



## [12] 发明专利说明书

专利号 ZL 03804546. X

[45] 授权公告日 2008 年 10 月 1 日

[11] 授权公告号 CN 100423013C

[22] 申请日 2003.2.13 [21] 申请号 03804546. X

[30] 优先权

[32] 2002.2.25 [33] US [31] 10/085,839

[86] 国际申请 PCT/US2003/004612 2003.2.13

[87] 国际公布 WO2003/073269 英 2003.9.4

[85] 进入国家阶段日期 2004.8.25

[73] 专利权人 英特尔公司

地址 美国加利福尼亚州

[72] 发明人 詹姆斯·萨顿二世 迈克尔·科祖克  
戴维·克劳罗克

[56] 参考文献

WO00/10283A1 2000.2.24

CN1334521A 2002.2.6

US5953502A 1999.9.14

WO02/03196A2 2002.1.10

EP0849657A1 1998.6.24

审查员 徐卫锋

[74] 专利代理机构 永新专利商标代理有限公司

代理人 王英

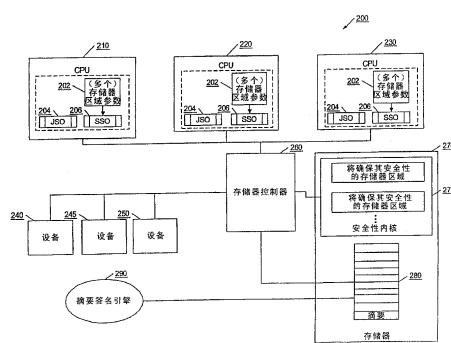
权利要求书 4 页 说明书 10 页 附图 6 页

[54] 发明名称

加载可信操作系统的方法和装置

[57] 摘要

本发明提供了一种方法和装置，可将可信操作系统加载到存储器的一个区域中。在多处理器计算机中，起始安全操作 (SSO) 触发连接安全操作 (JSO)，以停顿除一个中央处理单元 (CPU) 之外的所有其他 CPU。SSO 使得活动的 CPU 将操作系统的组件加载到存储器中的指定区域中，通过在存储器中记录所指定的区域的内容的加密哈希来注册所加载的操作系统的身份，在所指定的区域中的已知入口点处开始运行，并触发 JSO 来使得被停顿的 CPU 同样在所述已知入口点处开始运行。



1. 一种加载可信操作系统的方法，包括：

由多个处理器中的第一处理器执行起始安全操作；

由所述多个处理器中除所述第一处理器之外的其余处理器执行连接安全操作，所述连接安全操作防止所述多个处理器中的所述其余处理器干预所述第一处理器的操作；

所述多个处理器中的所述第一处理器标识出计算机的存储器中的一个区域；

在所述第一处理器的控制下将内容加载到所标识区域中；

注册所标识区域的所述内容的身份，所述注册包括：

记录所标识区域的所述内容的哈希摘要，以及

由具有可访问所述哈希摘要的安全通道的摘要签名引擎来签名所述哈希摘要，该签名的哈希摘要存储在所述计算机的存储器中的寄存器中，其可由外部实体访问以验证所述内容是否可被信任；

使得所述第一处理器跳转到所述内容中的已知入口点；以及

所述第一处理器完成所述起始安全操作，并且发信号通知所述其余处理器恢复活动。

2. 如权利要求 1 所述的方法，还包括：

当所述第一处理器正在将所述内容加载到所标识区域中时，防止所述多个处理器中的至少第二处理器干预所述标识、加载和注册。

3. 如权利要求 2 所述的方法，其中防止干预包括停顿所述多个处理器中的至少所述第二处理器，直到所述标识、加载和注册完成。

4. 如权利要求 2 所述的方法，还包括：

使得所述多个处理器中的至少所述第二处理器跳转到所述内容中的已知入口点。

5. 如权利要求 1 所述的方法，其中“标识”包括接收一个区域参数，该区域参数指定了所述区域的位置。

6. 如权利要求 5 所述的方法，其中所述位置是所述计算机的存储器

中可定位所述区域的地址范围。

7. 如权利要求 5 所述的方法，其中所述位置包括所述计算机的存储器中可定位所述区域的起始地址和长度。

8. 如权利要求 1 所述的方法，其中所述内容是将操作所述计算机的操作系统的组件。

9. 如权利要求 8 所述的方法，其中所述操作系统是 Windows 操作系统、Windows 95 操作系统、Windows 98 操作系统、Windows NT 操作系统、Windows 2000 操作系统、虚拟机监控器以及特权软件核中之一。

10. 如权利要求 1 所述的方法，其中所述标识、加载和注册不可中断。

11. 一种确保计算机的存储器中某个区域的安全性的方法，包括：

在计算机中的多个 CPU 中，停顿除一个 CPU 之外的所有其他 CPU；

阻止除了所述未被停顿的 CPU 之外的所有其他资源访问该计算机的存储器中的某个区域；

记录所述区域的加密哈希；以及

将所述未被停顿的 CPU 置入已知的特权状态。

12. 如权利要求 11 所述的方法，还包括使得所述未被停顿的 CPU 跳转到所述区域中的已知入口点。

13. 如权利要求 11 所述的方法，其中“停顿”包括使得多个 CPU 中除一个 CPU 之外的所有其他 CPU 进入专用停顿状态。

14. 如权利要求 13 所述的方法，还包括：使得所述被停顿的 CPU 在所述未被停顿的 CPU 已被置入所述已知特权状态后退出所述专用停顿状态。

15. 如权利要求 14 所述的方法，还包括：使得以前被停顿的 CPU 在退出所述专用停顿状态后在所述区域中的已知入口点处开始运行。

16. 如权利要求 11 所述的方法，其中记录所述加密哈希包括：

清除所述计算机的存储器中的哈希摘要区域；

将所需的平台信息记录在所述哈希摘要区域中；

计算所述区域的内容的加密哈希；以及

将所计算的加密哈希记录在所述哈希摘要区域中。

17. 如权利要求 16 所述的方法，其中所述哈希摘要区域是所述计算机的存储器中的寄存器。

18. 如权利要求 16 所述的方法，其中计算所述区域的内容的加密哈希由耦合到所述计算机的存储器的摘要签名引擎执行。

19. 如权利要求 11 所述的方法，其中在至少一个区域参数中指定所述区域。

20. 如权利要求 19 所述的方法，其中所述至少一个区域参数是所述计算机的存储器中其安全性将受确保的区域的地址。

21. 如权利要求 19 所述的方法，其中所述至少一个区域参数是所述计算机的存储器中其安全性将受确保的区域的长度。

22. 一种加载可信操作系统的装置，包括：

具有起始安全操作的第一处理器，所述起始安全操作具有存储器区域参数，其中所述第一处理器执行所述起始安全操作，以阻止对所述存储器区域参数中指定的存储器区域的访问，并将内容置入所指定的区域中；

摘要签名引擎，具有可访问哈希摘要的安全通道，该摘要签名引擎签名所述哈希摘要，并将该签名的哈希摘要存储在所述计算机的存储器中的寄存器中，其可由物理地独立于所述装置的外部实体访问以验证所述内容是否可被信任；以及

其中，所述第一处理器还执行所述起始安全操作，以便 (i) 清除所述哈希摘要的当前内容，并在所述哈希摘要中记录所指定的区域的内容的加密哈希，以及 (ii) 解除对所指定的区域的访问禁止，并跳转到所指定的区域的内容中的已知入口点。

23. 如权利要求 22 所述的装置，还包括：

第二处理器，所述第二处理器具有连接安全操作，其中所述第二处理器执行所述连接安全操作来防止所述第二处理器干预所述第一处理器对所述起始安全操作的执行。

24. 如权利要求 23 所述的装置，其中当所述第一处理器开始执行所述起始安全操作时，所述第二处理器开始执行所述连接安全操作。

25. 如权利要求 23 所述的装置，其中，为了防止所述第二处理器干预所述第一处理器对起始安全操作的执行，所述连接安全操作使得所述第二处理器进入停顿状态，直到所述第一处理器完成所述起始安全操作的执行。

26. 如权利要求 25 所述的装置，其中所述第一处理器执行连接安全操作来进一步使得所述第二处理器退出所述停顿状态，所述退出发生在所述第一处理器完成所述起始安全操作的执行并在所指定的区域的内容中的已知入口点处开始运行后。

27. 如权利要求 22 所述的装置，还包括具有可访问所述哈希摘要的安全通道的摘要签名引擎，该摘要签名引擎响应于执行所述起始安全操作的所述第一处理器的请求，计算所指定的区域中的内容的加密哈希。

28. 如权利要求 22 所述的装置，其中所述哈希摘要是所述装置的存储器中所指定的区域之外的寄存器。

## 加载可信操作系统的方法和装置

### 技术领域

本发明涉及微处理器。具体地说，本发明涉及处理器安全性。

### 背景技术

微处理器和通信技术的进展已经为超越传统经商方法的应用提供了许多机会。电子商务和“企业到企业”交易正变得非常普及，以很快的速度达到了全球市场。不幸的是，尽管现代微处理器系统向用户提供了方便而有效的经商、通信和交易方法，但是它们也容易受到肆无忌惮的攻击。这些攻击的例子包括病毒、入侵、安全性破坏以及篡改等等。因此，计算机安全性对于保护计算机系统的完整性并增加用户的信任来说都变得越来越重要。

在操作系统的上下文中，计算机安全性最初是由确认“你正在加载（或已经加载了）一个可信操作系统”来确定的。在可信操作系统中，用户或第三方随后可检查该系统，并确定是否已加载了给定的操作系统，如果是的话，则确定该系统是否已加载到安全的环境中。

然而，当引导普通操作系统时，必须引导多种代码组件。即使你可以选择应该加载哪个代码组件，操作系统仍包含了极大量的代码，以致难于确认该操作系统的具体身份，也难于确认你是否应信任它，即它是否已被加载到安全的环境中。

在多处理器环境中，确定是否可信任操作系统尤其困难。这是因为每个中央处理单元（CPU）（有时甚至是一个系统设备）都可以执行可能会改变并损害已加载的代码的完整性的代码流。因此，经常需要至少在操作系统级别来假设该操作系统是可信的。这一假设可能是错误的，并可导致计算机安全中的灾难性故障。

## 附图说明

在附图中以示例性实施例而非限制性地描述了本发明，类似的标号表示类似的元件，其中：

图 1 示出了典型的操作系统组件及对应的特权级的组织的一般性概要；

图 2 的框图示出了包含本发明的计算机系统的一个通用实施例，其中可实施本发明的某些方面；

图 3 的流程图示出了一种方法的某些方面，该方法将由执行图 2 所示的本发明一个实施例的计算设备所执行；

图 4 的流程图示出了一种方法的某些其他方面，该方法将由执行图 2 所示的本发明一个实施例的计算设备所执行；

图 5 的流程图示出了一种方法的某些方面，该方法将由执行图 2 所示的本发明另一个实施例的计算设备所执行；并且

图 6 的框图示出了一种计算机系统的一个通用实施例，在其中可实施本发明在图 2—5 中所示的某些方面。

## 具体实施方式

在下面对本发明若干方面的描述中，将描述用于加载可信操作系统的办法和装置。将给出具体细节以透彻地理解本发明。然而，对本领域内的技术人员来说，很清楚利用（或没有）所有这些具体细节（或其中一些），并且只是利用本发明在此描述的一些或所有方面，也可以实施本发明。在一些情形下，省略或简化了一些公知特征，以免混淆本发明。

本说明书的一些部分所使用的术语是本领域内的技术人员用来向其同行描述其工作的本质时采用的术语，包括计算机系统所执行的操作及其操作数的术语，例如发送、接收、检索、确定、生成、记录、存储等等。本领域内的技术人员将会理解到，这些操作数采用了电、磁或光信号的形式，所述操作涉及通过系统的电、磁或光组件来存储、传输、结合以及操作这些信号。所述系统包括这些组件的通用及专用设置，所述组件可以是单独的、附属的或嵌入式的。

若干操作将被描述为依次执行的多个分离步骤，该描述方式对理解本发明来说是很有帮助的。然而，所述描述顺序不应被理解为意味着这些操作必须以提供它们的顺序来执行，也不一定是顺序相关的。最后，重复使用的短语“在一个实施例中”未必指的是相同的实施例，尽管可能是相同的实施例。

在计算机系统或平台中提供安全性的一个原理是“实施特权级”这一概念。特权级限制了特定软件组件可以访问哪些系统资源（例如特权指令、存储器、输入/输出设备等等）。图 1 示出了典型的操作系统组件及对应的特权级的组织的一般性概要。在没有虚拟机（VM）技术的系统 100 中，操作系统 120 包括一个称为特权软件核 125 的小驻留程序组件，其以最高特权级 170 运行，即特权软件核 125 可以执行特权和非特权指令，并可访问存储器和 I/O 设备。另一类系统组件，即设备驱动程序 130 也可以以高特权级 170 运行，尤其是在系统支持直接存储器访问（DMA）事务的情况下，在 DMA 事务中，设备驱动程序 130 可以将其设备的内容直接写到存储器而不涉及处理器（例如不使用用于访问存储器的特权软件核 125）。其他类型的系统组件例如应用程序 140 以较低的特权级 180 运行，只能执行非特权或较低特权指令，或者可在操作系统 120 中对特权软件核 125 作出管理性调用（SVC）以执行特权指令，或者更一般地说是代表应用程序 140 访问特权系统资源。

在具有 VM 技术的系统 110 中，另一类系统组件以最高特权运行：虚拟机监控器（VMM）150。在 VM 系统 110 中，操作系统 120 实际运行的特权低于 VMM 150。在一些 VMM 实现中，VMM 150 可被分割成 VMM 核心组件 150 和一个或多个 VMM 扩展 160，VMM 扩展 160 的运行特权低于 VMM 核心组件 150 但高于操作系统 120。按照这种方式，VMM 核心组件 150 在出现问题的 VMM 扩展 160 时也可保持其完整性。

图 2 的框图示出了包含本发明的计算机系统 200 的一个通用实施例，其中可实施本发明的某些方面。应理解到，计算机系统 200 的各个组件之间的差别仅仅是逻辑上的差别；在实际中，这些组件中的任一个都可被集成到同一硅管芯（die）上、被划分成多个管芯或以上二者的组合，而不偏

---

离本发明的范围。在所示出的计算机系统 200 中，中央处理单元 (CPU) 210/220/230 或设备 240/245/250 具有使它们可启动存储器 270 中的事务所需的高特权级 170。存储器控制器 260 负责将来自存储器 270 的存储器事务转发到适当的目的地。

计算机系统 200 还包括加密哈希值的哈希摘要 280，所述哈希值标识出已被加载到存储器 270 的多个区域中的一个或多个操作系统组件的内容。应注意，加密哈希值在本领域中公知为由一个单向数学或其他函数生成，该函数具有一个称为预映射 (pre-image) 的可变长度输入串，并将之转换成固定长度的输出串，该输出串称为哈希值，一般较小。哈希函数是单向的，这是因为难于生成与另一个预映射的哈希值相匹配的预映射。哈希摘要签名引擎 290 具有可访问哈希摘要 280 的安全通道，并在接收到请求时对哈希摘要 280 的内容进行签名。对哈希摘要 280 的内容进行签名在本领域中是公知的，并被用来产生数字签名，该签名以后可用来认证签名者的身份，并确保哈希摘要 280 的内容未被篡改。通过请求这一签名，外部实体可观测系统组件由所述哈希报告的状态，并决定是否信任计算机系统 200，即哈希摘要 280 的签名内容是否与所述系统组件的预期签名相匹配。

为了确保由所述哈希报告的所述组件状态使得可信任计算机系统 200，该计算机系统的各个 CPU 210/220/230 都包含或都能够包含本发明的方法和装置的实施例，以促进可信操作系统的安装（或加载）。

在一个实施例中，本发明的方法和装置包括起始安全操作 (start secure operation, SSO) 206 和连接安全操作 (join secure operation, JSO) 204，二者都能够运行在该计算机系统的任一 CPU 210/220/230 上。SSO 206 和 JSO 204 是原子性地执行、以确保计算机系统 200 的完整性的逻辑操作。SSO 206 和 JSO 204 可被实现为在软件、硬件或其结合中执行的一系列特权指令，而不偏离本发明的范围。

在一个实施例中，SSO 206 取得存储器 270 在存储器区域参数 202 中已指定的一个区域（或多个区域），并使得计算机系统 200 执行多个操作，所述操作使得 CPU 210/220/230 中的一个将操作系统代码中的一个或

多个组件加载并注册到存储器 270 的指定区域中，而 JSO 204 防止其他 CPU 进行干预。加载所述一个或多个操作系统组件后，JSO 204 和 SSO 206 还迫使 CPU 210/220/230 跳转到存储器 270 中现在已确保其安全性的指定区域中的已知入口点（也称为安全性内核 275），该入口点处于已知的特权状态中，即根据所述 CPU 的对应的高特权级 170，可访问计算机系统 200 的资源的状态。

在一个实施例中，在通过存储器区域参数 202 或以其他方式标识出存储器 270 中将确保其安全性的区域或多个区域后，SSO 206 将要确保其安全性的代码置入存储器 270 中所标识出的区域中，即将所述操作系统代码（或其一部分）置入安全性内核 275 中。所述代码可以是任何希望被信任的代码，例如操作系统 120 的特权软件核 125，或者在具有 VM 的系统 110 中，所述代码是 VMM 核心 150，即 VM 监控器核心代码。

在一个实施例中，将所述代码置入安全性内核 275 之后，SSO 206 通过注册所述操作系统代码（例如特权软件核 125 或 VMM 核心 150）的身份来安全地启动该操作系统。SSO 206 通过计算并记录所述代码的哈希摘要 280，并使用哈希摘要签名引擎 290 来对哈希摘要 280 进行密码签名，从而注册所述代码的身份。注册后，所述操作系统成为可信操作系统，能够被外部实体所验证。

在具有多个 CPU 的计算机系统 200 中，如图 2 所示，计算机系统 200 必须还能够防止执行 SSO 206 的 CPU 210 之外的 CPU 220/230 干预可信操作系统的安全启动。因此，每个 CPU 210/220/230 都还设置了 JSO 204。当在 CPU 210 上启动 SSO 206 时，SSO 206 通知其他 CPU 220/230 执行 JSO 204。

在一个实施例中，JSO 204 迫使各 CPU 220/230 进入一个专用停顿状态，并将它们进入停顿状态这一事件发信号通知给启动 SSO 的 CPU 210。当启动 SSO 的 CPU 210 接收到来自所有其他 CPU 220/230 的停顿信号时，SSO 206 通过将所期望的代码置入安全性内核 275 并注册它，从而开始加载可信操作系统。启动 SSO 206 的 CPU 210 完成可信操作系统的加载后，即当已在安全性内核 275 中注册所述代码的身份时，SSO 206 迫使 CPU

210 跳转到安全性内核 275 中的已知入口点，由于 SSO 206 的操作，安全性内核 275 此时已具有已知的特权状态。另外，SSO 206 发信号通知其他 CPU 220/230 退出它们各自的专用停顿状态。在退出停顿状态后，JSO 204 迫使 CPU 220/230 也跳转到安全性内核 275 中的已知入口点。

在一个实施例中，将存储器区域参数 202 指定为存储器 270 中的一个地址范围，该参数包括一个或多个起始地址和停止地址对。然而，也可采用其他方式来指定将确保存储器 270 中的哪个或哪些区域的安全性，而不偏离本发明的范围。例如，存储器区域参数 202 的另一个实施例也可被指定为起始地址和区域长度。

现在参考图 3—5，参考一系列流程图，以计算机软件的方式来描述本发明的特定方法。将由计算机执行的所述方法构成了由计算机可执行指令组成的计算机程序。通过参考流程图来描述所述方法，这使得本领域内的技术人员可开发包括这些指令的程序，以在适当配置的计算机上执行所述方法（所述计算机的处理器执行来自计算机可访问介质的指令）。所述计算机可执行指令可以计算机编程语言来编写，或可以包含在固件逻辑或微引擎代码中等等。如果以遵从公认标准的编程语言编写，则这些指令可以在多种硬件平台上运行，并可接口到多种操作系统。另外，本发明并非是参考任何特定的编程语言来描述的。将可认识到，可使用多种编程语言来实现本发明在此的教导。而且，本领域中将具有多种形式（例如程序、过程、进程、应用程序等等）的软件视作为采取一个动作或造成某种结果是很常见的。这些表述仅仅是“计算机对软件的执行使得计算机的处理器执行一个动作或产生某种结果”的便捷表达方式。

图 3 的流程图示出了一种方法的某些方面，该方法将由执行图 2 所示的本发明一个实施例的计算设备所执行。具体地说，图 3 示出了将由执行 SSO 206 的计算机执行的一些动作，其中 SSO 206 包含了本发明的一个实施例。处理开始于过程 305，在此，计算机系统 200 的 CPU 之一例如 CPU 210 通过在过程 310 确保计算机系统 200 的所有其他 CPU 220/230 已执行 204，从而准备执行 SSO 206。JSO 204 使得计算机系统 200 的其他 CPU 220/230 进入停顿状态，因此它们在可信操作系统的加载期间不能干预

SSO 206 和 CPU 210。在一个实施例中，在已停顿所有其他 CPU 220/230 之后，SSO 206 在过程 315 处继续，使得 CPU 210（或者其他情形下是存储器控制器 260）阻止计算机系统 200 的设备 240/245/250 访问存储器 270 在存储器区域参数 202 中指定的区域（即安全性内核 275）。在 SSO 206 的持续期间阻止设备访问安全性内核 275 一般只是在支持直接存储器访问（DMA）的计算机系统 200 中才是必需的。在一个实施例中，阻止设备访问安全性内核 275 也可由标准芯片集来执行。

在一个实施例中，在过程 320，SSO 206 清除哈希摘要 280 的当前内容，以准备记录当前的平台和哈希摘要信息。在过程 325，SSO 206 将所述平台信息记录在哈希摘要 280 中。平台信息的记录可以是也可不是必需的，这取决于计算机系统 200 的体系结构，并可包括执行 SSO 206 的 CPU 210 的版本号等等。在过程 330，SSO 206 还计算当前出现在安全性内核 275 中的代码（即特权软件核 125 或 VMM 核心 150）的加密哈希摘要。SSO 206 还将这一信息记录在哈希摘要 280 中。在过程 335，在哈希摘要 280 中记录必要的信息后，SSO 206 将 CPU 210 置入到已知的特权状态中。一旦 CPU 210 处于已知的特权状态，SSO 206 就可以进一步迫使 CPU 210 跳转到安全性内核 275 中的已知入口点。所述已知的入口点可以是安全性内核 275 的任意可寻址区域。CPU 210 跳转到已知入口点后，SSO 206 完成，发信号通知其他 CPU 220/230 恢复活动并将控制权返回给 CPU 210。

SSO 206 完成后，外部实体可向哈希摘要签名引擎 290 发送请求，以激活可访问哈希摘要 280 的安全通道，并使得哈希摘要签名引擎 290 读取 SSO 206 所记录的摘要 280 的内容并对之进行密码签名。如前所述，通过请求这一签名，外部实体可观测由所述哈希报告的组件状态，并决定是否信任计算机系统 200，即是否已加载可信操作系统。

图 4 的流程图示出了一种方法的某些方面，该方法将由执行图 2 所示的本发明一个实施例的计算设备所执行。具体地说，图 4 示出了运行 JSO 204 的计算机执行的一些动作，其中 JSO 204 包含了本发明的一个实施例。处理开始于过程 405，在此，计算机系统 200 的每一个非 SSO CPU 例

---

如 CPU 220/230 响应于 CPU 210 上的 SSO 206 的动作而进入专用停顿状态。所述停顿状态防止 CPU 220/230 在可信操作系统的加载期间干预 SSO 206 和 CPU 210。CPU 220/230 中的每一个都在进入停顿状态时发信号通知 CPU 210 上的 SSO 206。JSO 204 继续过程 415 处的判定，一直等到接收到“CPU 210 上的 SSO 206 已完成可信操作系统的初始化”这一信号为止。所述初始化一完成，JSO 204 就继续过程 420，使得 CPU 220/230 退出所述专用停顿状态。在过程 425，JSO 204 使得 CPU 220/230 跳转到安全性内核 275 中的已知入口点，之后，JSO 204 在终点 430 处完成处理，并将控制权返回给各 CPU 220/230。

图 3-4 描述了 SSO 206 和 JSO 204 的处理的通用实施例，图 5 描述了 SSO 206 和 JSO 204 在具有 VM 的计算机系统 200（包括具有 32 位 CPU 的 VM 系统和 VMM 扩展 160）上的示例性实现。处理开始于过程 505，在此，计算机系统 200 的 CPU 之一（例如 CPU 210）上的 SSO 206 接收存储器区域参数 202，该参数具有表示为参数 EAX 的起始物理地址和表示为 ECX 的结束物理地址的形式。EAX 和 ECX 中指定的地址一起指定了存储器 270 中将确保其安全性的区域。SSO 206 在过程 510 采取准备性动作，以提供 SSO 206 将在其中运行的所需环境。所述准备性动作取决于计算机系统 200 的体系结构，并可包括（但不局限于）确保所述起始物理地址 EAX 具有小于结束物理地址 ECX 的值。另外，SSO 206 可确保启用了 CPU 210 的保护模式并禁止了调页（paging）、物理地址扩展和 VM 扩展模式，并将 CPU 210 的特权级暂时设置为零。其他可能的准备性动作可包括禁止对存储器 270 中将确保其安全性的区域或多个区域（即安全性内核 275）的直接存储器访问（DMA），以及禁止到 CPU 210 的硬件中断。禁止硬件中断有助于确保原子性地执行 SSO 206 和 JSO 204。更重要的是，SSO 206 使得其他 CPU 220/230 中的每一个启动 JSO 204 以确保所有其他非 SSO CPU 都停顿，从而防止其干预 SSO 206 的操作，这样，SSO 206 就提供加载可信操作系统所需的环境。

完成所述准备性动作后，SSO 206 在过程 515 处继续，为存储器 270 中的指定区域创建加密哈希 280，所述区域开始于地址 EAX，结束于地址

ECX。当确保存储器 270 中的多个区域的安全性时，重复过程 515，直到其安全性受确保的所有区域即整个安全性内核 275 都被包含在加密哈希 280 中。在过程 520，SSO 206 将加密哈希 280 记录在作为哈希摘要 280 的芯片集寄存器中。SSO 206 继续过程 525，引导 CPU 210 进入已知状态，并进一步继续过程 530，使得 CPU 210 跳转到存储器 270 中的哈希（即其安全性受确保的）区域，即安全性内核 275。SSO 206 结束在过程 535，在此 CPU 210 将处于所引导的已知状态，所有的中断被禁止，并且将确保安全性内核 275 的安全性。

图 6 示出了一种通用计算机系统 600 的一个实施例，在其中可实施本发明在图 2—5 中所示的一个实施例。本发明的一个实施例可实现在个人计算机（PC）体系结构上。然而，对本领域内的普通技术人员来说，很清楚也可采用其他计算机系统体系结构或其他处理器、可编程或电子设备。

一般地，图 6 所示的计算机系统包括一个或多个处理器 602，其通过总线 601 耦合到随机访问存储器（RAM）603、只读存储器（ROM）604 和大容量存储设备 607。大容量存储设备 607 表示持久性数据存储设备，例如软盘驱动器、（例如磁、光、磁光等等的）固定盘驱动器，或者流式磁带驱动器。处理器 602 表示任意体系结构类型的中央处理单元，例如复杂指令集计算机（CISC）、精简指令集计算机（RISC）、超长指令字（VLIW）或混杂式体系结构。在一个实施例中，处理器 602 与英特尔体系结构（IA）处理器（例如 Pentium<sup>TM</sup> 系列、IA-32<sup>TM</sup> 和 IA-64<sup>TM</sup>）兼容。在一个实施例中，计算机系统 600 包括任意数量的处理器，例如图 2 所示的 CPU 210/220/230。

显示设备 605 通过总线 601 耦合到（多个）处理器 602，并提供计算机系统 600 的图形输出。输入设备 606 例如键盘或鼠标耦合到总线 601，以传送信息和命令选择到处理器 602。输入/输出接口 610 也通过总线 601 耦合到处理器 602，该接口可用于控制连接到计算机系统 600 的电子设备（打印机、其他计算机等等）并向其传输数据。计算机系统 600 包括网络设备 608，用于将计算系统 600 连接到网络 614，可通过该网络从例如远程设备 612 接收数据。网络设备 608 可包括以太网设备、电话插孔和卫星

链路。对本领域内的普通技术人员来说很清楚也可利用其他网络设备。

本发明的一个实施例可全部存储为大容量存储设备 607 上的软件产品。本发明的另一个实施例可嵌入在硬件产品上，例如印刷电路板中、专用处理器中或可通信地耦合到总线 601 的专用编程逻辑设备中。本发明的其他实施例可部分实现为软件产品而部分实现为硬件产品。

当本发明的实施例被表示为存储在机器可访问介质（也称为计算机可访问介质或处理器可访问介质）例如大容量存储设备 607 上的软件产品时，该机器可访问介质可以是任何类型的磁、光或电存储设备，包括磁盘、CD-ROM、存储器设备（易失性或非易失性的）或类似的存储机构。所述机器可访问介质可包含指令、代码序列、配置信息或其他数据的各种集合。本领域内的技术人员将会认识到，实现在此描述的本发明所需的其他指令和操作也可存储在所述机器可访问介质上。在本发明的一个实施例中，所述机器可访问介质包括一些指令，所述指令被机器执行时可使得该机器执行包含 SSO 206 和 JSO 204 在内的操作。

因此，在此描述了一种用于加载可信操作系统的新方法。从前面的描述中，本领域内的技术人员将会认识到可对本发明作出许多变动。例如，当在大型机或可比较的机器类型上实现本发明时，可以不必禁止对存储器 270 中将确保其安全性的一个区域或多个区域（即安全性内核 275）的直接存储器访问（DMA），或者不必禁止到 CPU 210 的硬件中断。另一方面，当在具有 PC 体系结构的机器上实现本发明时，可能需要这些附加的保护机制来提供可在其中实施本发明的操作环境。因此，本发明并不受限于所描述的细节。相反，在所附权利要求的精神和范围之内，可以对本发明进行修正和改动来实施本发明。

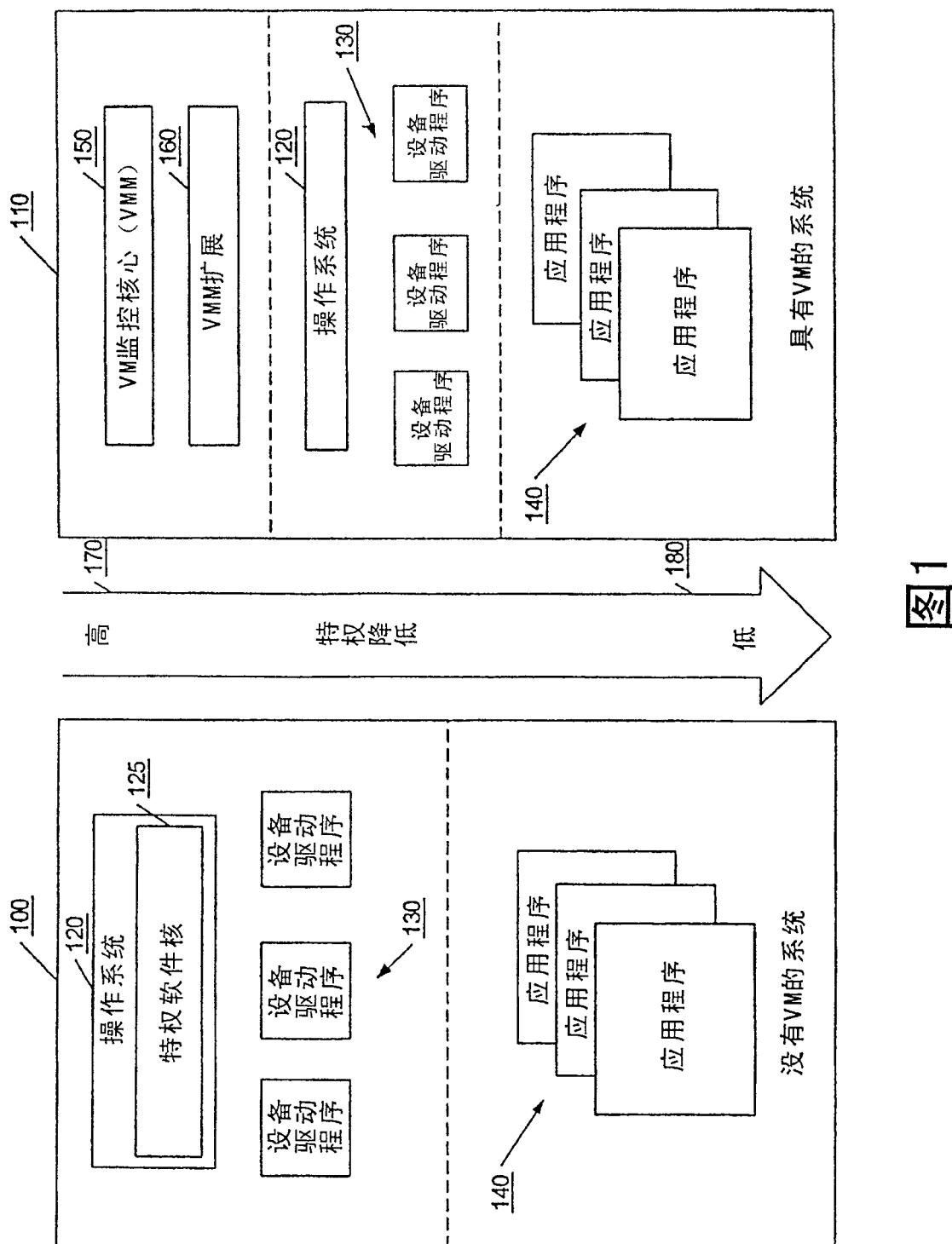


图 1

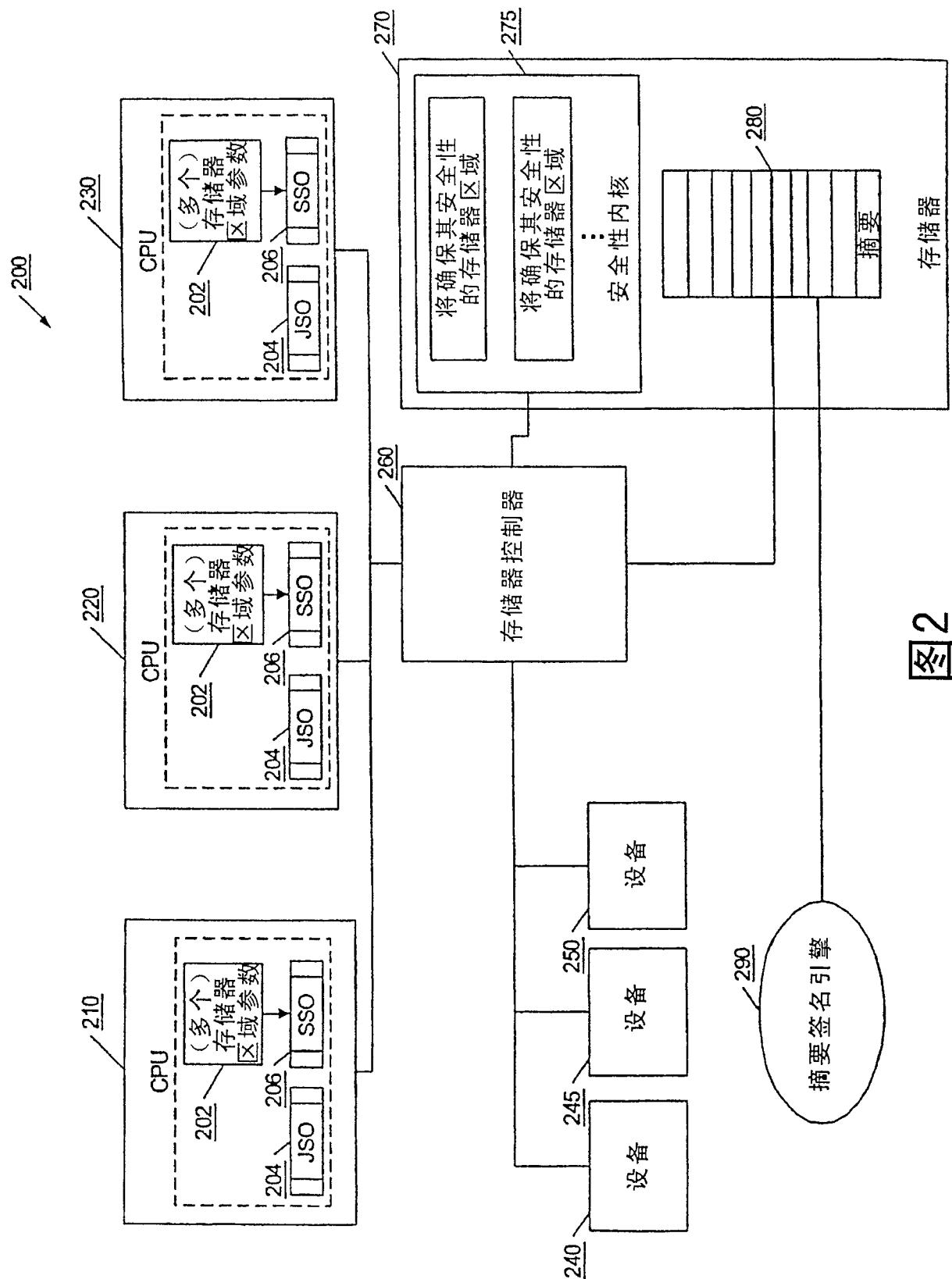


图2

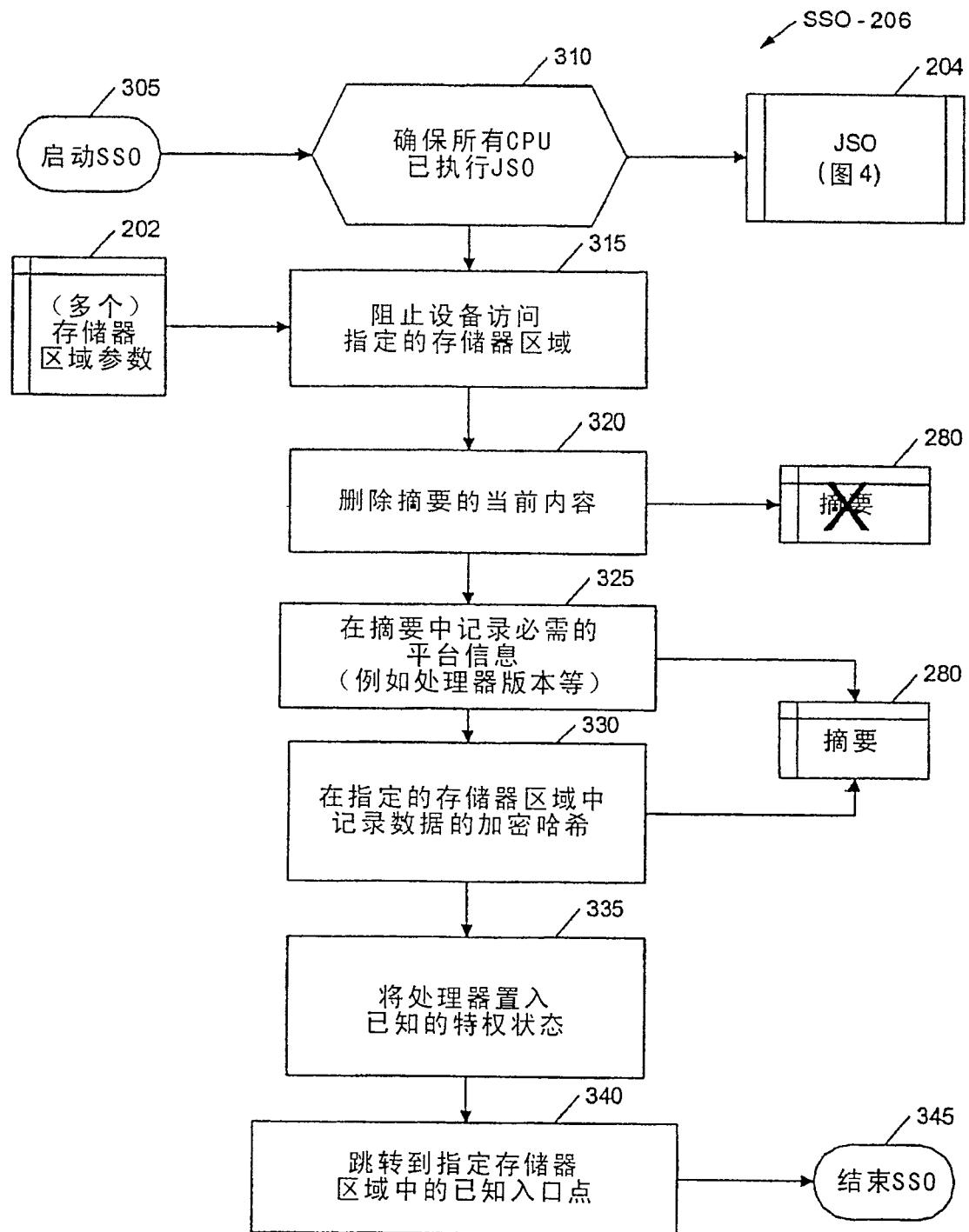


图3

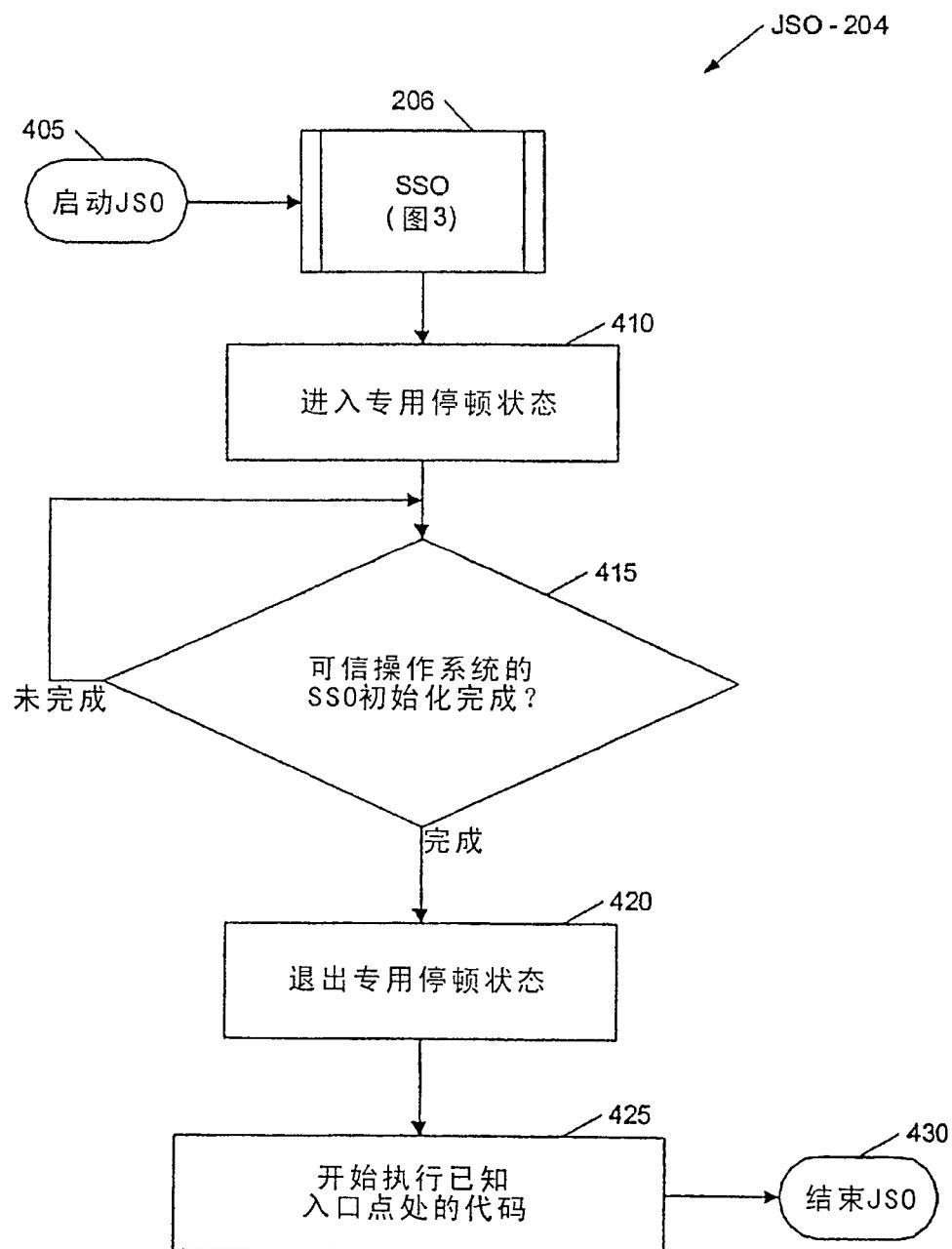


图4

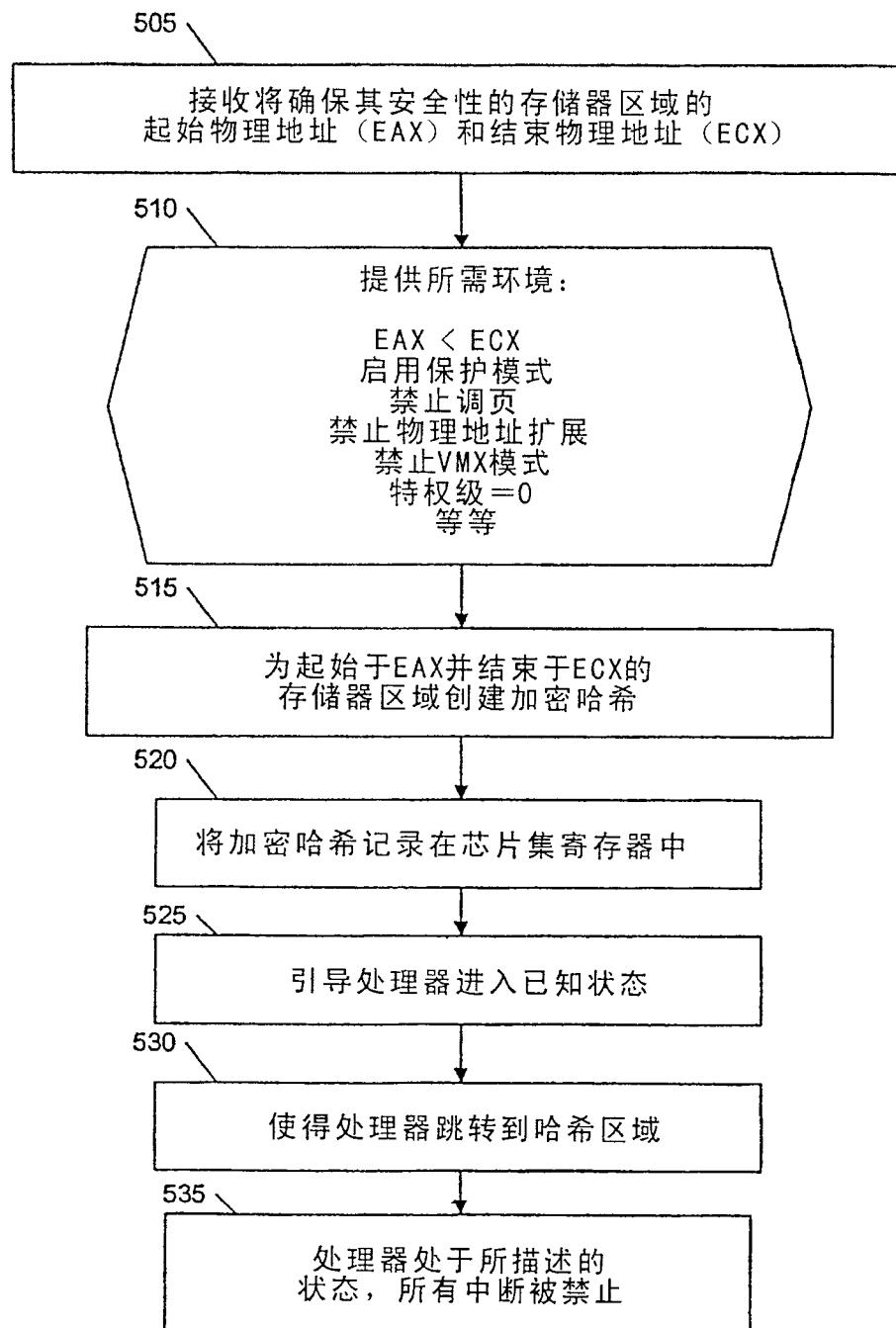


图5

