[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/455 (2006.01)

[12] 发明专利说明书

专利号 ZL 200510076070.6

[45] 授权公告日 2009 年 8 月 19 日

[11] 授权公告号 CN 100530102C

[22] 申请日 2005.5.30
[21] 申请号 200510076070.6
[30] 优先权
[32] 2004.6.30 [33] US [31] 10/883,496
[73] 专利权人 微软公司
地址 美国华盛顿州
[72] 发明人 R·A·维嘉 E·P·托劳特
[56] 参考文献
US2003/0140245A1 2003.7.24
US6412043B1 2002.6.25
US5644755A 1997.7.1
US4965720 1990.10.23
审查员 王 洵

[74] 专利代理机构 上海专利商标事务所有限公司
代理人 钱慰民

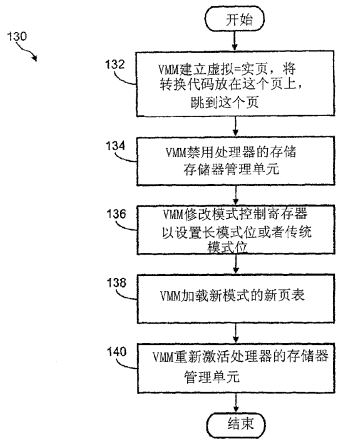
权利要求书 3 页 说明书 11 页 附图 6 页

[54] 发明名称

用于在 64 位 X86 处理器上运行传统 32 位 X86 虚拟机的系统和方法

[57] 摘要

本发明提供一种用于在 x86－64 体系结构中的长超级模式和传统超级模式之间实时转换的虚拟化计算系统和方法。 在这么做的时候，虚拟机，它依赖于传统 32 位模式，即实模式和保护模式(V86 子模式、环－0 子模式和环－3 子模式)，能够与在 x86－64 计算机硬件(即 64 位)上的其它应用程序并行运行。 执行临时处理器模式上下文切换的方法包括虚拟机监控程序设置一个“虚拟＝实”页、将执行处理器模式上下文切换的转换代码放在该页上、跳到该页、禁用 x86－64 计算机硬件的存储器管理单元(MMU)、修改模式控制寄存器来设置长超级模式位或传统超级模式位、加载新页表以及重新激活 x86－64 计算机硬件的 MMU。



1.一种用于执行多模式体系结构计算机系统中第一模式和第二模式之间的临时处理器模式上下文切换的方法，其特征在于，所述方法包括：

建立一个“虚拟=实”页，其中用于所述页的虚拟地址也是用于所述页的实存储器地址，其中所述“虚拟=实”页为“切换页”；

将执行处理器模式上下文切换的一组转换代码拷贝到所述切换页；

跳至所述切换页以继续执行；

禁用所述多模式体系结构计算机系统的存储器管理单元MMU；

修改模式控制寄存器来将上下文比特从所述第一模式设置成所述第二模式；

为所述第二模式加载一个新页表；以及

重新激活所述多模式体系结构计算机系统的MMU。

2.如权利要求1所述的方法，其特征在于，所述方法的至少一个步骤由虚拟机监控程序执行。

3.如权利要求1所述的方法，其特征在于，所述方法由虚拟机监控程序执行。

4.如权利要求1所述的方法，其特征在于，所述第一模式是X86-64体系结构处理器的长模式，而所述第二模式是所述X86-64体系结构处理器的传统模式。

5.如权利要求1所述的方法，其特征在于，所述第二模式是X86-64体系结构处理器的长模式，而所述第一模式是所述X86-64体系结构处理器的传统模式。

6.如权利要求1所述的方法，其特征在于，所述方法用来在X86-64体系结构处理器上执行传统模式中的32位环-0代码。

7.如权利要求1所述的方法，其特征在于，所述方法用来在X86-64体系结构处理器上执行传统模式中的32位v86模式代码。

8.一种用于执行多模式体系结构计算机系统中第一模式和第二模式之间的临时处理器模式上下文切换的系统，其特征在于，所述系统包括：

用于建立一个“虚拟=实”页的子系统，其中用于所述页的虚拟地址也是用于所述页的实存储器地址，其中所述“虚拟=实”页为“切换页”；

用于将执行处理器模式上下文切换的一组转换代码拷贝到所述切换页的子系统；

用于跳至所述切换页以继续执行的子系统；

用于禁用所述多模式体系结构计算机系统的存储器管理单元MMU的子系统；

用于修改模式控制寄存器来将上下文比特从所述第一模式设置成所述第二模式的子系统；

用于为所述第二模式加载一个新页表的子系统；以及

用于重新激活所述多模式体系结构计算机系统的MMU的子系统。

9.如权利要求8所述的系统，其特征在于，包括一个虚拟机监控程序。

10.如权利要求8所述的系统，其特征在于，所述系统是用于虚拟机监控程序的子系统。

11.如权利要求8所述的系统，其特征在于，所述第一模式是X86-64体系结构处理器的长模式，而所述第二模式是所述X86-64体系结构处理器的传统模式。

12.如权利要求8所述的系统，其特征在于，所述第二模式是X86-64体系结构处理器的长模式，而所述第一模式是所述X86-64体系结构处理器的传统模式。

13.如权利要求8所述的系统，其特征在于，所述系统用来在X86-64体系结构处理器上执行传统模式中的32位环-0代码。

14.如权利要求8所述的系统，其特征在于，所述系统用来在X86-64体系结构处理器上执行传统模式中的32位v86模式代码。

15.一种用于执行多模式体系结构计算机系统中第一模式和第二模式之间的临时处理器模式上下文切换的硬件控制设备，其特征在于，所述硬件控制设备包括：

用于建立一个“虚拟=实”页的装置，其中用于所述页的虚拟地址也是用于所述页的实存储器地址，其中所述“虚拟=实”页为“切换页”；

用于将执行处理器模式上下文切换的一组转换代码拷贝到所述切换页的装置；

用于跳至所述切换页以继续执行的装置；

用于禁用所述多模式体系结构计算机系统的存储器管理单元MMU的装置；

用于修改模式控制寄存器来将上下文比特从所述第一模式设置成所述第二模式的装置；

用于为所述第二模式加载一个新页表的装置；以及

重新激活所述多模式体系结构计算机系统的MMU的装置。

16.如权利要求15所述的硬件控制设备，其特征在于，还包括装置，用于使所述第一模式是X86-64体系结构处理器的长模式，而所述第二模式是所述X86-64体系结构处理器的传统模式。

17.如权利要求15所述的硬件控制设备，其特征在于，还包括装置，用于使所述第二模式是X86-64体系结构处理器的长模式，而所述第一模式是所述X86-64体系结构处理器的传统模式。

18.如权利要求15所述的硬件控制设备，其特征在于，还包括用于在X86-64体系结构处理器上执行传统模式中的32位环-0代码的装置。

19.如权利要求 15 所述的硬件控制设备，其特征在于，还包括用于在 X86-64 体系结构处理器上执行传统模式中的 32 位 v86 模式代码的装置。

用于在64位X86处理器上运行传统32位X86虚拟机的系统和方法

技术领域

本发明通常涉及虚拟机（也称为“处理器虚拟化”）和在虚拟机环境中执行的软件的领域。特别地，本发明直接涉及在不同于 64 位主机操作系统的处理器模式中运行传统32位虚拟机的方法。

背景技术

计算机包括被设计成执行特定的系统指令集的通用中央处理单元（CPU）。具有类似的体系结构或设计规范的一组处理器被认为是同一处理器家族的成员。当前的处理器家族的示例包括由亚利桑那州菲尼克斯市的 Motorola 公司生产的 Motorola 680X0 处理器家族；由加利福尼亚州桑尼维尔市的 Intel 公司生产的 Intel 80X86 处理器家族；以及由 Motorola 公司生产的，并在由加利福尼亚州库珀蒂诺市的 Apple 计算机公司生产的计算机中使用的 PowerPC 处理器家族。尽管一组处理器由于其类似的体系结构和设计考虑可处于同一家族中，然而根据其时钟速度和其它参数性能，家族中的处理器也会有很大的不同。

每一微处理器家族执行对该处理器家族唯一的指令。处理器或处理器家族可执行的一组集体的指令被称为处理器的指令集。作为一个示例，由 Intel 80X86 处理器家族使用的指令集与由 PowerPC 处理器家族使用的指令集不兼容。Inter 80X86 指令集基于复杂指令集计算机（CISC）格式。Motorola Power PC 指令集基于精简指令集计算机（RISC）格式。CISC 处理器使用大量的指令，其中一些可执行相当复杂的功能，但是它一般需要许多时钟周期来执行。RISC 处理器使用较少数量的可用指令，来执行以更高的速率执行的一组较简单的功能。

处理器家族在计算机系统之中的唯一性通常导致计算机系统的硬件体系结构的其它元件之间的不兼容性。用来自 Intel 80X86 处理器家族的处理器制造的计算机系统具有与用来自 PowerPC 处理器家族的处理器制造的计算机系统的硬件体系结构不同的硬件体系结构。由于处理器指令集以及计算机系统的硬件体系结构的唯一性，应用软件程序通常被书写成在运行特定操作系统的特定计算机系统上运行。

计算机制造商希望通过令更多而不是更少的应用程序运行在与计算机制造商的产品线相关联的微处理器家族上来将其市场份额最大化。为扩展可运行在计算机系统上的操作系统和应用程序的数量，开发了一种技术领域，其中，称为主机的具有一种类型 CPU 的给定计算机将包括一仿真器程序，它允许主机计算机仿真一种称为访客的 CPU 的不相关类型指令。由此，主机计算机将执行促使一个或多个主机指令响应于给定访客指令而被调用的应用程序。由此，主机计算机可运行为其自己的硬件体系结构设计的软件和为具有不相关硬件体系结构的计算机书写的软件两者。作为一个更具体的示例，例如，由 Apple 计算机制造的计算机系统可运行为基于 PC 的计算机系统书写的操作系统和程序。也可能使用一仿真器程序以在单个 CPU 上并发地操作多个不兼容的操作系统。在这一装置中，尽管每一操作系统与其它操作系统不兼容，但是仿真器程序可主宿两个操作系统之一，从而允许不兼容的操作系统在同一计算机系统上并发地运行。

当在主机计算机系统上仿真访客计算机系统时，访客计算机系统被称为“虚拟机”，因为访客计算机系统仅作为一种特定硬件体系结构的纯软件表示存在于主机计算机系统中。术语仿真器、虚拟机和处理器仿真有时可互换地使用，以表示模仿或仿真一个完整的计算机系统的硬件体系结构的能力。作为一个示例，由加利福尼亚州圣马特奥市的 Connectix 公司创建的 Virtual PC 软件仿真包括 Intel 80X86 Pentium 处理器和各种主板组件和卡的整个计算机。这些组件的操作在运行于主机上的虚拟机中仿真。在诸如具有 PowerPC 处理器的计算机系统等主机计算机的操作系统软件和硬件体系结构上执行的仿真器程序模仿整个访客计算机系统的操作。

仿真器程序担当主机机器的硬件体系结构和由运行在被仿真的环境中的软件发送的指令之间的交换点。该仿真器程序可以是一主机操作系统（HOS），它是直接运行在物理计算机硬件上的操作系统。或者，被仿真的环境也可以是虚拟机监控程序（VMM），它是一软件层，其直接运行在硬件之上，并通过展示与 VMM 正在虚拟化（允许 VMM 不被其上运行的操作系统层注意）的硬件相同的接口来虚拟化该机器的所有资源。主机操作系统和 VMM 可在同一物理硬件上并排运行。·

英特尔(Intel)公司（圣克拉拉(Santa Clara)，加利福尼亚(CA)）的x86体系结构的发展演变始于16位处理器（x86-16），然后扩展到32位处理器（x86-32），而且目前正扩展到64位处理器（x86-64）。64位x86体系结构，它通常称为x86-64体系结构，正由加利福尼亚(CA)桑尼维尔(Sunnyvale)的先进微设备(Advanced Micro Devices)（AMD）公司以及英特尔开发。例如，AMD64位产品在商业上称为AMD64。

但是，这里讨论的64位x86体系结构和另一个由Hewlett-Packard（帕洛阿尔托(Palo Alto)，加利福尼亚(CA)）和英特尔联合开发的称为IA64的64位产品间已有了不同。IA64具有在安腾(Itanium)®处理器中实现的64位指令集体系结构。通常，IA64体系结构没有向后兼容性，因而由于不同的指令集，x86体系结构的软件将不运行在IA64体系结构上。因此，这里讨论的64位体系结构专门指x86-64体系结构而不是IA64体系结构。

提供允许为32位传统OS编写的虚拟机运行于64位OS上的向后兼容性对软件厂商很重要，因为向后兼容性使得新32位产品更短时间推向市场并扩展了传统32位应用程序的使用。x86-64体系结构支持若干不同的操作模式，包括传统32位x86体系结构的操作模式，它如下概述。

传统 32 位 x86 体系结构（x86-32）

- 实模式
- 保护模式：
  - V86 子模式
  - 环-0, -1, -2, -3 子模式

（注：如这里所使用的，术语“模式(mode)”、“子模式(sub-mode)”和“超级模式(super-mode)”用于更好地分辨不同结构的不同模式层；但是，正如本领域那些熟练技术人员所知道和易于理解的，所有这些变化通常都简称为“模式”，而忽略相关结构。）

通常，实模式是同时只允许一个程序的执行的操作模式。在实模式中，程序只访问1024K的内存并使用16位数据路径。保护模式提供对虚拟内存和多任务（同时运行多于一个程序）的支持。保护模式程序可访问超过1024K的地址并使用32位数据路径。实模式是保护模式的先驱，其中每个程序需要所有内存运行并不允许在同一时间另一应用程序的执行。保护模式还包括环-0子模式、环-1子模式、环-2子模式、环-3子模式和V86子模式（虚拟8086）。（通常，当前应用程序不使用环-1和环-2子模式。）

环-0子模式指Intel 80286保护模式结构。环-0子模式是最高优先级，可访问所

有系统资源。环-0子模式是最优先代码，可由OS及其驱动程序使用，它们具有高信任级。环-3子模式也称为Intel 8028保护模式结构。环-3子模式是最低优先级，用于具有低信任级并由所有用户应用程序使用的代码。V86子模式指Intel 80386保护模式体系结构，是其中CPU仿真8086实模式寻址的子模式，但维护对页面调度和某些访问限制的支持。

相反，x86-64体系结构支持两种主要的超级模式，即传统超级模式和长超级模式，如下概述。

扩展的 62 位 x86 体系结构（x86-64）：

- 传统超级模式
  - 实模式
  - 保护模式：
    - V86 子模式
    - 环-0, -1, -2, -3 子模式
- 长超级模式
  - 兼容模式
    - 环-3 子模式
  - 原长模式

x86-64体系结构传统超级模式包括传统32位x86体系结构的所有模式和子模式，即，实模式、保护模式等。此外，x86-84体系结构长超级模式包括原长模式和兼容模式。原长模式允许运行64位环-0代码。兼容模式允许32位环-3应用程序运行在运行64位原长模式的OS顶层，即混合环境。因此，32位应用程序可运行在兼容模式而64位应用程序同时运行在原长模式中。

但是，问题存在于x86-64体系结构的兼容模式只支持环-3子模式。兼容模式不支持V86子模式或环-0子模式，而对于32位虚拟机（VM），需要至少这两者之一（如果不是都需要的话）来完全无缝地仿真/虚拟化32位硬件环境。因此，所需要

的是用于依赖于传统32位模式，即实模式和保护模式（V86子模式、环-0子模式和环-3子模式）的虚拟机在64位处理器上并行运行其它应用程序但仍可执行环-0代码以及访问V86子模式的特性的方法。

一个解决方案是使运行于 x86-64 体系结构的虚拟机环境能够自由地在长超级模式和传统超级模式之间前后转换，以便例如在 64 位处理器上运行 32 位 OS 的访客和运行 64 位 OS 的主机之间前后转换。但是，AMD64 机器的实现，例如，未预先考虑长超级模式与传统超级模式之间前后转换；相反，AMD64 机器的实现假设机器始于传统超级模式，当 OS 加载时进行到长超级模式的单向转换，并再不返回到传统超级模式。所需要的是在 x86-64 体系结构中的长超级模式和传统超级模式之间前后切换的方法。

发明概述

本发明提供了一个系统和方法，用于在x86-64体系结构中原长模式和传统保护模式之间的实时切换，即“处理器模式上下文切换”。为此目的，本发明的各种实施例针对这样的系统和方法，用于通过在x86-64体系结构中长超级模式和传统超级模式之间前后转换的机制在64位x86处理器上运行依赖于传统32位模式，即实模式和保护模式（V86子模式、环-0子模式和环-3子模式）的x86虚拟机。

本发明的某些实施例针对用于在 x86-64 体系结构中长超级模式和传统超级模式之间转换的虚拟化计算系统。在这种情况下，虚拟机可在 x86-64 计算机硬件（即 64 位）上并行运行其它应用程序。执行临时处理器模式上下文切换的方法——“临时”意思是期望，例如，在计算机系统运行从初始化时间到关机时间期间，在长超级模式中执行的系统将转换到传统超级模式，然后最终转换回长超级模式（或某个其它模式），从而至少完成两次模式转换，每次都利用所述方法——包括下列步骤：（a）建立一“虚拟=实”页（“切换页”），（b）将执行处理器模式上下文切换的转换代码放到该页上，（c）跳至该页，（d）禁用 x86-64 计算机硬件的存储器管理单元（MMU），（e）修改模式控制寄存器以设置长超级模式位或传统超级模式位，（f）加载新页表，以及（g）重新激活 x86-64 计算机硬件的 MMU。对于本发明的若干实施例，这些步骤的一或多个由 VMM 执行，而对某些实施例所有七个步骤都由 VMM 执行。

附图说明

前述发明内容以及以下较佳实施例的详细描述，在结合附图阅读时可更好地理解。为了说明本发明的目的，附图中示出了本发明的示例性构造；但是本发明不限于所公开的特定方法和设备。附图中：

图1是一方框图，表示本发明的各方面可具体表现于其中的计算机系统；

图2示出用于计算机系统中仿真操作环境的硬件和软件体系结构的逻辑层次；

图3A示出虚拟化计算系统；

图3B示出一虚拟化计算系统的替换实施例，包括与主机操作系统并行运行的虚拟机监控程序；

图4示出虚拟化计算系统的另一替换实施例，包括与x86-64主机操作系统并行运行的虚拟机监控程序；以及

图5示出一用于在x86-64位处理器上运行传统32位虚拟机的执行临时处理器模式上下文切换操作的方法的流程图。

详细说明

本发明的主题是用细节来描述的，以满足法定要求。然而，该描述本身并不制造限制本专利的范围。相反，发明人构想所要求保护的本发明也可结合其它现有或未来技术用其它方法来实现，以包括不同的步骤或类似于本文档中所揭示的那些步骤的组合。此外，尽管此处使用了术语“步骤”意味着所采用方法的不同元素，然而该术语不应当被揭示为暗示此处所揭示的各个步骤之中或之间的任何特定顺序，除非当明确地描述了个别步骤的顺序。

计算机环境

本发明的主题是用细节来描述的，以满足法定要求。然而，该描述本身并不制造限制本专利的范围。相反，发明人构想所要求保护的本发明也可结合其它现有或未来技术用其它方法来实现，以包括不同的步骤或类似于本文档中所揭示的那些步骤的组合。此外，尽管此处使用了术语“步骤”意味着所采用方法的不同元素，然而该术语不应当被揭示为暗示此处所揭示的各个步骤之中或之间的任何特定顺序，除非当明确地描述了个别步骤的顺序。

虚拟机

从概念上的观点来看，计算机系统一般包括运行在硬件的功能层上的一个或

多个软件层。这一分层是为了抽象的原因而完成的。通过为给定软件层定义接口，该层可由其上的其它层来不同地实现。在设计良好的计算机系统中，每一层仅知道（并且仅依赖于）紧靠其之下的那一层。这允许层或“栈”（多个邻接的层）被替换，而不会对所述层或栈上的层产生消极的影响。例如，软件应用程序（较高层）通常依赖于操作系统的较低层（较低层）来将文件写入某一形式的永久存储中，并且这些应用程序不需要理解将数据写入软盘、硬盘驱动器或网络文件夹之间的区别。如果较低层用用于写文件的新操作系统来替换，则较高层软件应用程序的操作保持不受影响。

分层软件的灵活性允许虚拟机（VM）呈现一虚拟硬件层，它实际上是另一软件层。以此方式，VM 可为其上的软件层创建所述软件层正运行在其自己的专用计算机系统上的假象，并且由此，VM 可允许多个“访客系统”并发地运行在单个“主机系统”上。

图 2 所示是用于计算机系统中仿真的操作环境的硬件和软件体系结构的逻辑分层的图示。仿真程序 94 运行在主机操作系统和/或硬件体系结构 92 上。仿真程序 94 仿真访客硬件体系结构 96 和访客操作系统 98。软件应用程序 100 进而运行在访客操作系统 98 上。在图 2 的被仿真操作环境中，由于仿真程序 94 的操作，即使软件应用程序 100 被设计成运行在一般与主机操作系统和硬件体系结构 92 不兼容的操作系统上，软件应用程序 100 也可运行在计算机系统 90 上。

图 3A 示出了一个虚拟化的计算系统，它包括直接运行在物理计算机硬件 102 上的主机操作系统软件层 104，并且主机操作系统（主机 OS）通过展示与主机 OS 正在虚拟化（令主机 OS 能够不被其上运行的操作系统层注意到）的硬件相同的接口，来虚拟化机器的所有资源。

或者，虚拟机监控程序，或 VMM 软件层 104'可以代替主机操作系统 104"或与其并排地运行，后一选项在图 3B 中示出。为了简明起见，后文所有的讨论（尤其是关于主机操作系统 104 的讨论）应当针对图 3A 所示的实施例；然而，这些讨论的每一方面应当同等地应用于图 3B 的实施，其中图 3B 的 VMM 104'本质上在功能层替换了下文描述的图 3A 的主机操作系统 104 的角色。

再次参考图 3A，在主机 OS 104（或 VMM 104'）上的是两个虚拟机（VM）实现－VM A 108，它可以是，例如虚拟化的 Intel 386 处理器；以及 VM B 110，它可以是，例如 Motorola 680X0 处理器家族之一的虚拟化版本。在每一 VM A 108 和 110 之上的分别是访客操作系统（访客 OS）A 112 和 B 114。在访客 OS A 112

上运行两个应用程序－应用程序 A1 116 和应用程序 A2 118，在访客 OS B 114 上的是应用程序 B1 120。

在x86-64处理器虚拟机上运行虚拟机

图4例示虚拟化计算系统的另一个替换实施例，该虚拟化计算系统包括一个与x86-64主机操作系统并行运行的虚拟机监控程序。图4的虚拟化计算系统在所有方面与图3B的系统相同，除了它包括x86-64主机OS 122，它基于x86-64的OS，运行在包括基于x86的64位处理器（未示出）的x86-64计算机硬件124之上。图4的虚拟化计算系统的VMM 104'、VM A 108、VM B 110、访客OS A 112和访客OS B 114表示与x86-64主机OS 122（用于在虚拟机环境中在x86-64计算机硬件124上执行传统OS和应用）并行运行的32位x86虚拟机。

使用VMM 104'，本发明在x86-64体系结构中的长超级模式和传统超级模式之间提供来回切换的方法。在这么做时，VM A 108，例如，依赖于传统32位模式即实模式和保护模式（V86子模式、环-0子模式和环-3子模式），能够与x86-64计算机硬件124的64位处理器的其它应用并行运行。

通常，“上下文切换(context switch)”描述为VMM对它需要在主机OS中启用服务的识别。因此，VMM临时放弃对主机OS的控制，主机OS执行它的任务，随后主机OS放弃控制回到VMM。现今的32位VM中的上下文切换仅包括交换出状态。然而，为了按照本发明在x86的64位处理器上运行传统32位VM，本发明的系统和方法包括切换上下文切换操作内的处理器模式--即，在长超级模式的原长模式中运行的主机OS切换到传统超级模式的保护模式（不是切换到长超级模式的兼容模式）并且运行访客OS。这个“处理器模式上下文切换”操作是由VMM 104'和x86-64主机OS 122完成的，如参考图5描述的。

图5例示方法130的流程图，它实时执行临时处理器模式上下文切换操作用于在x86的64位处理器上运行传统32位虚拟机。在长超级模式（64位模式）或传统超级模式（32位模式）中操作的x86-64系统中，方法130提供一种方法在两个超级模式之间转换以访问在长超级模式中不单独存在的功能。

通常，方法130的步骤由VMM 104'执行，同时VMM 104'正在32位模式中运行而x86-64主机OS 122在64位模式中运行。（注意：特定步骤是由特定的x86-64处理器规定的，但下面提供一般的步骤）。

在步骤132，VMM 104'建立一个“虚拟=实”页（即，用于页的虚拟寻址与在存

储器中实存储器地址相同的页），随后将用于执行处理器模式上下文切换的转换代码拷贝到这个页，并且随后代码执行“跳”到这个页以执行上下文切换。在步骤134，使用控制寄存器0（CR0），VMM 104'禁用x86-64计算机硬件124的64位处理器的MMU。MMU是负责处理由主处理器请求存储器访问的硬件设备。这一般包括将虚拟地址转换成物理地址、高速缓存控制、总线判优、存储器保护和各种异常的产生。禁用MMU是必需的，因为代码正运行在由MMU定义的地址空间中；因而，如果关闭MMU，在下一个指令上发生什么是不肯定的，因为地址转换已经突然改变。因此，在尝试执行下一个指令时存在危险，即在错误的指令流上操作，除非它保证代码正在逻辑地址与物理地址相同的页上运行。因此，这要求V=R映射（虚拟=实映射），如在步骤132设置的。

在步骤136，VMM 104'修改模式控制寄存器（这可以与处理器体系结构不同），它包含传统超级模式位和长超级模式位。在方法130提供从长超级模式（64位模式）到传统超级模式（32位模式）的转换的情况下，模式控制寄存器的长超级模式位被禁止而传统超级模式位被设置，从而执行从主机上下文到VM上下文的转换。在方法130提供从传统超级模式（32位模式）到长超级模式（64位模式）的转换的情况下，模式控制寄存器的传统超级模式位被禁止而长超级模式位被设置，从而执行从VM上下文到主机上下文的转换。

在步骤138，VMM 104'载入新模式的一个新页表，它是在虚拟存储器系统内的数据结构，通过它在虚拟地址与物理地址之间映射是简单的。（注意：载入新页表以创建“虚拟=实”页（其中用于所述页的虚拟地址是用于所述页的实存储器地址）与要将代码传送到的VMM的虚拟地址之间的映射。新页表也反映新模式的格式--用于两个模式的页表格式是不同的，并且因而是不兼容的。）在步骤140，VMM 104'重新激活x86-64计算机硬件124的64位处理器的MMU并且该方法结束。

如这里使用的，术语“临时(temporary)”仅意味着有时在计算机系统的执行期间发生至少两个模式转换--例如，其中在长超级模式中执行的系统转换到传统超级模式，并且然后，在稍后某个时候，转换回长超级模式（或者完全转换到另一个模式）以完成至少一个模式转换，其每个支路使用所述的方法。而且，术语“临时”不意味着任何时间性质（关于多少时间发生），也不意味着启始模式是结束模式，但再次，只是在执行期间的至少两个模式转换。在这点上，关于一个特定模式的术语“临时模式(temporary mode)”指在既不是在系统启动时的初始模式也不是在系统关闭时的最后模式的操作期间的任何模式。

结论

此处所描述的各种系统、方法和技术可以用硬件或软件，或在适当时用两者的组合来实现。由此，本发明的方法和装置，或其某些方面，可以采用包含在有形介质，如软盘、CD-ROM、硬盘或任何其它机器可读存储介质中的程序代码（即，指令）的形式，其中，当程序代码被加载到诸如计算机等机器上并由其执行时，该机器变为用于实施本发明的装置。在可编程计算机上的程序代码执行的情况下，计算机一般包括处理器、处理器可读的存储介质（包括易失性和非易失性存储器和/或存储元件）、至少一个输入设备以及至少一个输出设备。一个或多个程序较佳地以高级过程语言或面向对象的编程语言来实现，以与计算机系统通信。然而，如有需要，程序可以用汇编语言或机器语言来实现。在任何情况下，语言可以是已编译或已解释的语言，并与硬件实现相组合。

本发明的方法和装置也可以用程序代码的形式来实施，该程序代码通过某一传输介质来发送，如通过电线或电缆、通过光纤或通过任一其它形式的传输，其中，当程序代码由诸如 EPROM、门阵列、可编程逻辑器件（PLD）、客户机计算机、录像机等机器接收被装载到其中由其执行时，该机器变为用于实施本发明的装置。当在通用处理器上实现时，程序代码与处理器相结合，以提供用于调用本发明的功能的唯一装置。

尽管结合各个附图的较佳实施例描述了本发明，然而可以理解，在不脱离本发明的情况下，可以使用其它类似的实施例，或对所描述的实施例作出修改和添加，以执行本发明的相同功能。例如，尽管本发明的示例性实施例是在仿真个人计算机的功能的数字设备的环境中描述的，然而本领域的技术人员可以认识到，本发明不限于这类数字设备，如本申请中所描述的，本发明可应用于任何数量的现有或新兴计算设备或环境，如游戏控制台、手持式计算机、便携式计算机等等，无论它们是有线还是无线的，并且本发明可应用于通过通信网络连接并在网络上交互的任何数量的这类计算设备。此外，应当强调，此处构想了各种计算机平台，包括手持式设备操作系统和其它应用专用硬件/软件接口系统，尤其是当无线联网的设备的数量持续增长时。因此，本发明不应当限于任何单个实施例，而是相反，应当依照所附权利要求书的宽度和范围来解释。

最后，此处所描述的揭示的实施例可适用于其它处理器体系结构、基于计算机的系统、或系统虚拟化，并且这些实施例由此处的揭示明确地考虑在内，因此，

本发明不应限于此处所描述的具体实施例，而是相反，应当被更广泛地解释。同样，为除处理器虚拟化之外的目的而使用合成指令也由此处的揭示考虑在内，并且在除处理器虚拟化之外的环境中对合成指令的任何这样的使用应当被更广泛地包含在此处的揭示的意义之内。

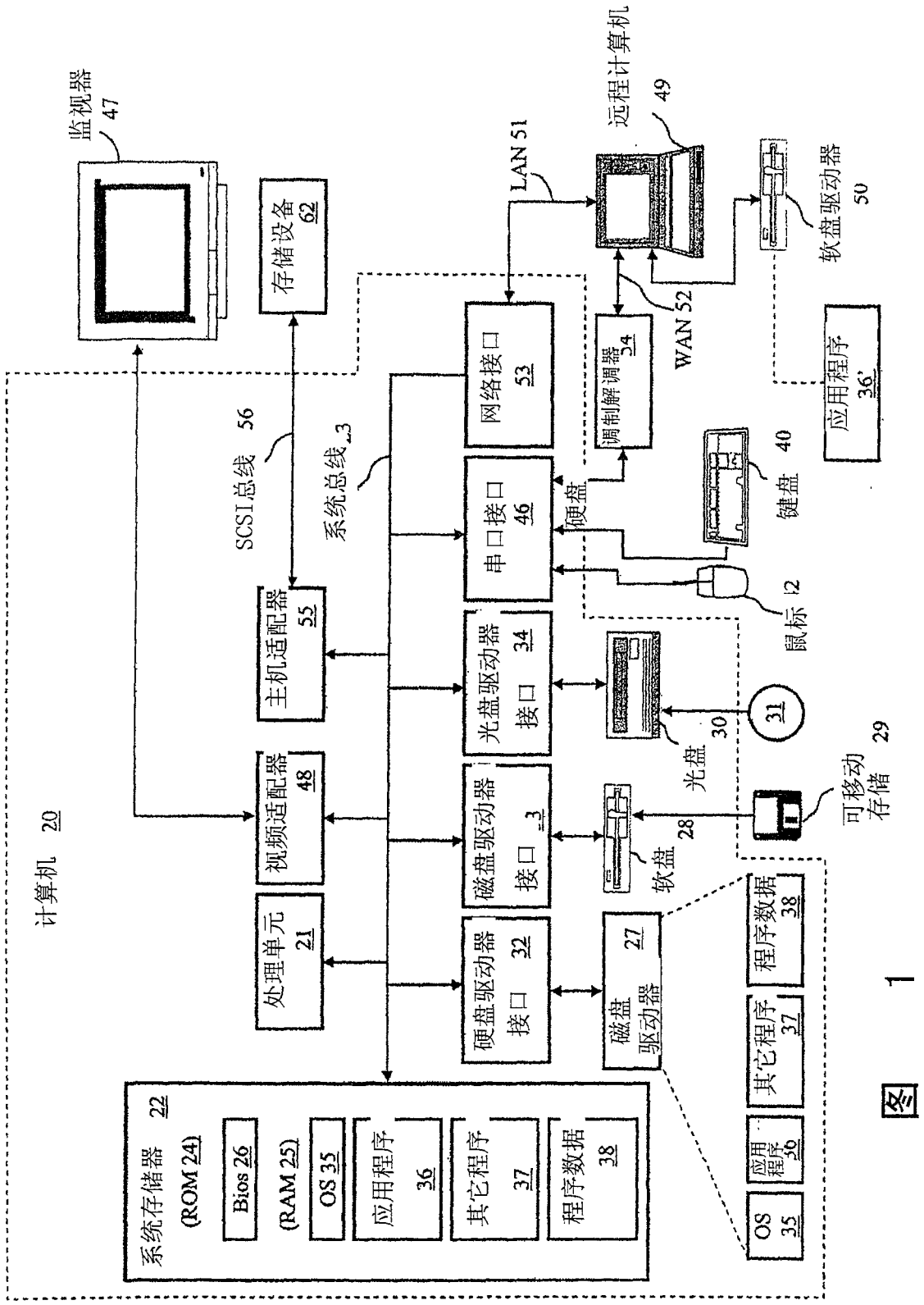

计算机 20
监视器 47
存储设备 62
SCSI总线 56
系统总线 23
LAN 51
远程计算机 49
WAN 52
软盘驱动器 50
应用程序 36'
网络接口 53
调制解调器 54
串口接口 46
硬盘
键盘 40
鼠标 42
主机适配器 55
视频适配器 48
光盘驱动器接口 34
磁盘驱动器接口 33
硬盘驱动器接口 32
处理单元 21
光盘 30
31
软盘 28
可移动存储 29
磁盘驱动器 27
系统存储器 22
(ROM 24)
Bios 26
(RAM 25)
OS 35
应用程序 36
其它程序 37
程序数据 38
OS 35
应用程序 36
其它程序 37
程序数据 38

图 1

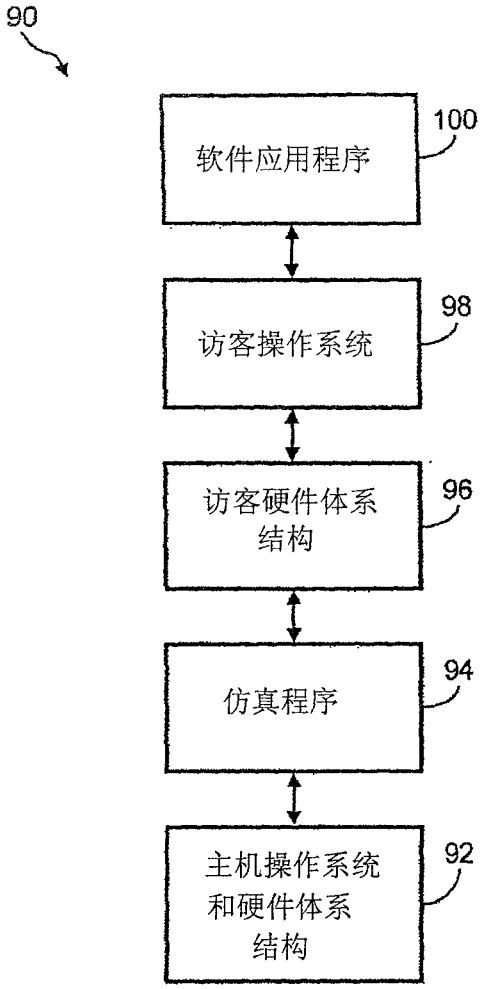

90
软件应用程序
100
访客操作系统
98
访客硬件体系结构
96
仿真程序
94
主机操作系统和硬件体系结构
92

图 2

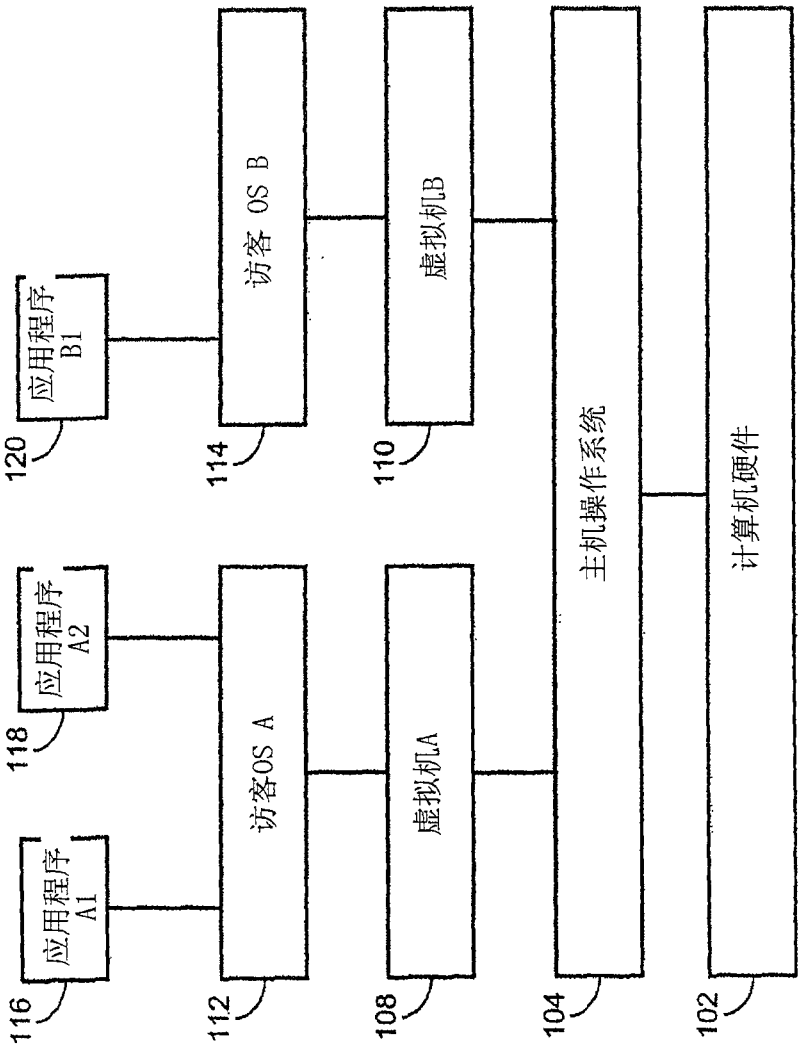

应用程序 A1
应用程序 A2
应用程序 B1
访客OS A
访客 OS B
虚拟机A
虚拟机B
主机操作系统
计算机硬件
116
118
120
112
114
108
110
104
102

图 3A

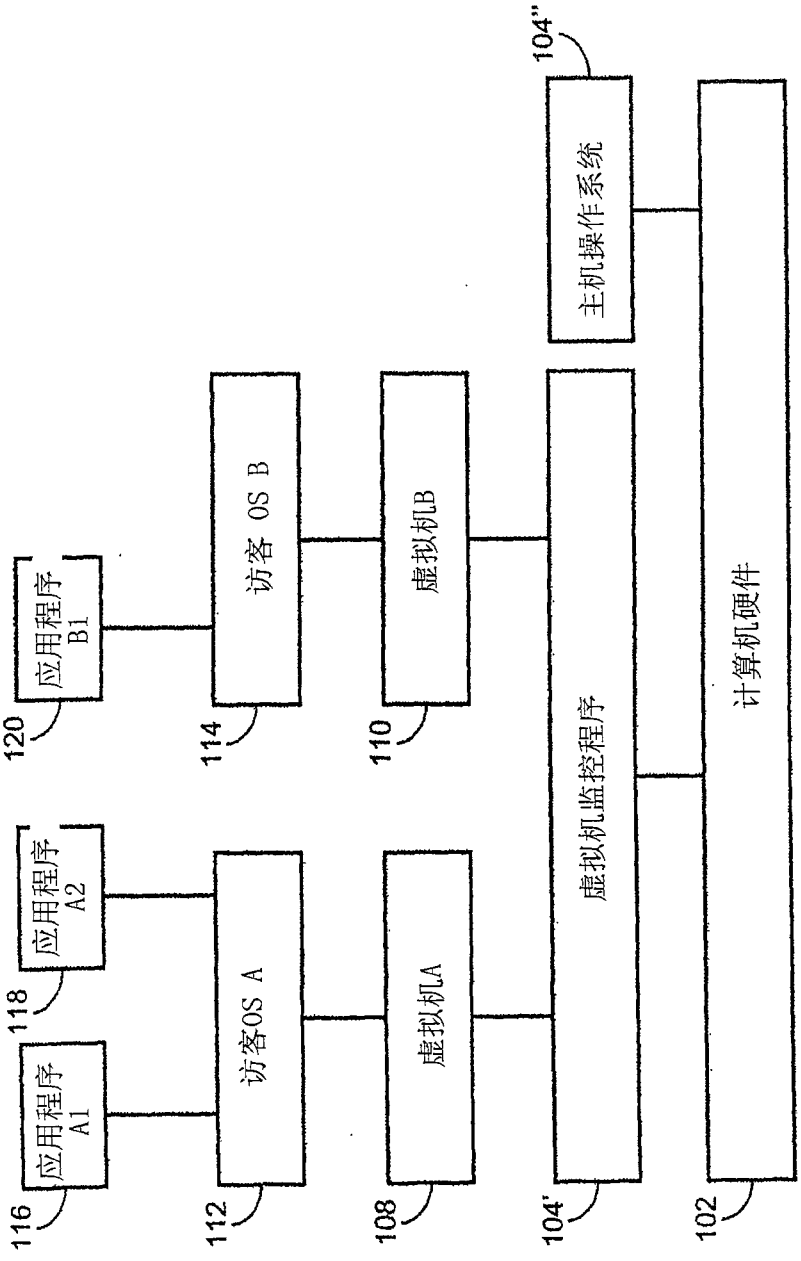

104''
主机操作系统
102
计算机硬件
104'
虚拟机监控程序
108
虚拟机A
110
虚拟机B
112
访客OS A
114
访客 OS B
116
应用程序 A1
118
应用程序 A2
120
应用程序 B1

图 3B

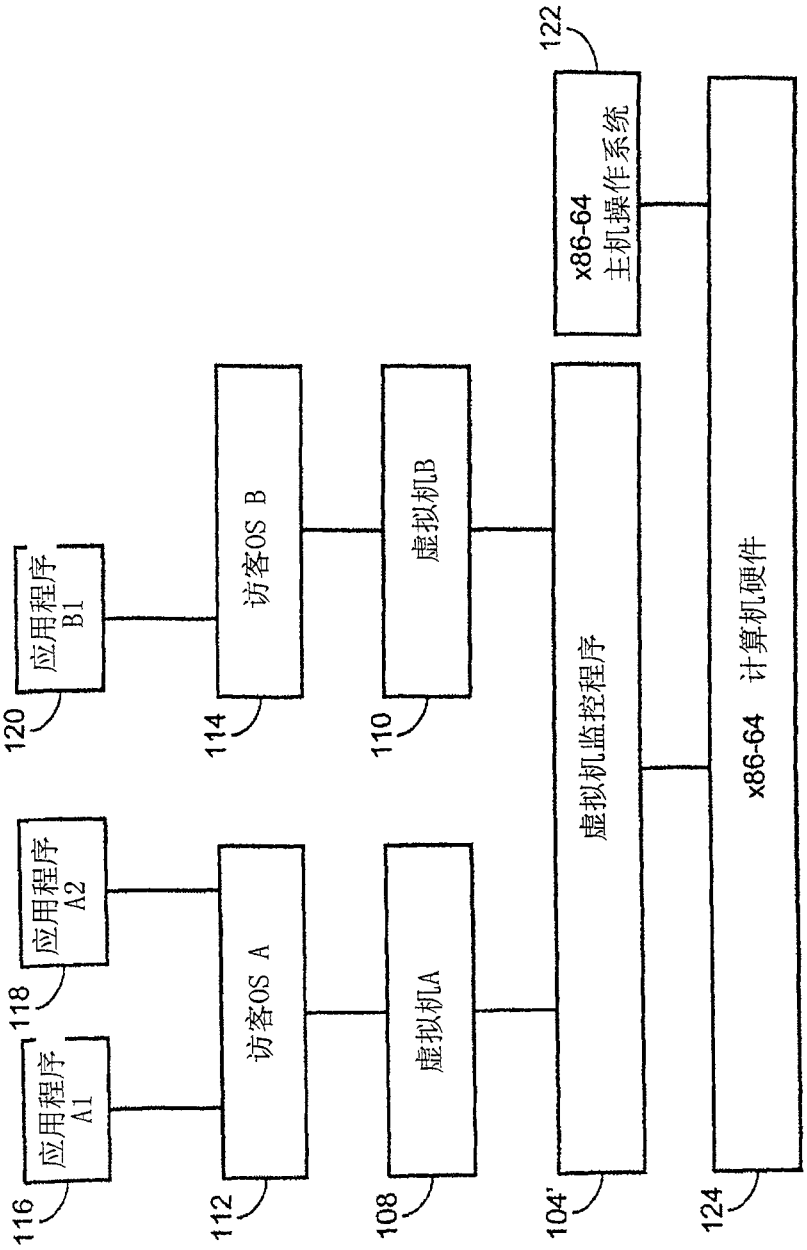

应用程序 A1
应用程序 A2
应用程序 B1
访客OS A
访客OS B
虚拟机A
虚拟机B
虚拟机监控程序
x86-64 计算机硬件
x86-64 主机操作系统
116
118
120
112
114
108
110
104'
124
122

图 4

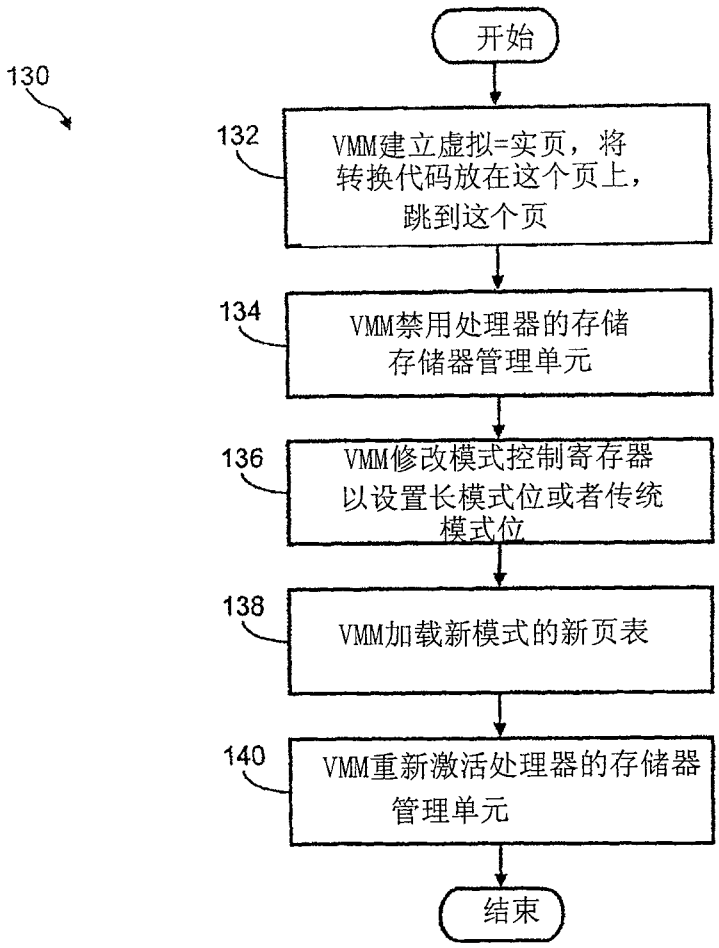

130
开始
132
VMM建立虚拟=实页，将转换代码放在这个页上，跳到这个页
134
VMM禁用处理器的存储存储器管理单元
136
VMM修改模式控制寄存器以设置长模式位或者传统模式位
138
VMM加载新模式的新页表
140
VMM重新激活处理器的存储器管理单元
结束

图 5