监视器，而所述第二地址空间包括客户软件。

10. 如权利要求 9 所述的方法，其特征在于，还包括在虚拟机进入操作时维持所述条目。

11. 一种用于在一体系结构事件期间维持一处理器资源的方法，包括：

在客户软件和虚拟机监视器之间切换；

确定对受保护位置的存储是否引起了所述切换；以及

选择性地清除对应于所述受保护位置的处理器资源的至少一个条目，同时维持对应于所述客户软件的所述处理器资源的其他条目。

12. 如权利要求 11 所述的方法，其特征在于，还包括如果在地址空间标识符列表中不存在对应于所述虚拟机监视器的地址空间标识符，则清除所述处理器资源内选定地址空间的页面条目。

13. 如权利要求 11 所述的方法，其特征在于，还包括使用无效条目指令选择性地清除所述至少一个条目。

14. 如权利要求 11 所述的方法，其特征在于，所述切换是由控制寄存器移动指令引起的。

15. 如权利要求 11 所述的方法，其特征在于，选择性地清除所述至少一个条目包括清除未由当前地址空间使用的至少一个条目。

16. 如权利要求 11 所述的方法，其特征在于，还包括使用微代码启用对其他条目的维

持。

17. 一种用于在一体系结构事件期间维持一处理器资源的装置，包括：

具有第一和第二地址空间的条目的处理器资源，所述第一和第二地址空间对应于相应的第一和第二地址空间标识符；以及

耦合至所述处理器资源的执行引擎，以当所述第二地址空间存在在一地址空间标识符列表内时，在上下文切换至所述第二地址空间时维持所述第一地址空间的至少一部分条目，所述地址空间标识符列表被存储在处理器的便笺式存储器中。

18. 如权利要求 17 所述的装置，其特征在于，还包括耦合至所述处理器资源以向所述条目提供所述第一和第二地址空间标识符的地址空间缓冲器。

19. 如权利要求 17 所述的装置，其特征在于，如果对受保护位置的存储引起所述上下文切换，则所述执行引擎适于执行微代码以选择性地清除对应于受保护位置的条目。

20. 如权利要求 17 所述的装置，其特征在于，所述处理器资源包括翻译后备缓冲器。

21. 一种用于在一体系结构事件期间维持一处理器资源的系统，包括：

处理器，包括具有第一和第二地址空间的条目的处理器资源，所述第一和第二地址空间分别对应于第一和第二地址空间标识符，以及耦合至所述处理器资源的执行引擎，所述执行引擎当所述第二地址空间存在在一地址空间标识符列表内时，在上下文切换至所述第二地址空间时维持所述第一地址空间的至少一部分条目，所述地址空间标识符列表被存储在处理器的便笺式存储器中；以及

耦合于所述处理器的动态随机存取存储器。

22. 如权利要求 21 所述的系统，其特征在于，还包括耦合至所述处理器资源的地址空间缓冲器，以向所述处理器资源提供所述第一和第二地址空间标识符。

23. 如权利要求 21 所述的系统，其特征在于，还包括地址空间标识符列表，以将具有条目的地址空间标识符存储在所述处理器资源内。

24. 如权利要求 21 所述的系统，其特征在于，所述处理器资源包括翻译后备缓冲器。

体系结构事件期间维持处理器资源

[0001] 背景

[0002] 本发明一般涉及数据处理系统，尤其涉及使用处理器在不同上下文中进行处理。

[0003] 许多现有的计算机系统使用虚拟存储器系统来管理存储器，并将其分配给在系统中运行的多个进程，这就允许在该系统上运行的每个进程像它能控制由该系统提供的全部范围的地址那样来工作。操作系统 (OS) 将用于每个进程的虚拟地址空间映射至用于该系统的实际物理地址空间。通常使用页面表来维持从物理地址到虚拟地址的映射。

[0004] 处理器性能可经由多级流水线体系结构改进，在该多级流水线体系结构中诸如高速缓存、缓冲器、阵列等的各种流水线资源可更为有效地用于执行指令。改进虚拟存储器系统的使用的一种流水线资源是翻译后备缓冲器 (TLB)。TLB 是处理器流水线内存储器的相对较小部分，用于高速缓存部分系统虚拟地址至物理地址的翻译。具体地，翻译集的少量元素被存储在处理器能极快访问的 TLB 中。系统中可存在各种 TLB。例如，存在用于指令和数据的单独 TLB（分别为指令 TLB(iTLB) 和数据 TLB(dTLB)。此外，在某些系统中可存在二级 dTLB(STLB)。

[0005] 如果在 TLB 中不存在对特定虚拟地址的翻译，则会出现“翻译遗漏”并且使用更一般机制来解决地址翻译。处理以此方式继续直到出现上下文切换。上下文切换会在多任务 OS 停止时运行一个进程（例如，应用软件）并开始运行另一个进程时发生。当发生上下文切换时，包括页面目录的页面表和用于新进程的页面表被载入，并且该 TLB 和其他流水线资源必须被清除。清除意味着资源内容被清空。

[0006] 某些系统可实现其中虚拟机监视器 (VMM) 可向其他软件呈现一个或多个虚拟机的抽象的虚拟机 (VM) 环境。每个 VM 都可用作自主式平台，从而运行它自己的“客户操作系统”（即，由 VMM 提供的 OS) 以及被统称为客户软件（或简称为“客户”）的其他软件。客户软件期望像在专用计算机而非虚拟机上运行那样来工作。因此，客户软件期望控制各种事件并能够访问诸如驻留处理器资源、驻留于存储器内的资源和驻留于基础硬件平台的资源的硬件资源。

[0007] 在虚拟机环境中，VMM 通常对这些事件和硬件资源有着最终控制权，以提供客户软件的合适操作，并且为在不同虚拟机上运行的客户软件或在这些软件之间提供保护。为了实现这一目标，VMM 通常在客户软件访问受保护资源时或在出现其他事件（诸如中断或异常）时接收控制（即，影响环境切换）。

[0008] 在 VM 或其他环境中的上下文切换会引起现代微处理器中相当大的额外开销。该额外开销会因为在每次上下文切换时必须清除并重新加载的巨大二级 TLB 而加剧。这样该额外开销就会对性能，特别是带有多个活动上下文的系统中的性能产生不利影响。于是就需要在上下文切换时能更有效地维持流水线资源。

[0009] 附图简述

[0010] 图 1 是根据本发明一实施例的线性地址翻译。

[0011] 图 2 是根据本发明一实施例的系统的一部分的框图。

[0012] 图 3 是根据本发明一实施例的方法的流程图。

[0013] 图 4 是根据本发明另一实施例的方法的流程图。

[0014] 图 5 是根据本发明又一实施例的方法的流程图。

[0015] 图 6 是本发明一实施例的代表性计算机系统的框图。

[0016] 详细描述

[0017] 在本发明的各个实施例中，诸如 TLB、跟踪缓存 (TC)、分支预测单元 (BPU) 阵列、小型标志等的流水线资源或结构可支持多地址空间。在此使用的术语“地址空间”指的是存储器内对应于给定应用程序 (例如上下文) 的一组地址。

[0018] 在各个实施例中，地址空间可受到确定或影响物理翻译线性的体系结构控制寄存器的组合 (例如，包括页面目录基址寄存器 (CR3.PDBR)、页面级高速缓存禁用位 (PCD)、页面级直写位 (PWT) 的控制寄存器 3 ；包括分页位 (CR0.PG)、保护启用位 (CR0.PE) 的控制寄存器 0 ；包括页面大小扩展位 (CR4.PSE) 以及页面全局启用位和物理地址扩展位 (CR4.PGE 和 PAE)、经扩展的功能启用长寄存器模式寻址 (EFER.LMA) 和非执行位 (EFER.NXE) 的控制寄存器 4) 的控制。

[0019] 在各个实施例中，地址空间标识符 (ASID) 可用于以指向与各个流水线资源中的线性地址相关联的上下文的指针来增加这些地址。在这里使用的“地址空间标识符”可以是标识与其相关联的一个或多个地址空间的任何数字、代码或其他符号。在一个实施例中，ASID 可使用每个线程两位 ASID(即，四个地址空间上下文) 来实现，尽管本发明的范围并非如此有限。这就允许多个应用程序上下文共享流水线结构，从而降低了上下文切换的额外开销。例如，当上下文切换出现时，可改变当前的 ASID 值，而不是清除流水线结构。类似地，在某些实施例中，可提供线程标识符 (线程 ID) 以标识用于对应地址空间的给定处理器线程。

[0020] 根据本发明的各个实施例，各种体系结构事件都可能会引起 TLB 的选择性清除。这些事件例如可包括 ：移至 CR3(MOV to CR3) 指令 ；改变 CR0 页面模式位 ：CR0.PE 和 CR0.PG 以及 EFER.LMA ；改变 CR4 页面模式位 ；VM 进入 (Entry)/ 退出 (Exit) ；以及 32 位上下文切换 (如果改变了 CR0 或 CR4 中的分页模式位)。在某些实施例中，这些事件中仅有 MOV to CR3 操作或 VMEntry/VMExit 事件可改变 / 递增 ASID。

[0021] 许多流水线结构被线性标志或索引。在某些实施例中，ASID 可用指向对应地址空间的指针来增加这些流水线资源内的线性地址。在这些实施例中，微处理器可以维持在创建新的地址空间时或改变至不同的先前已见的地址空间时更新的全局当前 ASID 寄存器。可用当前 ASID 值扩展 TLB 插入，并且只有当 ASID 标志与当前 ASID 值匹配时 TLB 查找才匹配。当上下文切换 (例如，MOV to CR3 或者 VM entry/exit) 触发了地址空间改变时，微处理器可切换至表示此新地址空间的不同 ASID 值，而不清除 TLB 或其他流水线结构。在某些实施例中，不清除任何条目或者选择性地清除对应于特定地址空间的部分或所有条目可为带有多个上下文的环境提供实质上的性能增益。

[0022] 现参看图 1，示出了根据本发明一实施例的适于 64 位地址翻译的线性地址翻译。如图 1 所示，可使用线性地址 300 来寻址存储器 240 内的页面。如图 1 所示，线性地址 300 包括索引四级分页结构以访问存储器 240 内物理地址的多个字段。偏移字段 (位 0 至 11) 用于寻址存储器 240 的一个页面内的物理地址。页面表条目 (位 12 至 20) 寻址页面表 (PT)310 内的页面表条目。页面目录条目 (位 21 至 29) 寻址页面目录 (PD)320 内的页面目

录条目。页面目录指针（位 30 至 38）寻址页面目录指针表 (PDPT)330 内的页面目录指针条目。最后，页面映射级四 (PML4) 指针（位 39 至 47）寻址 PML4 340 内的 PML4 条目。使用 CR3 内的指针就可访问 PML4 340 的基址。以此方式，可使用 64 位线性地址实现四级分页结构以访问物理地址。

[0023] 在各个实施例中，根据一实施例的 TLB 可包括物理地址（标志地址）和相关联的有效线程 ID 以及 ASID 位，连同相对应的线性和物理地址。TLB 条目包括在其标志内指示它们所对应地址空间的 ASID 字段。该 ASID 字段包括在载入 TLB 条目时的当前 ASID 值。因为 ASID 字段在 TLB 条目的标志内，所以 TLB 条目只有在当前 ASID 值与该 TLB 条目内的 ASID 字段相匹配时才能被命中。

[0024] 当一地址空间改变（例如，上下文切换）时，微代码会将关于当前地址空间 / ASID(CR3、CR0 和 CR4 分页位、LMA) 的信息保存到便笺式存储器，并且可将新地址空间与每个 ASID 的已存储地址空间信息相比较。如果没有已存储地址空间 /ASID 与新地址空间相匹配，则在之前未看到过该地址空间。因而微代码可为该地址空间分配新 ASID，或选择一现有 ASID 来代替例如最近最少使用 (LRU) 策略。在后一种情况下，微代码可清除用于该 LRU ASID 的 TLB 条目。

[0025] 现参看图 2，示出了根据本发明一实施例的系统的一部分的框图。如图 2 所示，系统 10 包括系统存储器 20 和多个软件实体，包括第一客户软件（即，第一客户）30、第二客户软件（即，第二客户）40 和虚拟机扩展 (VMX) 监视器 50。在一个实施例中，系统存储器 20 可包括代码和数据两者，并且可由例如动态随机存取存储器 (DRAM) 形成。

[0026] 在图 2 的实施例中，VMX 监视器 50 可向其他软件（例如，客户软件）呈现一个或多个虚拟机的抽象。VMX 监视器 50 可向各种客户机提供相同或不同的抽象。在各 VM 上运行的客户软件可包括客户 OS 和各种客户软件应用程序。这些客户 OS 和软件应用程序在此被统称为客户软件。客户软件期望访问该客户软件在其上运行的 VM 内的物理资源（例如，处理器寄存器、存储器和输入 / 输出 (I/O) 设备）。VMX 监视器 50 可有助于访问客户软件所期望的资源，同时还保持对平台硬件内资源的最终控制。

[0027] 在一个实施例中，可使用存储在虚拟机控制结构 (VMCS)（未在图 2 中具体示出）内一个或多个指定字段中的数据来定义标准（控制传送标准和 / 或故障标准）。例如，VMCS 可存储客户软件的状态和信息以控制其操作。VMCS 可驻留于存储器 20 中并且可由处理器维持。应该理解，可使用任何其他的数据结构（例如，芯片内高速缓存、文件、查找表等）存储已存储在 VMCS 内的信息。

[0028] 在一个实施例中，如果传送标准确定当前事件必须由 VMX 监视器 50 处理，则控制就被传送至 VMX 监视器 50。VMX 监视器 50 然后可处理该事件并将控制传送回客户软件。从 VMM 或 VMX 监视器到客户软件的控制传送在这里可被称为 VM 进入，而从客户软件到 VMM 或 VMX 监视器的控制传送在此可被称为 VM 退出。

[0029] 如图 2 所示，存储器 20 可包括可使用关于第一客户软件 30 的页面表条目 (PTE) 访问的第一页面 35，以及可使用关于第二客户软件 40 的 PTE 访问的第二页面 45。也就是说，第一页面 35 和第二页面 45 可包括分别对应于第一客户 30 和第二客户 40 的代码和 / 或数据。在图 2 的实施例中，存储器 20 可以对应于包括代码和数据两者的物理地址，并且对应于客户软件 30 和客户软件 40 的 ASID 的地址可对应于用来指向物理地址空间的线性地址

（或其一部分）。

[0030] 仍参看图 2，第一客户 30 具有第一 ASID 值（即，ASID ＝ 1），并且还包括对应于第一地址空间的 CR3 值。同样地，第二客户 40 具有第二 ASID 值（即，ASID ＝ 2）并且还具有与其相关联的 CR3 值。

[0031] 控制第一客户软件 30 和第二客户软件 40 的执行的 VMX 监视器还具有与其相关联的 ASID 值。例如，如图 2 的实施例中所示，VMX 监视器 50 可具有三个与其相关联的 ASID 值。

[0032] 如图 2 所示，可执行 MOV to CR3 指令以产生如箭头 55 所示的上下文切换。此外，VMX 监视器 50 通过产生由标号 60 表示的 VM 退出来控制客户软件的操作，该 VM 退出引起从当前执行客户中退出以强制控制返回 VMX 监视器 50。类似地，VMX 监视器 50 可通过执行 VM 进入（由图 2 中的标号 70 表示）就可引起客户的起动或恢复。

[0033] 在各个实施例中，VMX 体系结构增强能允许 VMX 监视器在上下文切换和使用 ASID 的 VMX 转换时避免清除 TLB 或其他类似的处理器资源。在各个实施例中，既不需要软件也不需要硬件中的显式 ASID 管理来检测交叉地址空间污染。

[0034] 在各个实施例中，基于诸如图 2 所示的事件，包括使用 MOV to CR3 指令执行的上下文切换、VM 退出或 VM 进入，可避免对 TLB 或其他处理器资源的清除，从而在该上下文切换以及其他体系结构事件时改进性能并降低额外开销。

[0035] 在某些实施例中，系统存储器 20 内的页面以及其他被高速缓存的页面于是就可由跨所有地址空间的 VMX 监视器 50 来保护，从而对这些页面的任何尝试存储都可引起从起动这一存储的客户软件的 VMX 退出。对于这一事件，VMX 监视器 50 可使得对应于这些存储的地址在所有 TLB 和其他处理器资源内无效。

[0036] 在各个实施例中，处理器可不同地处理某些事件、指令和 VMX 转换。不同指令和事件的这些不同实现可在处理器微代码中实现。例如，对于客户软件内的 MOVto CR3 指令、VM 退出或 VM 进入，处理器可寻找与新地址空间匹配的现有 ASID。这一确定可通过比较新地址的 ASID 与处理器维持的 ASID 列表来作出。如果匹配存在，则处理器使该 ASID 成为当前 ASID 而无需清除任何 TLB 条目。否则，在一个实施例中对应于 LRU ASID 的所有 TLB 条目就被清除，并且最近最少使用 ASID 的 ASID 值就作为当前 ASID。如果该 ASID 未被清除，则可维持来自在前地址空间的 TLB 条目。

[0037] 与之相反，可不同地处理使用于特定页面的 TLB 条目无效的指令（例如，INVLPG 指令）。例如，在一个实施例中，可基于这一指令使所有 TLB 内的线性地址空间无效。不同的操作仍可对其他指令出现。例如，在一个实施例中，检测到 CR4 值内无改变的 MOV to CR4 指令可使每一 ASID 内的所有 TLB 条目无效。

[0038] 现在参看图 3，示出了根据本发明一实施例的方法的流程图。方法 100 可用于在处理器内的不同上下文之间切换。如图 3 所示，可进行第一和第二地址空间之间的切换（框 110）。虽然未在图 3 中示出，但是在这一切换之前，第一地址空间可被保存到对应于当前地址空间标识符的控制寄存器（或其他存储器）中。在其它实施例中，这一保存可在其他时刻进行。

[0039] 地址空间切换可对应于 VM 退出或 VM 进入，或者可由 MOV to CR3 指令或其他类似事件所引起。接着，可确定 ASID 列表中是否存在第二地址空间（菱形框 120）。这一列表可

由处理器维持以标识不同的活动地址空间、与其相对应的线性地址空间内的位置、以及其他任选信息。例如，ASID 列表可以是处理器中的便笺式存储器。

[0040] 如果 ASID 列表内存在第二地址空间，则控制就可传至框 130，其中可维持包括了对应于第一地址空间的 TLB 内任何条目的第一地址空间。此外，第二 ASID 可被存储在处理器的控制寄存器（例如，当前 ASID 寄存器）（或其他存储器位置）内以指示该第二地址空间（即，在菱形框 120 中找到的地址空间）是处理器内当前执行的地址空间（框 135）。

[0041] 如果相反在菱形框 120 处确定 ASID 列表内不存在第二地址空间，则接着就确定 ASID 列表是否已满（菱形框 140）。如果该列表未满，则控制可传至框 145，其中可维持包括了对应于第一地址空间的 TLB 内任何条目的第一地址空间。此外，第二 ASID 可被存储在处理器的控制寄存器（例如，当前 ASID 寄存器或其他存储器位置）内以指示该第二地址空间（即，来自以下讨论的框 160 的空地址空间或自由地址空间）是处理器内的当前执行地址空间（框 150）。

[0042] 如果相反在菱形框 140 处确定 ASID 列表已满，则在 ASID 列表内对应于 LRU 条目的 TLB 内的条目就被清除（菱形框 160）。随后控制可传至方框 145 以便于进一步的处理，其中如上所述在该框 145 处第二 ASID 可被存储在 LRU 条目内。因此，就能启用 TLB 条目以及与第二 ASID 相关联的其他资源。

[0043] 虽然在图 3 实施例中示为使用 LRU 策略来清除 TLB 条目，但是应该理解，在其它实施例中也可使用其他策略或机制来确定在 ASID 列表全满时清除的合适地址空间。

[0044] 在某些实施例中，取决于地址空间切换的原因，可使用软件清除 TLB 内的特定条目或其他处理器资源。例如，如上所述，如果对受保护页面的存储尝试引起 VM 退出，则清除对应于该受保护空间的 TLB 内的地址。也就是说，在 VMX 体系结构的各种实现中，VMX 监视器保护每个分页的分层结构。在虚拟化的环境中，软件可用于清除 TLB 或由客户支配的其他资源。随后，因为虚拟化软件处理本文中所述的清除活动，所以由客户承担的某些事件（例如，MOVto CR3 指令）将不会清除资源。于是在客户中，如果任何存储指令修改了由 CR3 在 CR3 目标值中使用的 TLB 条目，则会产生 VM 退出。

[0045] 使用根据本发明一实施例的半透明 ASID，软件在这一 VM 退出之后可清除对应于引起该 VM 退出的存储指令的所有 TLB 条目，即使客户在 VM 退出时刻并没有使用该 TLB 条目。在这一实施例中，如果软件期望清除单个条目，则软件可用合适地址空间内对应于 TLB 条目的线性地址来执行 INVLPG 指令。对于这一软件调用 INVLPG，处理器可清除对应于该线性地址的任何 TLB 条目，而无需考虑它们的 ASID。虽然描述了使用 INVLPG 指令来实现，但是在其它实施例中其他指令或操作也可清除这些条目。

[0046] 现在参看图 4，示出了根据本发明一实施例的方法的流程图。更具体地，图 4 示出了用于清除 TLB 内某些条目的方法 200。这一方法可在 VM 环境内执行，并且可基于 VM 退出起动。如图 4 所示，方法 200 可通过确定 VM 退出的原因（框 210）开始。例如，在一个实施例中，可分析在 VM 退出之前的最后一个操作以确定是否对受保护存储器空间执行存储或其他存储器操作引起了该 VM 退出。当然也存在使 VM 退出发生的许多其他原因。

[0047] 然后，在菱形框 220 处可确定存储操作是否尝试修改受保护的条目，诸如受保护的 TLB 条目（例如，对应于不同地址空间）。如果不是，在可继续 VMX 监视器的正常执行（框 230）。这一执行可取决于程序指令采取许多形式。在这一执行之后，控制可在需要时返回

给客户。

[0048] 如果相反确定存储尝试修改受保护的条目，则因为该 VMX 监视器能访问该客户不能访问的受保护位置，所以该 VMX 监视器可执行该存储操作（框 235）。然后，对应于该存储地址的所有 TLB 条目可从所有 TLB 和其他存储器资源中清除（框 240）。VM 进入然后将控制返回给客户以便继续执行（框 245）。这一执行例如可在下一个客户指令处继续。

[0049] 在其它实施例中，可期望清除所有 TLB 内的所有条目。例如，如果已经修改了更高级的分页机制，则可期望该清除能清空 TLB 或在 VMX 监视器不再保护地址空间时从 TLB 中移除该地址空间。在一实施例中，如果软件期望清除用于一地址空间的整个 TLB，则软件可用已经存储在那里的完全相同值执行 MOVto CR3 指令。因为该软件不指定哪个 ASID 需要被清除，所以当处理器看见这一 CR4 写入时，该处理器就清除用于所有 ASID 的所有 TLB。在各个实施例中，如果 PDE（或更高）条目被改变或者如果 CR3 从 CR3 目标值中移除并且监视器不再保护它，则软件就可清除用于一地址空间的整个 TLB。

[0050] 现参看图 5，示出了根据本发明一实施例的另一种方法的流程图。具体地，方法 250 用于清除所有 ASID 的所有 TLB。如图 5 所示，这一方法通过接收 MOV to CR3 指令（椭圆框 255）开始。处理器基于该指令的接收确定新 CR4 值是否与其前一值相同（菱形框 260）。如果该新地址值不同，则执行正常的 CR4 操作（框 270）。这些操作可基于 CR4 内不同位的状态而变化。例如，基于位状态可清除诸如 TLB 的处理器资源，并且可进行其他处理。例如，改变数据的 CR4 转换可引起对清除至少当前 TLB 内条目的需要。此外，这一 CR4 转换可要求地址空间的改变。可用与参看图 3 所述的相类似的方式来执行这一地址空间改变。

[0051] 如果相反确定新值与前一值相同，则这可以是指示 VMX 监视器或其他实体期望清除所有 TLB 的标志。如上所述，存在清除所有 TLB 的各种原因。因此，在框 280 处，可对所有 ASID 清除所有 TLB。最后，执行可在相同（即，初始）地址空间内继续（框 290）（例如，在下一个指令处）。

[0052] 虽然在图 5 的实施例中描述为基于 MOV to CR3 指令实现，但是应该理解，在其它实施例中也可使用其他指令或操作来作为指示 VMX 监视器或其他条目期望清除所有 TLB 的标志。此外，虽然参看图 5 的具体实现做出了描述，但是在其它实施例中也可分析因预定指令产生的任何选定值以确定是否有值改变。如果没有，就可对一个或多个处理器资源执行期望动作。

[0053] 现在参看图 6，示出了根据本发明一实施例的代表性计算机系统 400 的框图。如图 6 所示，计算机系统 400 包括处理器 401。在一个实施例中，处理器 401 可经由前侧总线 420 耦合至存储器集线器 430，而该存储器集线器 430 可经由存储器总线耦合至共享主存储器 440。如图 6 所示，处理器 401 根据本发明一实施例可包括 TLB 403 和 ASID 列表 405。此外，2 级 (L2) 高速缓存 407 可耦合至处理器 401。

[0054] 存储器集线器 430 还可（经由集线器链路）耦合至 I/O 集线器 435，而该 I/O 集线器 435 则耦合至 I/O 扩展总线 455 和外围总线 450。在各实施例中，I/O 扩展总线 455 可耦合至诸如键盘和鼠标的各种 I/O 设备。外围总线 450 可耦合至诸如可以是诸如闪存、插入卡之类存储器设备的外围设备 470 的各种组件。虽然本说明书提到了系统 400 的具体部件，但是对所示实施例的各种修改也是可能的。

[0055] 各实施例可用存储在存储介质上的计算机程序实现，其中这些程序具有编程计算

机系统来执行这些实施例的指令。存储介质可包括但不限于：包括软盘、光盘、光盘只读存储器（CD-ROM）、可重写光盘（CD-RW）和磁光盘的任何类型的盘；诸如只读存储器（ROM）、例如动态或静态随机存取存储器（RAM）的RAM、可擦除可编程只读存储器（EPROM）、电可擦除可编程只读存储器（EEPROM）、闪存、磁性卡或光学卡的半导体器件；或者用于存储电子指令的任何类型的介质。其他实施例可被实现为由可编程控制设备执行的软件模块。

[0056] 因为软件有助于处理器保护和无效TLB条目，所以在各实施例中可以在上下文切换（例如，MOV to CR3、VM退出和VM进入）时维持TLB条目。于是在各实施例中无需窥探TLB硬件，从而避免了窥探TLB容量或其他限制。

[0057] 在其它实施例中，半透明ASID可用于含有不将ID显式地分配给不同的地址空间、但要求处理器即使在TLB条目未被当前地址空间使用的情况下也在修改页面表之后采取具体动作清除TLB条目的页面表的任何体系结构中。

[0058] 因此，在各实施例中，处理器可支持ASID并避免TLB清除（例如，在上下文切换时），而无需专门的硬件检测交叉地址空间污染、也无需显式的软件控制和ASID分配。

[0059] 虽然已对有限数目的实施例描述了本发明，但是本领域技术人员应该认识到可从中做出许多修改和变化。所附权利要求旨在覆盖落入本发明真实精神和范围内的所有这些修改和变化。

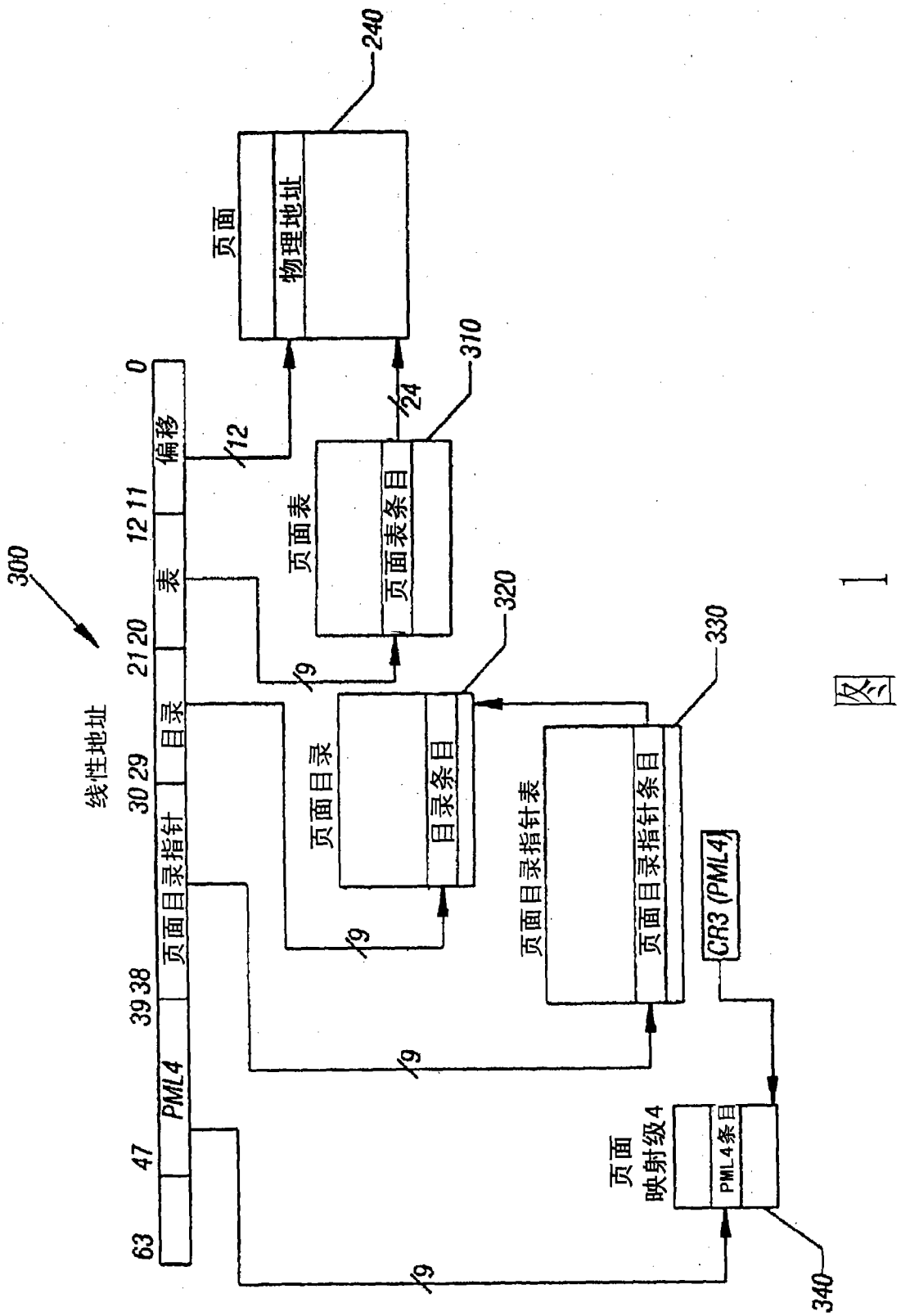

300
线性地址
63
47
39 38
30 29
21 20
12 11
0
PML4
页面目录指针
目录
表
偏移
9
12
9
9
9
页面
物理地址
240
页面表
页面表条目
310
24
页面目录
目录条目
320
页面目录指针表
页面目录指针条目
330
CR3 (PML4)
页面
映射级4
PML4条目
340

图 1

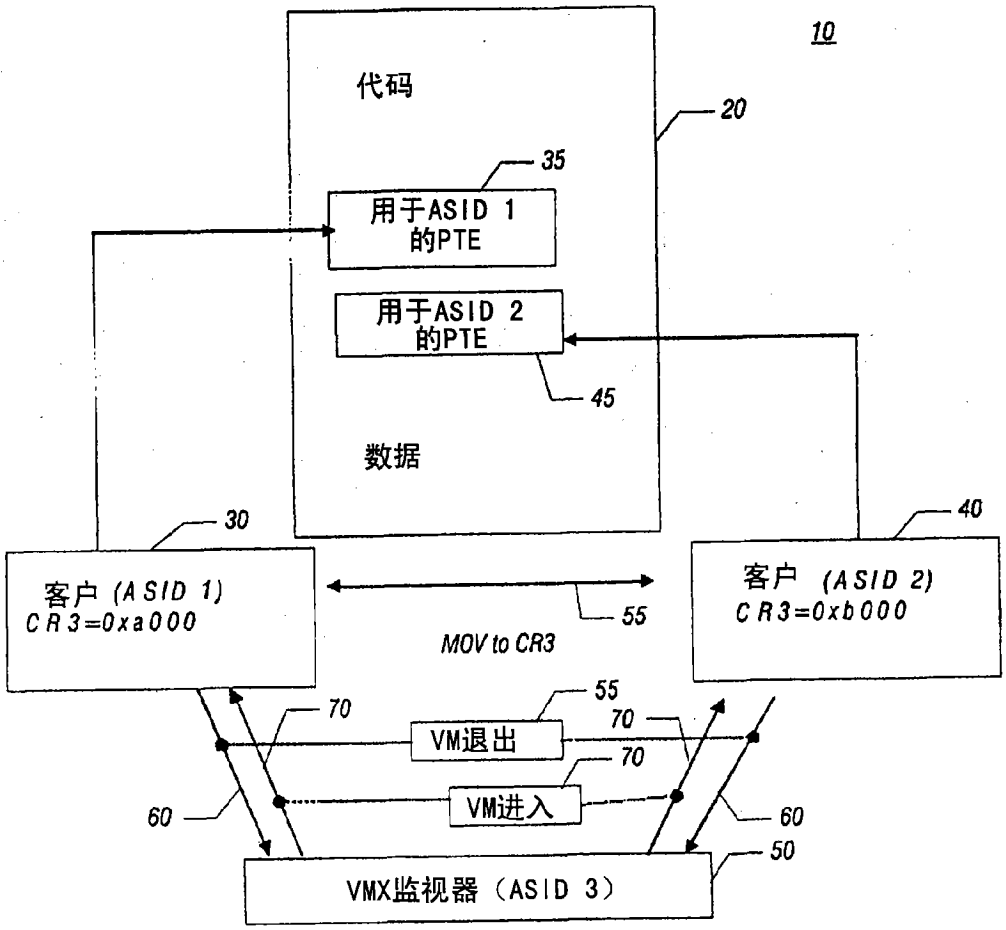

10
代码
20
35
用于ASID 1
的PTE
用于ASID 2
的PTE
45
数据
30
40
客户 (ASID 1)
CR3=0xa000
客户 (ASID 2)
CR3=0xb000
55
MOV to CR3
55
70
VM退出
70
70
VM进入
60
60
50
VMX监视器（ASID 3）

图 2

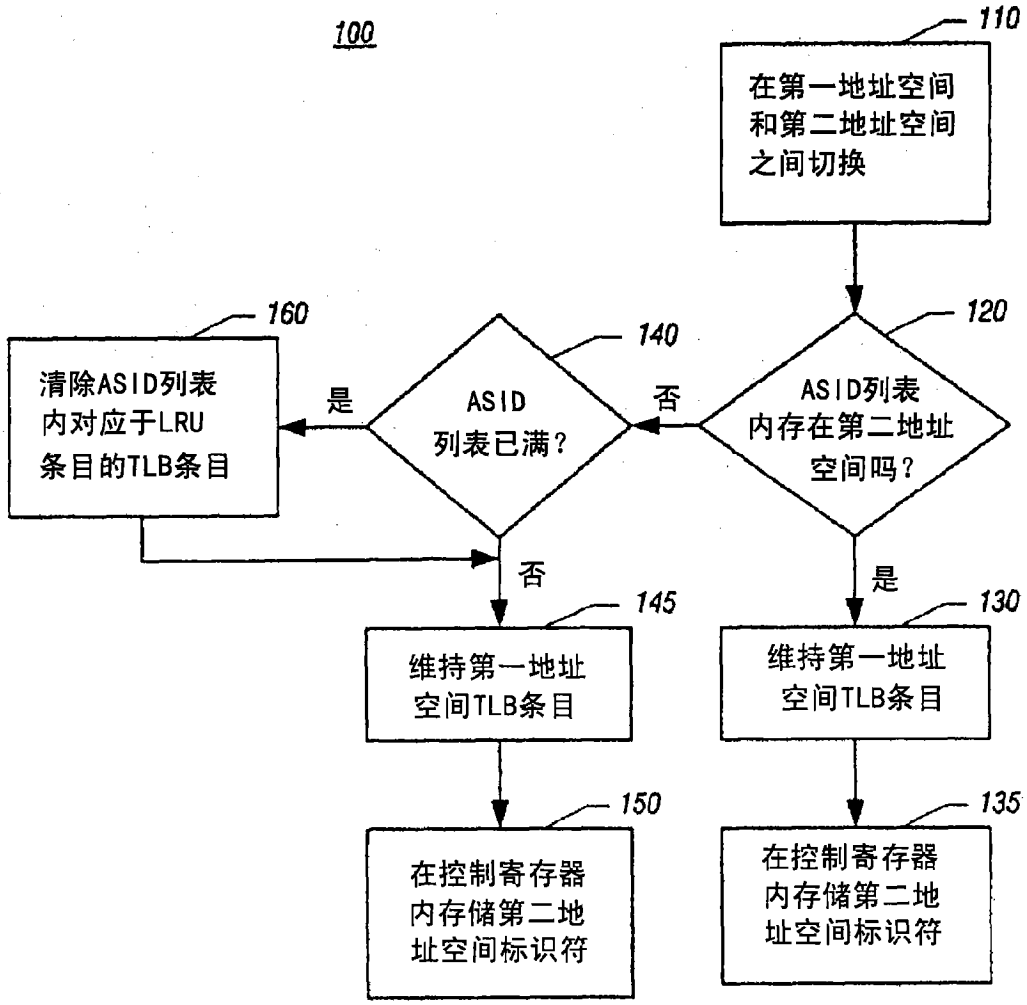

100
110
在第一地址空间和第二地址空间之间切换
120
ASID列表内存在第二地址空间吗？
否
是
140
ASID列表已满？
是
否
160
清除ASID列表内对应于LRU条目的TLB条目
130
维持第一地址空间TLB条目
135
在控制寄存器内存储第二地址空间标识符
145
维持第一地址空间TLB条目
150
在控制寄存器内存储第二地址空间标识符

图 3

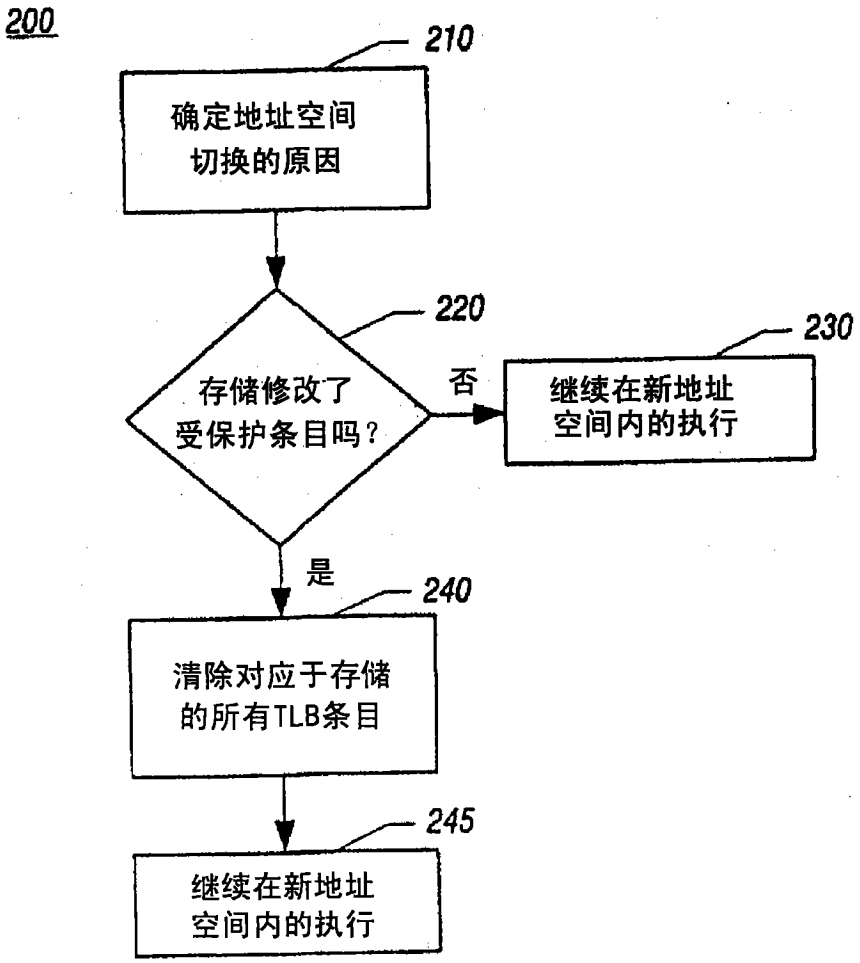

200
210
确定地址空间
切换的原因
220
存储修改了
受保护条目吗？
否
230
继续在新地址
空间内的执行
是
240
清除对应于存储
的所有TLB条目
245
继续在新地址
空间内的执行

图 4

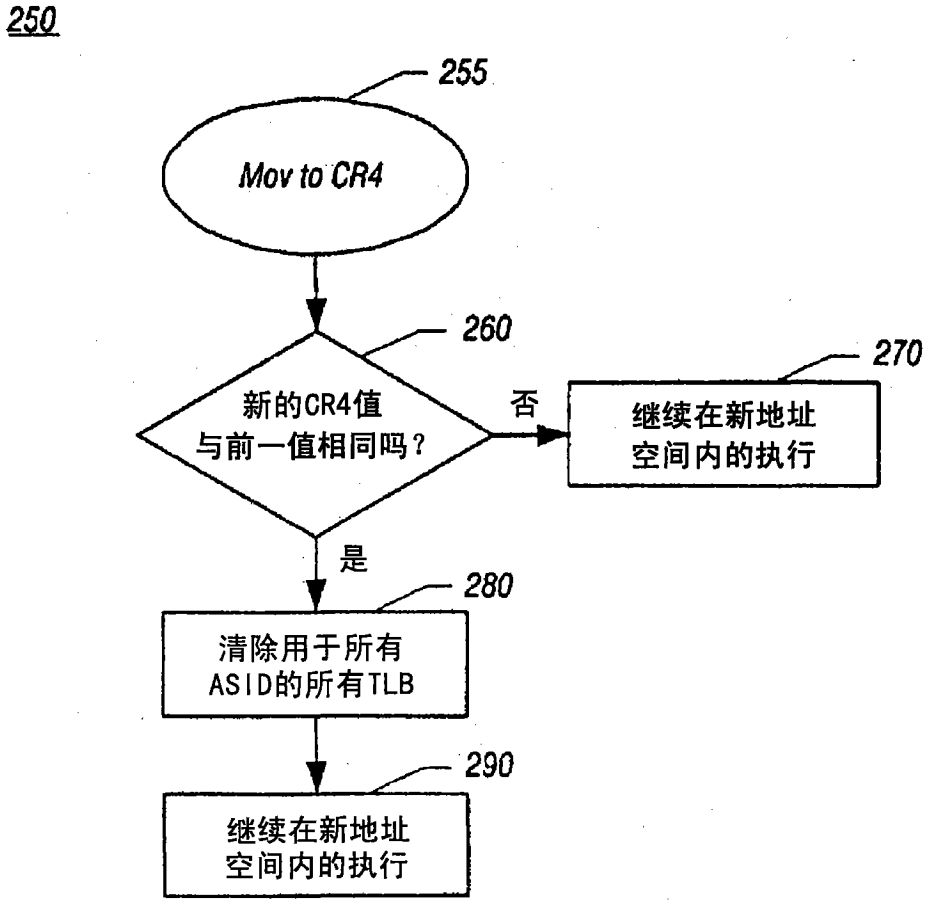

250
255
Mov to CR4
260
新的CR4值
与前一值相同吗？
否
270
继续在新地址
空间内的执行
是
280
清除用于所有
ASID的所有TLB
290
继续在新地址
空间内的执行

图 5

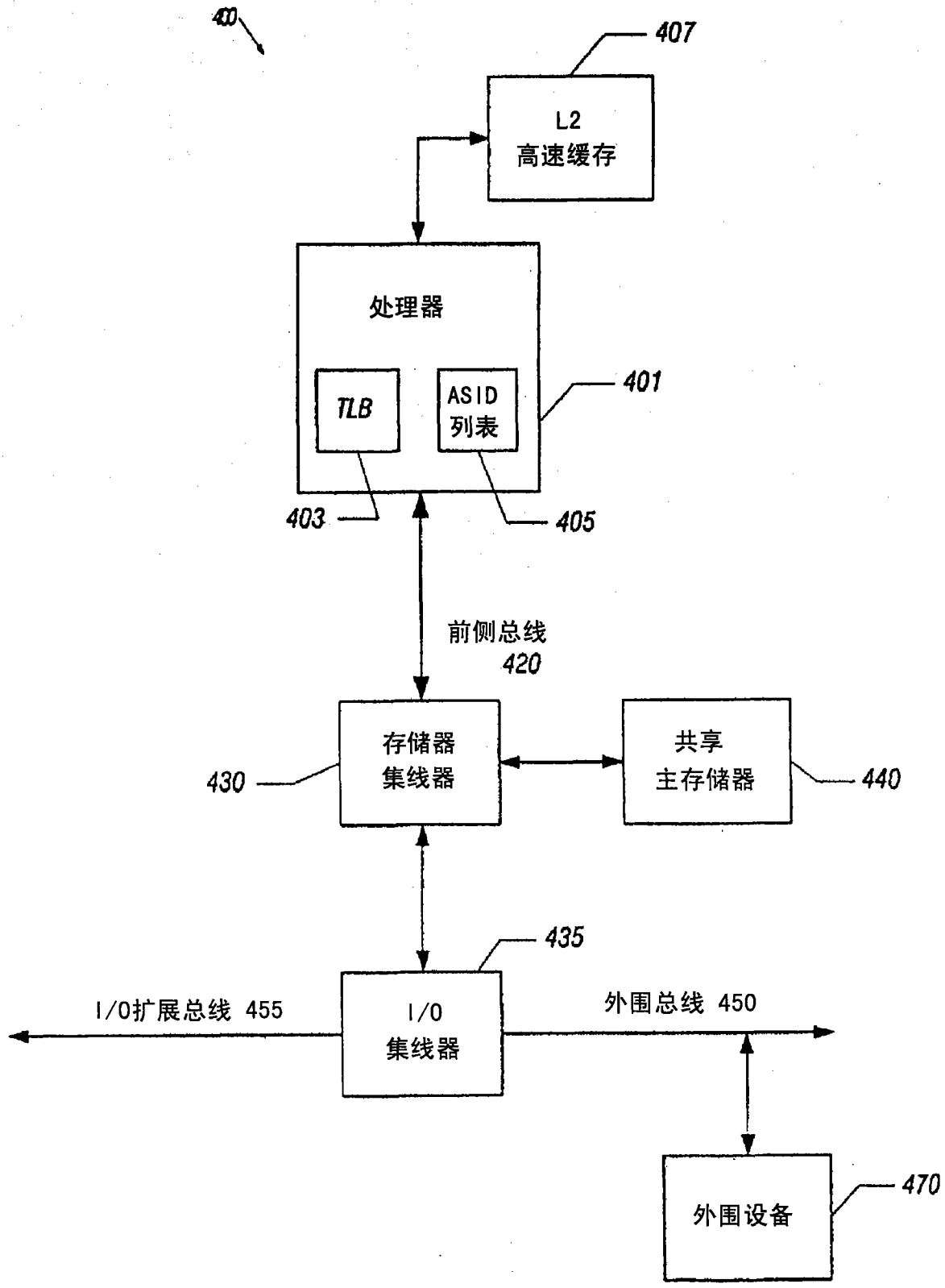

400
407
L2
高速缓存
处理器
TLB
ASID
列表
401
403
405
前侧总线
420
存储器
集线器
430
共享
主存储器
440
435
I/O扩展总线 455
I/O
集线器
外围总线 450
外围设备
470

图 6