(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 103270489 B
(45)授权公告日 2016.03.02

(21)申请号 201180062099.2
(22)申请日 2011.11.09
(30)优先权数据
12/976,981 2010.12.22 US
(85)PCT国际申请进入国家阶段日
2013.06.21
(86)PCT国际申请的申请数据
PCT/US2011/060011 2011.11.09
(87)PCT国际申请的公布数据
WO2012/087446 EN 2012.06.28
(73)专利权人 英特尔公司
地址 美国加利福尼亚州
(72)发明人 B·V·帕特尔 G·尼格
M·G·迪克森 J·S·科克
J·B·克罗斯兰
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 毛力
(51)Int.Cl.
G06F 9/30(2006.01)
G06F 9/06(2006.01)
(56)对比文件
US 5832299 A,1998.11.03,
US 6880068 B1,2005.04.12,
US 2008016327 A1,2008.01.17,
US 6507904 B1,2003.01.14,
CN 1495605 A,2004.05.12,
审查员 刘细金

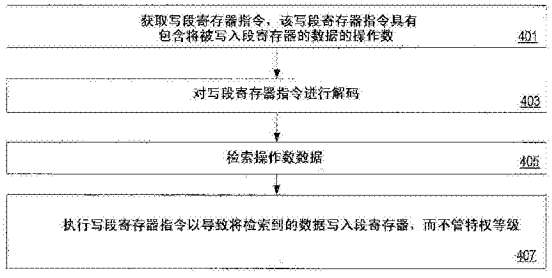
权利要求书1页 说明书8页 附图7页

(54)发明名称

用于进行段寄存器读和写而不管特权等级的系统、装置和方法

(57)摘要

描述了用于执行特权不可知的段基寄存器读或写指令的系统、装置和方法的实施例。示例性方法可包括获取特权不可知的段基寄存器写指令，其中该特权不可知的写指令包括64位数据源操作数，对所获取的特权不可知的段基寄存器写指令进行解码，以及执行经解码的特权不可知的段基寄存器写指令以便将源操作数的64位数据写入由该特权不可知的段基寄存器写指令的操作码标识的段基寄存器中。



CN 103270489 B

1. 一种在计算机处理器中执行特权不可知的段基寄存器写指令的方法，包括：

获取所述特权不可知的段基寄存器写指令，其中所述特权不可知的段基寄存器写指令包括 64 位数据源操作数；

对所获取的特权不可知的段基寄存器写指令进行解码；

执行经解码的特权不可知的段基寄存器写指令，以便将所述源操作数的 64 位数据写入由所述特权不可知的段基寄存器写指令的操作码所标识的段基寄存器中。

2. 如权利要求 1 所述的方法，其特征在于，所述段基寄存器是 IA32_FS_BASE。

3. 如权利要求 1 所述的方法，其特征在于，所述段基寄存器是 IA32_GS_BASE。

4. 如权利要求 1 所述的方法，其特征在于，所述特权不可知的段基寄存器写指令是特权等级 3 程序的一部分。

5. 如权利要求 1 所述的方法，其特征在于，还包括：

通过检查所述计算机处理器的 CPUID 特征标志来确定所述计算机处理器能够支持所述特权不可知的段基寄存器写指令。

6. 如权利要求 5 所述的方法，其特征在于，还包括：

在所述计算机处理器中设置指示支持所述特权不可知的段基寄存器写指令的标志。

7. 如权利要求 1 所述的方法，其特征在于，还包括：

通过检查所述计算机处理器的 CPUID 特征标志来确定所述计算机处理器不能够支持所述特权不可知的段基寄存器写指令；以及

在所述计算机处理器中设置指示不支持所述特权不可知的段基寄存器写指令的标志。

8. 一种在计算机处理器中执行特权不可知的段基寄存器读指令的方法，包括：

获取所述特权不可知的段基寄存器读指令，其中所述特权不可知的段基寄存器读指令包括 64 位数据目的地操作数；

对所获取的特权不可知的段基寄存器读指令进行解码；

执行经解码的特权不可知的段基寄存器读指令，以便读取由所述特权不可知的段基寄存器的操作码所标识的段基寄存器中的 64 位数据，并将该 64 位数据存储到由所述 64 位数据目的地操作数所标识的位置中。

9. 如权利要求 8 所述的方法，其特征在于，所述段基寄存器是 IA32_FS_BASE。

10. 如权利要求 8 所述的方法，其特征在于，所述段基寄存器是 IA32_GS_BASE。

11. 如权利要求 8 所述的方法，其特征在于，所述特权不可知的段基寄存器读指令是特权等级 3 程序的一部分。

12. 如权利要求 8 所述的方法，其特征在于，还包括：

通过检查所述计算机处理器的 CPUID 特征标志来确定所述计算机处理器能够支持所述特权不可知的段基寄存器读指令。

13. 如权利要求 12 所述的方法，其特征在于，还包括：

在所述计算机处理器中设置指示支持所述特权不可知的段基寄存器读指令的标志。

14. 如权利要求 8 所述的方法，其特征在于，还包括：

通过检查所述计算机处理器的 CPUID 特征标志来确定所述计算机处理器不能够支持所述特权不可知的段基寄存器读指令；

在所述计算机处理器中设置指示不支持所述特权不可知的段基寄存器读指令的标志。

用于进行段寄存器读和写而不管特权等级的系统、装置和方法

发明领域

[0001] 本发明领域一般涉及计算机处理器架构，尤其涉及在被执行时产生特定结果的指令。

[0002] 背景

[0003] 许多现代处理器具有不同的特权等级，这些特权等级确定什么能够和不能够被特定实体执行。对此的一个常见观点将特权等级分成不同的环。如图 1 所示，环 0 是最高特权等级，并因此在环 0（特权等级 0 或内核模式）中运行的程序能够用系统做任何事情，而在环 3（用户模式）中运行的代码具有较少的特权，并因此其能力受限。该特权分割针对另一个环 3 应用保护一个环 3 应用，并针对环 3 应用保护环 0 内核。环 3 应用应当能够在不影响计算机系统的其余应用的情况下在任何时刻失效，因为保持计算机运行的关键任务对环 0 内核而不是环 3 应用可用。环 1 和环 2 比环 0 更多受限，但比环 3 更少受限。

[0004] 这些等级提供针对软件对系统环境（以及对应的系统安全分支）的意外或蓄意破坏的硬件保护。只有系统软件的“受信”部分被允许在内核模式的不受限环境中执行，并且仅仅当绝对必要时才如此。所有其他软件在一个或多个用户模式中执行。

[0005] 附图简述

[0006] 在附图各图中通过示例而不是限制说明了本发明，其中相同标记指示相同元件，且其中：

[0007] 图 1 示出了常见的特权等级架构。

[0008] 图 2 示出了设置支持指示的方法的实施例。

[0009] 图 3 示出了段基寄存器读 / 写的方法的实施例。

[0010] 图 4 示出了处理特权不可知的段基寄存器写指令的方法的实施例。

[0011] 图 5 示出了处理特权不可知的段基寄存器读指令的方法的实施例。

[0012] 图 6 是示出根据本发明的各实施例的示例性无序核架构的框图。

[0013] 图 7 示出了根据本发明的一个实施例的系统的框图。

[0014] 图 8 示出了根据本发明的一个实施例的第二系统的框图。

[0015] 图 9 示出了根据本发明的一个实施例的第三系统的框图。

[0016] 详细描述

[0017] 在以下描述中，陈述了多个具体细节。然而，应当理解，本发明的实施例可在没有这些具体细节的情况下实践。在其他实例中，公知的电路、结构和技术未被详细示出以免混淆对本描述的理解。

[0018] 说明书中对“一个实施例”、“实施例”、“示例实施例”等等的引用表示所描述的实施例可包括特定特征、结构或特性，但是，每一个实施例可以不一定包括该特定特征、结构，或特征。此外，这样的短语不一定是指同一个实施例。此外，当结合实施例描述特定特征、结构或特性时，认为本领域技术人员知道结合无论是否明显描述的其它实施例来实现这些特征、结构或特性。

[0019] 当今，即使不是全部，大多数 OS 支持单个地址空间中的多个执行线程。处理器、CPU 或核心中的线程专用信息被称为线程上下文。操作系统负责确保在一 CPU 上运行线程之前该特定 CPU 上存在正确的线程上下文。当操作系统切换线程上下文作为从一个线程切换到另一个线程的一部分时，这通常被称为上下文切换。操作系统还提供用于提供被称为线程专用存储的执行线程专用存储的某种机制。在 x86 处理器上，使用 FS 和 / 或 GS 段寄存器来提供对线程专用存储的访问。通过使用段寄存器，每一线程都能够在访问数据时使用同一段和偏移，但访问对该线程是唯一的数据集。这通常是许多 IA32 和 Intel ® 64 位操作系统或其等效物中的基于非零的非最大限制段的唯一用途。段描述符是线程上下文的一部分并且必须与线程上下文的其余部分一起进行上下文切换。许多操作系统在系统中的所有线程中共享相同的环 0 或内核地址空间。这些操作系统通常将具有与用于环 3 或用户模式的线程专用存储不同的用于内核的线程专用存储。在内核中，线程专用存储实际上将会是逻辑处理器专用存储，并且将包含对该处理器唯一的信息。这些操作系统必须在环转换时改变段描述符并且修改这些段描述符以供在上下文交换代码中的用户代码中使用。

[0020] 当引入 x86 架构的 64 位变体（Intel ® 64）时，除了 FS 和 GS 段寄存器之外，从 64 位模式中移除分段。段描述符未被增强以支持较大的地址空间。这些段描述符作为 8 字节描述符被留下并且只支持 32 位基址。许多操作系统将虚拟地址空间分成上面的内核地址空间和下面的用户地址空间。另外，内核逻辑处理器专用数据需要在内核地址空间中，而线程专用数据需要在用户地址空间中。即使用户和内核地址空间被翻转，也无法用 32 位基址来实现。为了允许这些段被放置在任何位置，64 位模式中所支持的 48 位虚拟地址空间需要一种替换机制。所实现的机制是提供映射对应段的基址的两个 MSR，即 IA32_FS_BASE MSR(C0000100H) 和 IA32-GS_BASE MSR(C0000101H)。段的基址可通过简单地写入这些 MSR 来改变。RDMSR 和 WRMSR 是允许写入 64 位的特权指令，对用户级代码不可用。SWAPGS 指令还简化了环转换时的内核模式和用户模式 GS 段之间的切换。指令 RDMSR、WRMSR 和 SWAPGS 被限于环 0 或内核模式以确保环 3 应用无法操纵对整个操作系统的运行是至关重要的 MSR。上述指令被限于环 0 这一事实，该限制人为地将环 3 应用的能力限于修改 FS 和 GS 段。因此，对于当前 IA 系统，应用被限制进行系统调用或改变段描述符以切换 FS 或 GS 基址。系统调用是缓慢的，因为它涉及环转换，并且由于段描述符只保存 32 位段基址（如上所述），因此将这两个段基址的值限于最低 4GB （64 位系统中的可用地址空间中的非常小的部分）并且将段描述符的总数限于小于 8K。传统的 x86 处理器还包括两个系统表寄存器，即全局描述符表寄存器（GDTR）和局部描述符表寄存器（LDTR）。这两个寄存器都存储 32 或 64 位线性基址。

[0021] 对于在上下文切换之间只做出少量工作的线程，系统调用、上下文切换以及到用户模式代码的后续返回经常是该线程在它每次运行时所完成的工作的主要部分。针对该问题的一种可能的解决方案是用户模式线程，其中线程之间的上下文切换在用户模式代码中完成。这些用户模式线程中的每一个都需要其自己唯一的线程专用存储副本。在没有其自己的唯一副本的情况下，用户模式线程必须被限于单个 OS 线程（线程专用存储将不会在它们在 OS 线程之间移动时跟随它们），并且然后存在使用同一存储的彼此不知道的多个实体的问题。

[0022] 在大多数现有系统中，从“用户模式”切换到“内核模式”是非常昂贵的。由此，上

述的从段寄存器的读取 / 向段寄存器的写入的方式是不理想的。以下详述的是可用于执行用于对诸如 IA32_GS_BASE 或 IA32_FS_BASE 等段基寄存器进行读和写而不管 64 位模式中的当前特权等级的指令的系统、体系结构等的实施例。这些指令可以是特权等级 3 程序的一部分。

[0023] 特权不可知的写指令的示例是“WRGSBASE RAX”，其中“RAX”是包含将被写入 GS 段基寄存器的 64 位数据值的寄存器或存储器操作数。WRGSBASE 是操作码并且标识 IA32_GS_BASE 寄存器。该指令的执行将 RAX 寄存器的数据写入 GS 段基寄存器。类似地，WRFSBASE RAX 是在被执行时将 RAX 的数据写入 FS 段寄存器（IA32_FS_BASE）的指令。

[0024] 特权不可知的读指令的示例是“RDGSBASE RAX”，其中“RAX”是将存储从 GS 段基寄存器读取的 64 位数据的寄存器或存储器操作数。RDGSBASE 是操作码并且标识 IA32_GS_BASE 寄存器。该指令的执行导致将 GS 段基寄存器的值读取到 RAX 寄存器中。类似地，RDFSBASE RAX 是在被执行时从 FS 段基寄存器（IA32_FS_BASE）读取其数据并将该数据存储在 RAX 中的指令。

[0025] 在一些实施例中，在执行特权不可知的段基寄存器读 / 写指令之前，操作系统、处理器或芯片组中的一个设置该处理器能够支持这一指令的指示。设置该指示的方法的实施例被图 2 中示出。

[0026] 在 201 确定处理器的逻辑处理器是否支持以下操作的执行 ：从处理器段基寄存器中的一个或多个读取和 / 或向处理器段基寄存器中的一个或多个写入 64 位值而不管特权等级。通常，该判定通过检查处理器的指示存在支持的 CPUID 特征标志来完成。

[0027] 如果不存在支持，则在 203 使用从段基寄存器读取 / 向段基寄存器写入的较不高效的手段。如果存在支持，则在控制寄存器中设置指示对这些类型的指令的支持的至少一个标志。例如，在一些实施例中，控制寄存器 4 具有为该指示设置的标志。当然，其他寄存器也可用于该目的。

[0028] 图 3 示出了段基寄存器读 / 写的方法的实施例。在 301，确定是否启用特权不可知的基本寄存器读 / 写指令。例如，如果使用诸如 x86 架构中的控制寄存器 4 等控制寄存器，则检查对应标志以查看支持什么。如果支持特权不可知的段基寄存器读 / 写指令，则将在 313 在例如获取这些指令时处理这些指令。

[0029] 如果不支持特权不可知的段基寄存器读 / 写指令，则在 303 确定逻辑处理器所处的特权等级是否是特权等级 0。如上所详述的，特权等级 0 意味着对于能够运行什么有极少（如果有的话）限制。如果这是特权等级，则在 305，可通过使用相应的 RDMSR 或 WRMSR 特权指令从 IA32_GS_BASE 或 IA32_FS_BASE 进行读取 / 向 IA32_GS_BASE 或 IA32_FS_BASE 进行写入以在 64 位模式中更新 GS 和 FS 基址。

[0030] 如果特权等级高于 0，则在 307 进行到特权等级 0 的切换。一旦在特权等级 0 中，则在 309，通过使用相应的 RDMSR 或 WRMSR 特权指令从 IA32_GS_BASE 或 IA32_FS_BASE 进行读取 / 向 IA32_GS_BASE 或 IA32_FS_BASE 进行写入以在 64 位模式中更新 GS 和 FS 基址。在进行该动作之后，在 311 做出回到先前特权等级的切换。当然，这增加了较低的性能惩罚。

[0031] 图 4 示出了处理特权不可知的段基寄存器写指令的方法的实施例。在该示例性方法的任一步骤之前，可能已经存在支持指令的判定。在 401，接收特权不可知的段基寄存器写指令。该指令包括包含将被写入段基寄存器（例如，IA32_GS_BASE 或 IA32_FS_BASE）的

64 位数据的操作数。该操作数可以是存储器位置或寄存器，但更典型的是寄存器。

[0032] 特权不可知的段基寄存器写指令由解码逻辑在 403 进行解码并且在 405 检索操作数数据。例如，如果操作数是诸如 RAX 之类的寄存器，则检索来自寄存器的数据。

[0033] 特权不可知的段基寄存器写指令在 407 执行以使得将检索到的数据写入适当的段基寄存器，而不管当前特权等级。

[0034] 图 5 示出了处理特权不可知的段基本寄存器读指令的方法的实施例。在该示例性方法的任一步骤之前，已经存在支持指令的判定。在 501，接收特权不可知的段寄存器读指令。该指令包括标识要存储从段 MSR（例如，IA32_GS_BASE 或 IA32_FS_BASE）读取的 64 位数据的位置的操作数。该操作数可以是存储器位置或寄存器。

[0035] 特权不可知的段基寄存器读指令由解码逻辑在 503 解码。例如，如果操作数是诸如 RAX 等寄存器，则检索来自寄存器的数据。

[0036] 特权不可知的段基寄存器读指令在 505 执行以导致读取适当的段基寄存器中的数据，而不管特权等级。在 507，该数据然后被存储在操作数定义的位置。

[0037] 示例性计算机系统和处理器

[0038] 以下详述能够执行上述指令的装置和系统的实施例。图 6 是示出根据本发明的各实施例的示例性无序核体系结构的框图。然而，上述指令也可以在有序体系结构中执行。在图 6 中，箭头指示两个或更多个单元之间的耦合，且箭头的方向指示这些单元之间的数据流的方向。该体系结构中的组件可用于处理以上详述的指令，包括这些指令的获取、解码和执行。

[0039] 图 6 包括耦合到执行引擎单元 610 和存储器单元 615 的前端单元 605；执行引擎单元 610 还耦合到存储器单元 615。

[0040] 前端单元 605 包括耦合到二级（L2）分支预测单元 622 的一级（L1）分支预测单元 620。这些单元允许核获取并执行指令而不等待分支被解析。L1 和 L2 分支预测单元 620 和 622 耦合到 L1 指令高速缓存单元 624。L 指令高速缓存单元 624 保存将可能由执行引擎单元 610 执行的指令或一个或多个线程。

[0041] L1 指令高速缓存单元 624 耦合到指令转换后备缓冲器（ITLB）626。ITLB626 耦合到指令获取和预解码单元 628，该指令获取和预解码单元 628 将字节流分成分立指令。

[0042] 指令获取和预解码单元 628 耦合到指令队列单元 630 以存储这些指令。解码单元 632 解码包括上述指令的排队指令。在一些实施例中，解码单元 632 包括复杂解码器单元 634 和三个简单解码器单元 636、638 和 640。简单解码器可处理大多数（如果不是全部的话）x86 指令，其解码成单个微指令。复杂解码器可解码映射到多个微指令的指令。解码单元 632 还可包括微代码 ROM 单元 642。

[0043] L1 指令高速缓存单元 624 还耦合到存储器单元 615 中的 L2 高速缓存单元 648。指令 TLB 单元 626 还耦合到存储器单元 615 中的二级 TLB 单元 646。解码单元 632、微代码 ROM 单元 642 和环流检测器（LSD）单元 644 各自耦合到执行引擎单元 610 中的重命名 / 分配器单元 656。LSD 单元 644 检测何时执行软件中的环，停止预测分支（及可能不正确预测环的最后分支）以及其外部的流指令。在一些实施例中，LSD644 高速缓存微操作。

[0044] 执行引擎单元 610 包括耦合到退役单元 674 和统一调度器单元 658 的重命名 / 分配器单元 656。重命名 / 分配器单元 656 在任何寄存器重命名之前确定所需资源并分配可

用资源用于执行。该单元还将逻辑寄存器重命名至物理寄存器文件的物理寄存器。

[0045] 退役单元 674 还耦合到执行单元 660 且包括重排序缓冲器单元 678。该单元在指令完成时使指令退役。

[0046] 统一调度器单元 658 还耦合到物理寄存器文件单元 676，物理寄存器文件单元 676 耦合到执行单元 660。该调度器在处理器上运行的不同线程之间被共享。

[0047] 物理寄存器文件单元 676 包括 MSR 单元 677A、浮点寄存器单元 677B 和整数寄存器单元 677C，且可包括未示出的附加寄存器文件(例如，混叠在 MMX 打包整数平面寄存器文件 550 上的标量浮点栈寄存器文件)。MSR 单元包括 IA32_GS_BASE 和 IA32_FS_BASE 寄存器。

[0048] 执行单元 660 包括三个混合标量和 SIMD 执行单元 662、664 和 672；负载单元 666；存储地址单元 668；存储数据单元 670。负载单元 666、存储地址单元 668 和存储数据单元 670 执行负载 / 存储和存储器操作，且各自进一步耦合到存储器单元 615 中的 TLB 单元 652。

[0049] 存储器单元 615 包括耦合到数据 TLB 单元 6452 的二级 TLB 单元 646。数据 TLB 单元 652 耦合到 L1 数据高速缓存单元 654。L1 数据高速缓存单元 654 还耦合到 L2 高速缓存单元 648。在一些实施例中，L2 高速缓存单元 648 还耦合到存储器单元 615 内部和 / 或外部的 L3 和更高级的高速缓存单元 650。

[0050] 以下是适用于执行本文详述的指令的示例性系统。对于膝上计算机、台式机、手持 PC、个人数字助理、工程师工作站、服务器、网络设备、网络集线器、交换器、嵌入式处理器、数字信号处理器(DSP)、图形设备、视频游戏设备、机顶盒、微控制器、蜂窝电话、便携式媒体播放器、手持设备以及各种其它电子设备，其它业内已知的系统设计和配置也是适用的。一般而言，本文中公开的各种能够合并处理器和 / 或其它执行逻辑的系统或电子设备一般是适用的。

[0051] 现在参考图 7，所示出的是根据本发明一实施例的系统 700 的框图。系统 700 可包括耦合至图形存储器控制器中枢(GMCH)720 的一个或多个处理元件 710、715。附加的处理元件 715 的任选性在图 7 中通过虚线来表示。

[0052] 每个处理元件可以是单核，或可替代地包括多核。处理元件可任选地包括除处理核之外的其它片上元件，诸如集成存储器控制器和 / 或集成 I/O 控制逻辑。此外，对于至少一个实施例，处理元件的(多个)核可多线程化，因为它们对每个核可包括一个以上的硬件线程上下文。

[0053] 图 7 示出 GMCH720 可耦合至存储器 740，该存储器 740 可以是例如动态随机存取存储器(DRAM)。对于至少一个实施例，DRAM 可以与非易失性高速缓存相关联。

[0054] GMCH720 可以是芯片组或芯片组的一部分。GMCH720 可以与(多个)处理器 710、715 进行通信，并控制处理器 710、715 和存储器 740 之间的交互。GMCH720 还可担当(多个)处理器 710、715 和系统 700 的其它元件之间的加速总线接口。对于至少一个实施例，GMCH720 经由诸如前端总线(FSB)795 之类的多点总线与(多个)处理器 710、715 进行通信。

[0055] 此外，GMCH720 耦合至显示器 745(诸如平板显示器)。GMCH720 可包括集成图形加速器。GMCH720 还耦合至输入 / 输出(I/O)控制器中枢(ICH)750，该输入 / 输出(I/O)控制器中枢(ICH)750 可被用于将各种外围设备耦合至系统 700。在图 7 的实施例中作为示例示出了外部图形设备 760 以及另一外围设备 770，该外部图形设备 760 可以是耦合至 ICH750 的分立图形设备。

[0056] 替代地，系统 700 中还可存在附加或不同的处理元件。例如，附加(多个)处理元件 715 可包括与处理器 710 相同的附加处理器、与处理器 710 异类或不对称的附加(多个)处理器、加速器(诸如例如图形加速器或数字信号处理(DSP)单元)、现场可编程门阵列或任何其它处理元件。按照包括体系结构、微体系结构、热、功耗特征等等优点的度量谱，物理资源 710、715 之间存在各种差别。这些差别会有效显示为处理元件 710、715 之间的不对称性和异类性。对于至少一个实施例，各种处理元件 710、715 可驻留在同一管芯封装中。

[0057] 现在参照图 8，所示出的是根据本发明一实施例的第二系统 800 的框图。如图 8 所示，多处理器系统 800 是点对点互连系统，并且包括经由点对点互连 850 耦合的第一处理元件 870 和第二处理元件 880。如图 8 所示，处理元件 870 和 880 中的每一个都可以是多核处理器，包括第一和第二处理器核(即，处理器核 874a 与 874b 以及处理器核 884a 与 884b)。

[0058] 替代地，处理元件 870、880 中的一个或多个可以是除处理器之外的元件，诸如加速器或现场可编程门阵列。

[0059] 虽然仅以两个处理元件 870、880 来示出，但应理解本发明的范围不限于此。在其它实施例中，在给定处理器中可存在一个或多个附加处理元件。

[0060] 第一处理元件 870 还可包括存储器控制器中枢(MCH)872 和点对点(P-P)接口 876 和 878。类似地，第二处理元件 880 可包括 MCH882 与 P-P 接口 886 和 888。处理器 870、880 可以经由使用点对点 (PtP) 接口电路 878、888 的点对点 (PtP) 接口 850 来交换数据。如图 8 所示，MCH872 和 882 将处理器耦合到相应的存储器，即存储器 842 和存储器 844，这些存储器可以是本地附连到相应处理器的主存储器部分。

[0061] 处理器 870、880 可各自经由使用点对点接口电路 876、894、886、898 的单独 PtP 接口 852、854 与芯片组 890 交换数据。芯片组 890 还可经由高性能图形接口 839 与高性能图形电路 838 交换数据。本发明的实施方式可以置于具有任意数目的处理核的任意处理器中，或置于图 8 的 PtP 总线代理中的每一个中。在一个实施例中，任意处理器核可包括本地高速缓存存储器(未示出)或者以其它方式关联于本地高速缓存存储器(未示出)。此外，共享高速缓存(未示出)可被包括于在这两个处理器的外部但经由 p2p 互连与这些处理器连接的任一处理器中，从而如果一处理器被置于低功率模式，则任一个或这两个处理器的本地高速缓存信息可被存储在该共享的高速缓存中。

[0062] 第一处理元件 870 和第二处理元件 880 可分别经由 P-P 互连 876、886 和 884 耦合到芯片组 890。如图 8 所示，芯片组 890 包括 P-P 接口 894 和 898。此外，芯片组 890 包括将芯片组 890 与高性能图形引擎 848 耦合的接口 892。在一个实施例中，总线 849 可被用于将图形引擎 848 耦合到芯片组 890。替代地，点对点互连 849 可耦合这些部件。

[0063] 芯片组 890 又经由接口 896 耦合至第一总线 816。在一个实施例中，第一总线 816 可以是外围部件互连(PCI)总线或诸如 PCI Express 总线或另一第三代 I/O 互连总线之类的总线，虽然本发明的范围不限于此。

[0064] 如图 8 所示，各种 I/O 设备 814 可连同总线桥 818 一起耦合到第一总线 816，总线桥 818 将第一总线 816 耦合到第二总线 820。在一个实施例中，第二总线 820 可以是低引脚数(LPC)总线。多个设备可耦合至第二总线 820，包括例如键盘 / 鼠标 822、通信设备 826 以及数据存储单元 828(诸如盘驱动器或其它大容量存储设备，在一个实施例中其可包括代码 830)。此外，音频 I/O824 可耦合至第二总线 820。注意，其它体系结构是可能的。例如，

代替图 8 的点对点架构，系统可实施多点总线或另一此类架构。

[0065] 现在参照图 9，所示出的是根据本发明实施例的第三系统 900 的框图。图 8 和 9 中的相同元件使用相同附图标记，且在图 9 中省略了图 8 的某些方面以避免混淆图 9 的其它方面。

[0066] 图 9 示出处理元件 870、880 可分别包括集成存储器和 I/O 控制逻辑（“CL”）872 和 882。对于至少一个实施例，CL872、882 可包括诸如以上结合图 7 和 8 所描述的存储器控制器中枢逻辑（MCH）。此外，CL872、882 还可包括 I/O 控制逻辑。图 9 示出不仅存储器 842、844 耦合至 CL872、882，而且 I/O 设备 914 也耦合至控制逻辑 872、882。传统 I/O 设备 915 耦合至芯片组 890。

[0067] 本文中公开的机构的实施例可按照硬件、软件、固件或此类实现方法的组合来实现。本发明的实施例可被实现为在包括至少一个处理器、数据储存器系统（包括易失性和非易失性存储器和 / 或储存元件）、至少一个输入设备以及至少一个输出设备的可编程系统上执行的计算机程序。

[0068] 可将诸如图 8 中所示的代码 830 的程序代码应用于输入数据以执行本文中所描述的功能，并产生输出信息。可按照已知方式将输出信息应用于一个或多个输出设备。为了此应用的目的，处理系统包括具有诸如例如数字信号处理器（DSP）、微控制器、专用集成电路（ASIC）或微处理器之类的处理器的任意系统。

[0069] 程序可按照高级过程或面向对象的高级编程语言来实现，以与处理系统通信。程序代码在需要时还可按照汇编或机器语言来实现。实际上，本文中描述的机制在范围上不限于任何特定编程语言。在任何情况下，该语言可以是编译或解释语言。

[0070] 至少一个实施例的一个或多个方面可以由存储在机器可读介质上的代表性数据来实现，该数据表示处理器中的各种逻辑，其在被机器读取时使得该机器生成执行本文描述的技术的逻辑。被称为“IP 核”的这些表示可以被存储在有形的机器可读介质上，并被提供给各个顾客或生产设施以加载到实际制造该逻辑或处理器的制造机器中。

[0071] 此类机器可读存储介质可包括但不限于通过机器或设备制造或形成的粒子的有形排列，包括存储介质，诸如：硬盘；包括软盘、光盘、压缩盘只读存储器（CD-ROM）、可重写压缩盘（CD-RW）以及磁光盘的任何其它类型的盘；诸如只读存储器（ROM）之类的半导体器件；诸如动态随机存取存储器（DRAM）、静态随机存取存储器（SRAM）之类的随机存取存储器（RAM）；可擦除可编程只读存储器（EPROM）；闪存；电可擦除可编程只读存储器（EEPROM）；磁卡或光卡；或适于存储电子指令的任何其它类型的介质。

[0072] 因此，本发明的实施例也包括非瞬态有形机器可读介质，该介质包含诸如 HDL 之类的设计数据，该设计数据限定本文中描述的结构、电路、装置、处理器和 / 或系统特征。此类实施例也可被称为程序产品。

[0073] 本文公开的指令的某些操作可由硬件组件执行，且可体现在机器可执行指令中，该指令用于导致或至少致使电路或其它硬件组件以执行该操作的指令编程。电路可包括通用或专用处理器、或逻辑电路，这里仅给出几个示例。操作也可任选地由硬件和软件的组合来执行。执行逻辑和 / 或处理器可包括响应于从机器指令导出的机器指令或一个或多个控制信号以存储指令指定的结果操作数的专用或特定电路或其它逻辑。例如，本文公开的指令的实施例可在图 7、8 和 9 的一个或多个系统中执行，且指令的实施例可存储在将在系统

中执行的程序代码中。

[0074] 上述描述旨在说明本发明的优选实施例。根据上述讨论，还应当显而易见的是，在发展迅速且进一步的进展难以预见的此技术领域中，本领域技术人员可在安排和细节上对本发明进行修改，而不背离落在所附权利要求及其等价方案的范围内的本发明的原理。例如，方法的一个或多个操作可组合或进一步分开。

[0075] 可选实施例

[0076] 尽管已经描述了将自然执行本文公开的指令的实施例，但本发明的可选实施例可通过运行在执行不同指令集的处理器(例如，执行美国加利福亚州桑尼维尔的 MIPS 技术的 MIPS 指令集的处理器、执行加利福亚州桑尼维尔的 ARM 保持的 ARM 指令集的处理器) 上的仿真层来执行指令。同样，尽管附图中的流程图示出本发明的某些实施例的特定操作顺序，按应理解该顺序是示例性的(例如，可选实施例可按不同顺序执行操作、组合某些操作、使某些操作重叠等)。

[0077] 在以上描述中，为解释起见，阐明了众多具体细节以提供对本发明的实施例的透彻理解。然而，将对本领域技术人员明显的是，在没有这些具体细节中的一些的情况下，也可实践一个或多个其他实施例。提供所描述的具体实施例不是为了限制本发明而是为了说明本发明的实施例。本发明的范围不是由前面提供的具体示例来确定的，而是仅由所附权利要求来确定的。

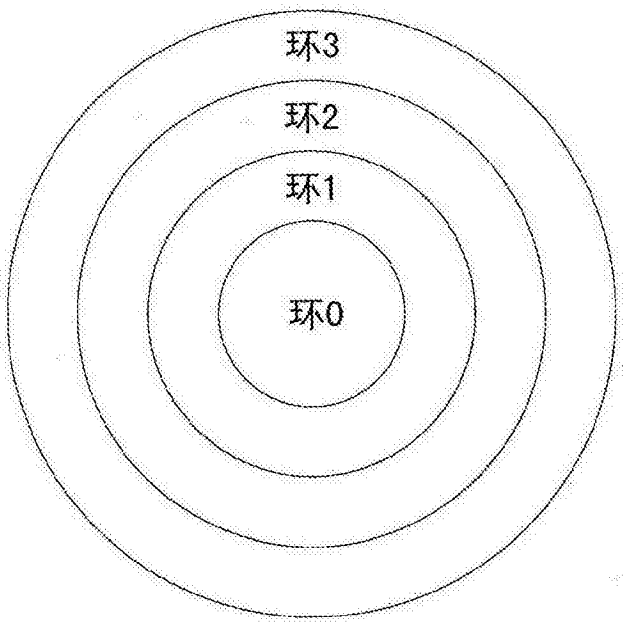

环3
环2
环1
环0

图 1

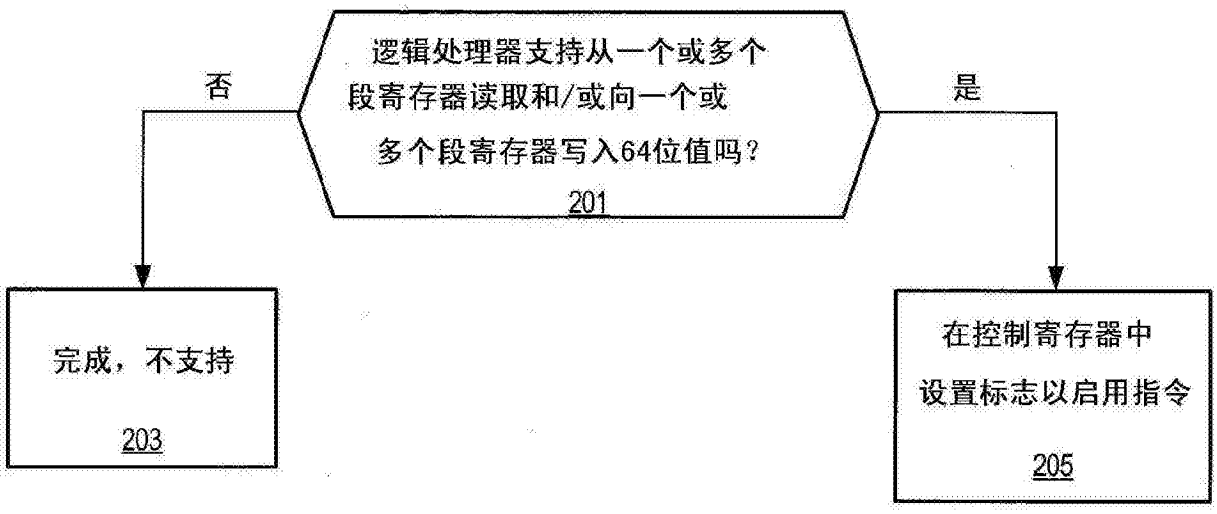

逻辑处理器支持从一个或多个段寄存器读取和/或向一个或多个段寄存器写入64位值吗？
201
否
是
完成，不支持
203
在控制寄存器中设置标志以启用指令
205

图 2

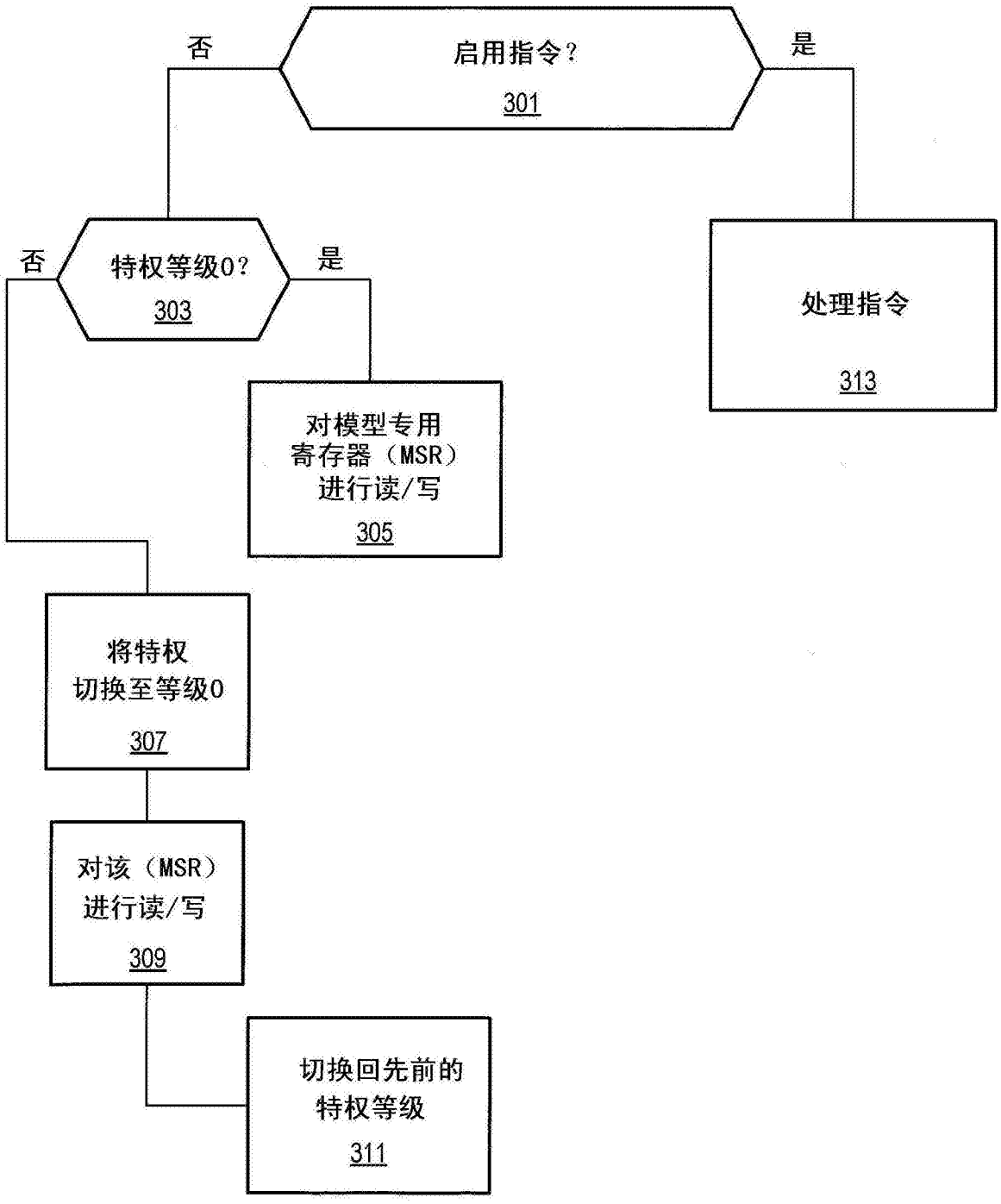

启用指令？
301
否
是
特权等级0？
303
否
是
处理指令
313
对模型专用
寄存器（MSR）
进行读/写
305
将特权
切换至等级0
307
对该（MSR）
进行读/写
309
切换回先前的
特权等级
311

图 3

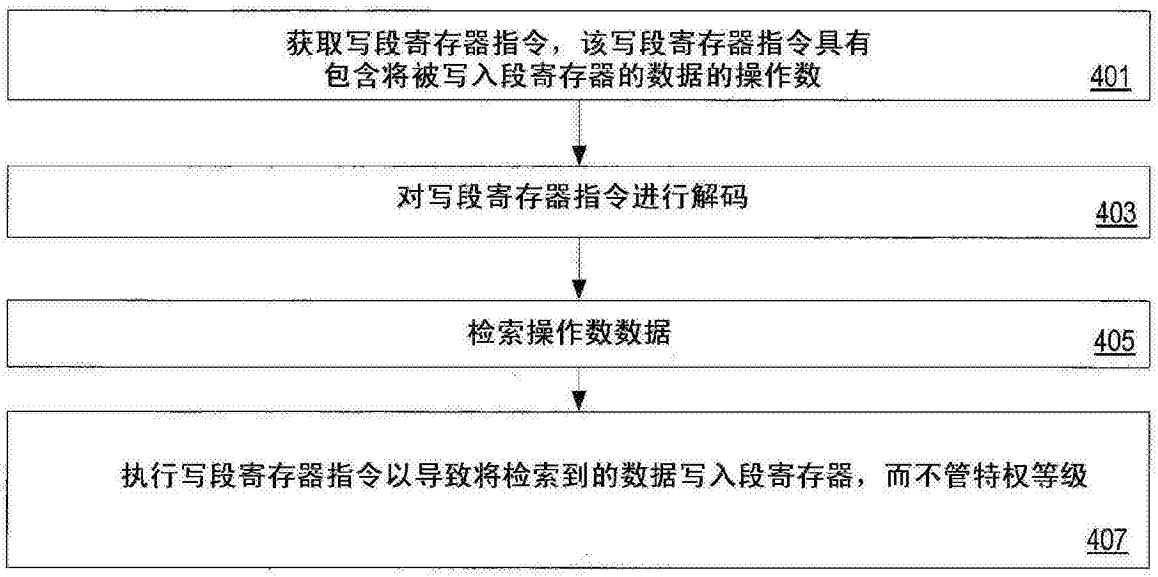

获取写段寄存器指令，该写段寄存器指令具有
包含将被写入段寄存器的数据的操作数
401
对写段寄存器指令进行解码
403
检索操作数数据
405
执行写段寄存器指令以导致将检索到的数据写入段寄存器，而不管特权等级
407

图 4

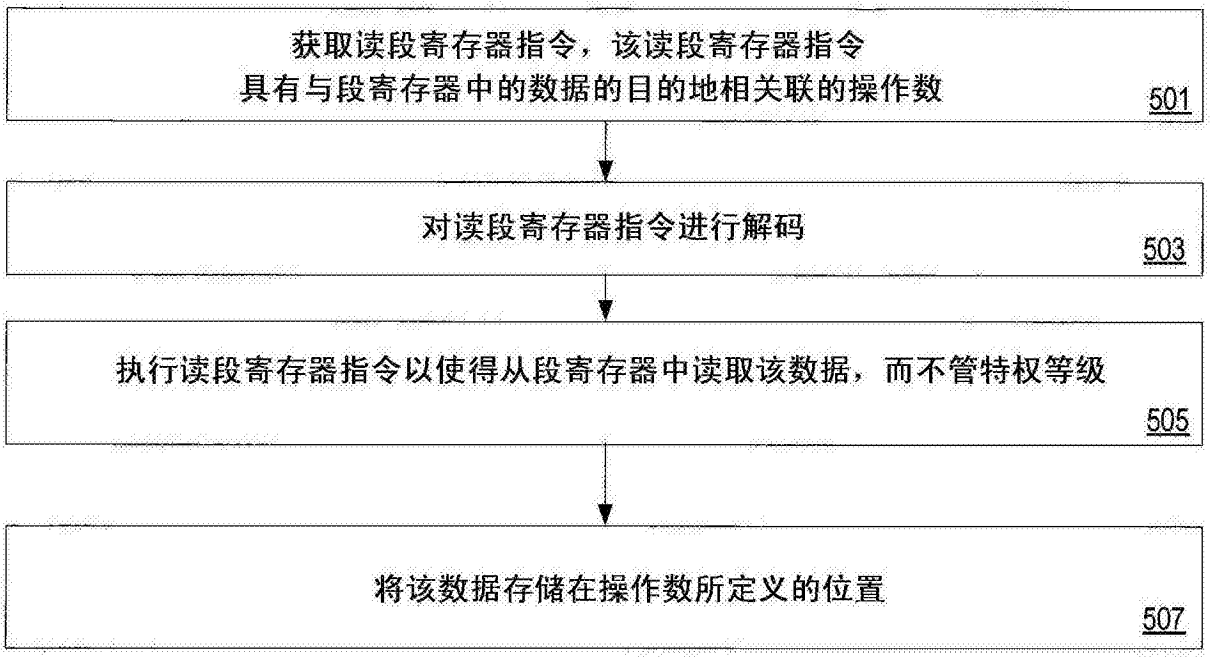

获取读段寄存器指令，该读段寄存器指令
具有与段寄存器中的数据的目的地相关联的操作数
501
对读段寄存器指令进行解码
503
执行读段寄存器指令以使得从段寄存器中读取该数据，而不管特权等级
505
将该数据存储在操作数所定义的位置
507

图 5

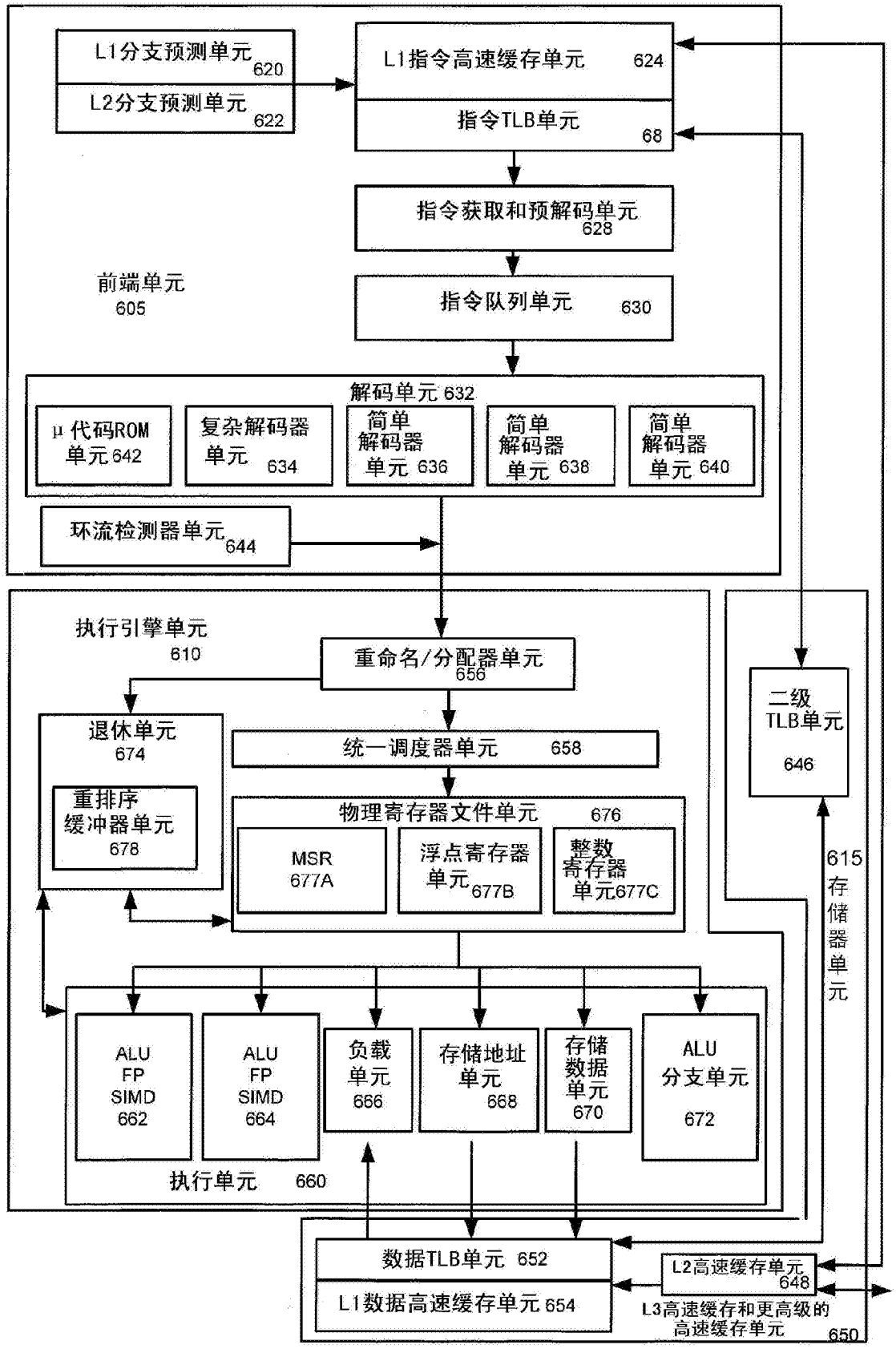

L1分支预测单元 620
L2分支预测单元 622
L1指令高速缓存单元 624
指令TLB单元 68
指令获取和预解码单元 628
前端单元 605
指令队列单元 630
解码单元 632
μ 代码ROM 单元 642
复杂解码器 单元 634
简单 解码器 单元 636
简单 解码器 单元 638
简单 解码器 单元 640
环流检测器单元 644
执行引擎单元 610
重命名/分配器单元 656
二级 TLB单元 646
退休单元 674
重排序 缓冲器单元 678
统一调度器单元 658
物理寄存器文件单元 676
MSR 677A
浮点寄存器 单元 677B
整数 寄存器 单元 677C
615 存储器单元
ALU FP SIMD 662
ALU FP SIMD 664
负载 单元 666
存储地址 单元 668
存储 数据 单元 670
ALU 分支单元 672
执行单元 660
数据TLB单元 652
L1数据高速缓存单元 654
L2高速缓存单元 648
L3高速缓存和更高级的 高速缓存单元 650

图 6

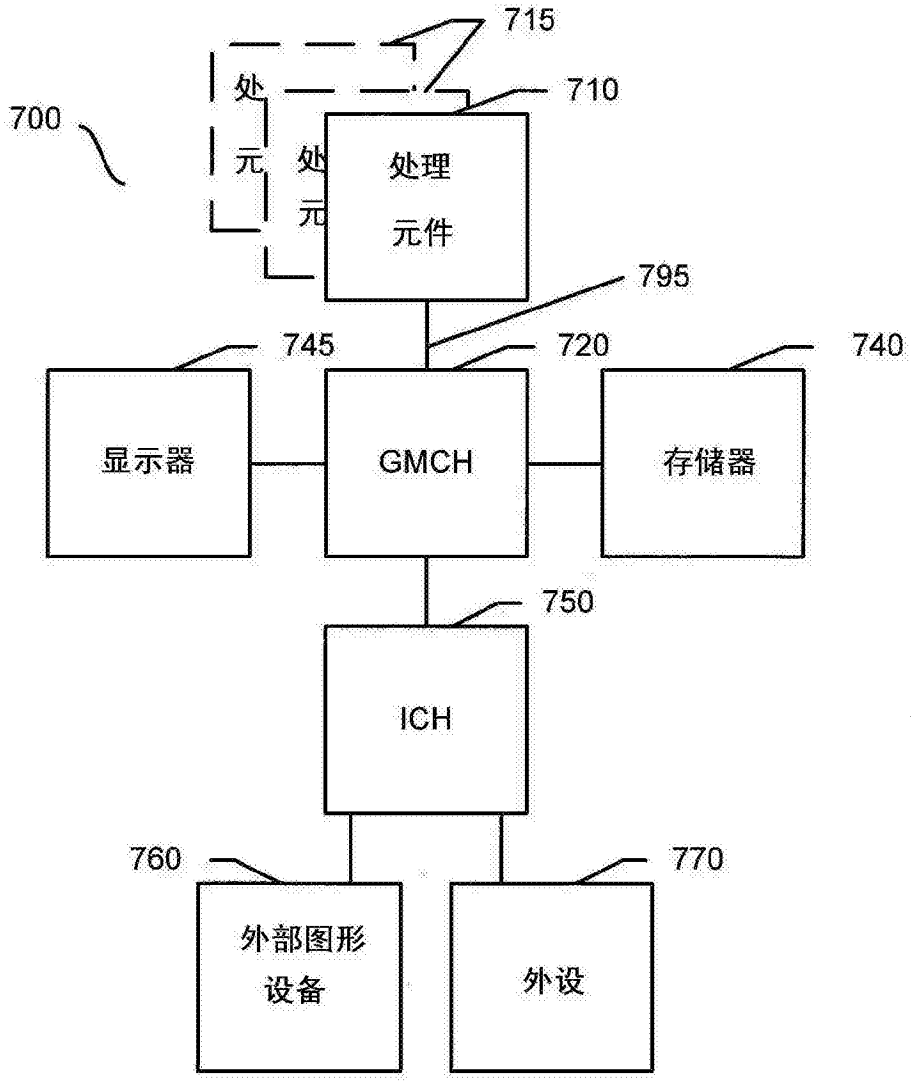

715
处
元
处
元
710
700
处理
元件
795
745
720
740
显示器
GMCH
存储器
750
ICH
760
770
外部图形
设备
外设

图 7

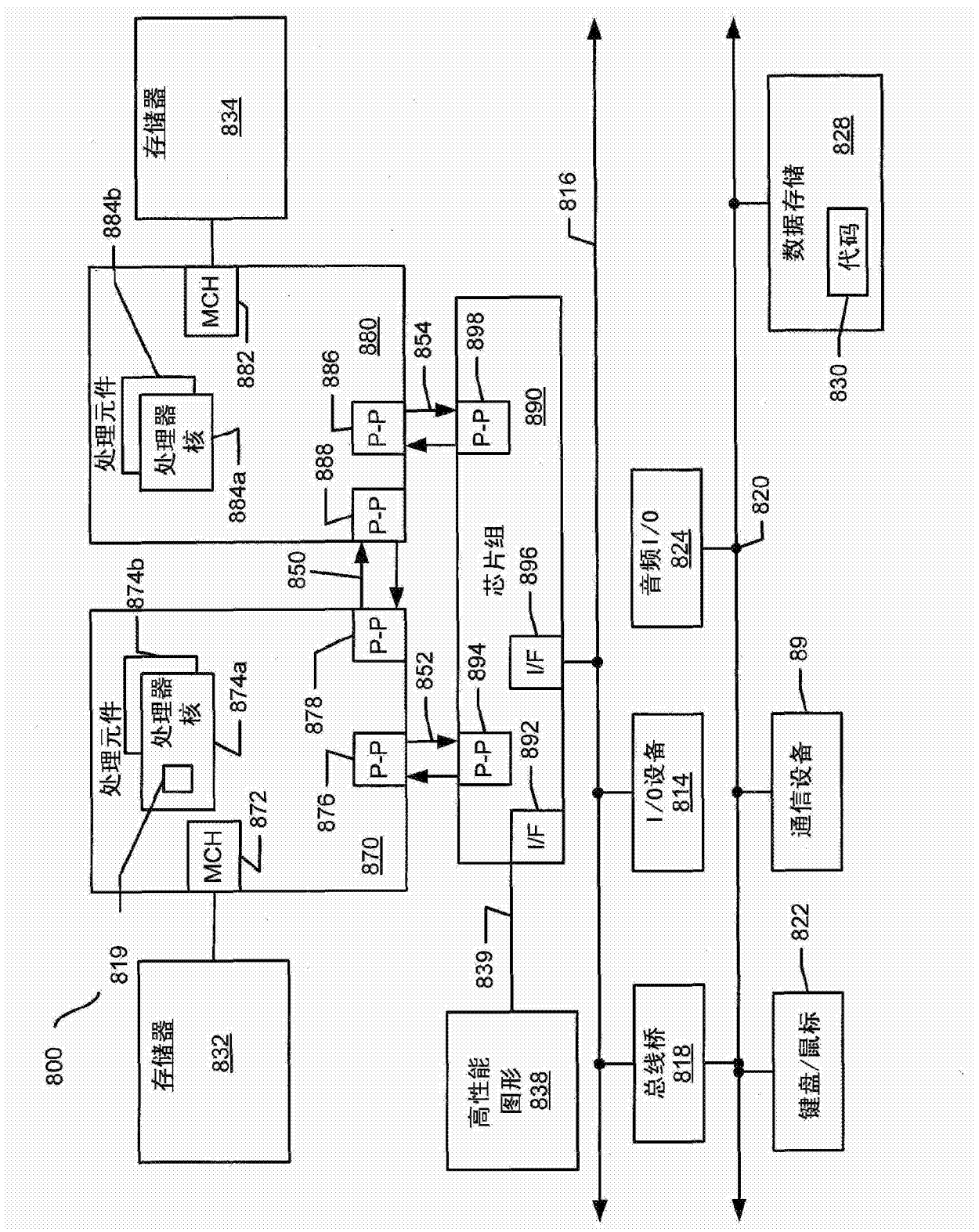

800
819
存储器
832
MCH
872
处理元件
处理器核
874a
874b
P-P
876
878
870
850
P-P
886
888
880
882
MCH
存储器
834
处理元件
处理器核
884a
884b
852
854
P-P
894
P-P
898
890
芯片组
I/F
892
I/F
896
839
高性能图形
838
816
I/O设备
814
总线桥
818
音频I/O
824
820
通信设备
89
键盘/鼠标
822
数据存储
828
代码
830

图 8

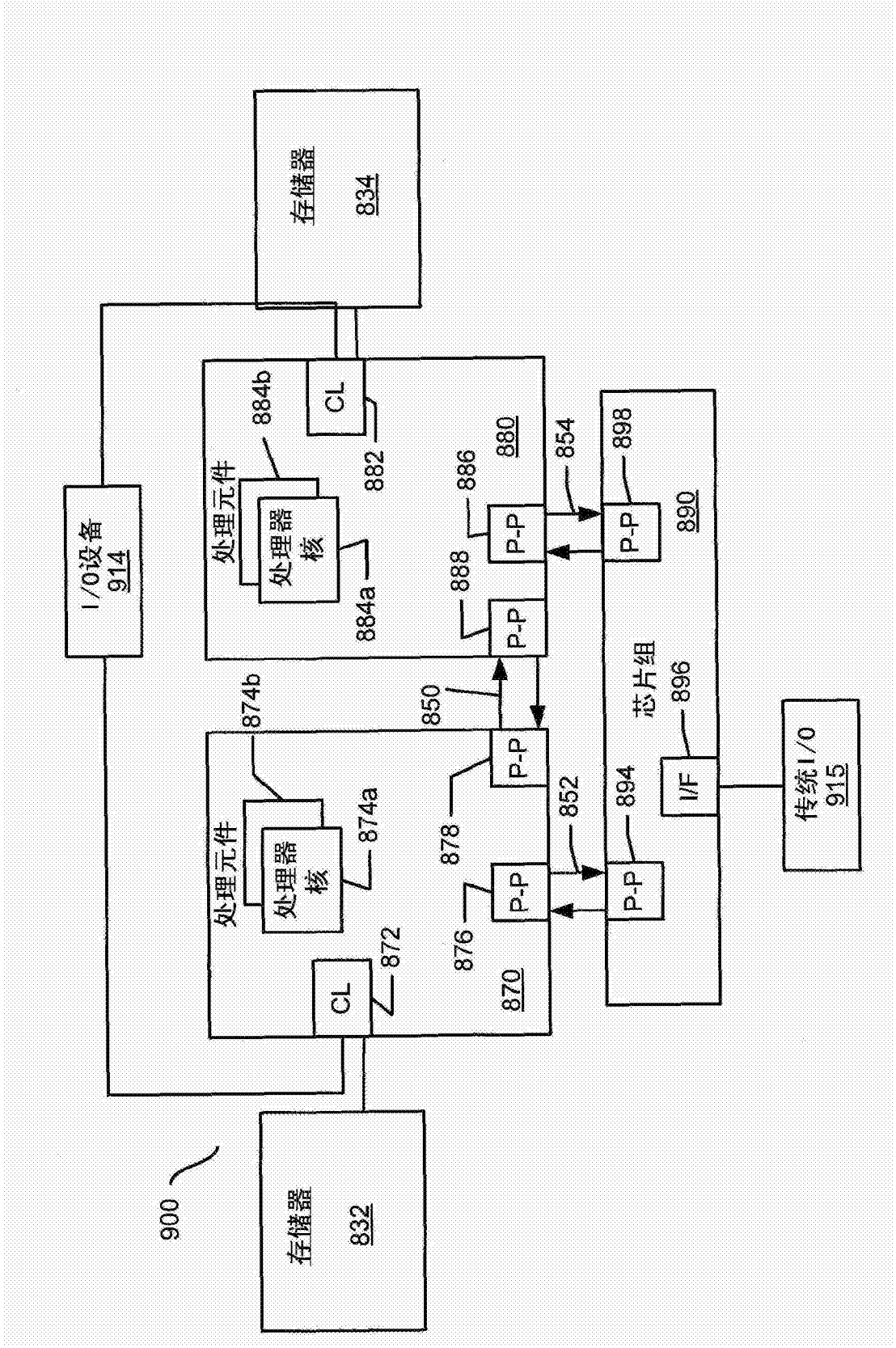

存储器
834
I/O设备
914
处理元件
884b
处理器
核
884a
CL
882
880
886
888
P-P
854
898
890
芯片组
850
874b
874a
878
852
894
876
872
870
896
I/F
传统I/O
915
900
存储器
832

图 9