



(12) 发明专利申请

(10) 申请公布号 CN 119072684 A

(43) 申请公布日 2024. 12. 03

(21) 申请号 202380035796.1

(22) 申请日 2023.03.17

(30) 优先权数据

2206164.2 2022.04.28 GB

(85) PCT国际申请进入国家阶段日

2024.10.23

(86) PCT国际申请的申请数据

PCT/GB2023/050653 2023.03.17

(87) PCT国际申请的公布数据

W02023/209323 EN 2023.11.02

(71) 申请人 ARM有限公司

地址 英国剑桥

(72) 发明人 西蒙·约翰·克拉斯克

约翰·迈克尔·霍利

(74) 专利代理机构 北京东方亿思知识产权代理
有限责任公司 11258

专利代理师 杨佳婧

(51) Int.Cl.

G06F 12/14 (2006.01)

G06F 9/30 (2006.01)

G06F 12/1009 (2006.01)

G06F 9/38 (2006.01)

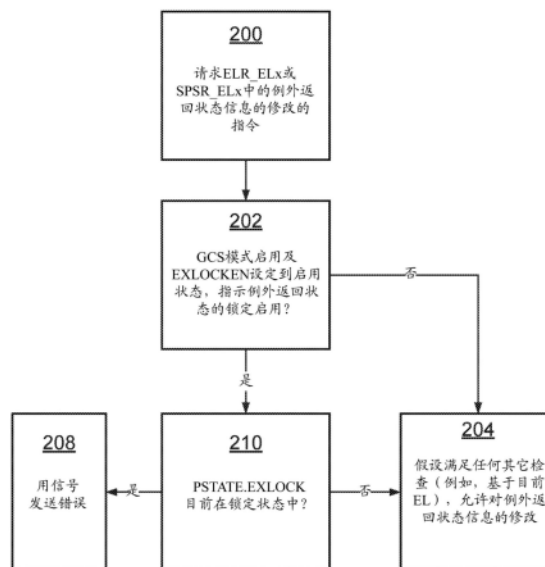
权利要求书3页 说明书21页 附图12页

(54) 发明名称

例外返回状态锁定参数

(57) 摘要

一种设备,包括例外返回状态寄存器存储器及处理电路系统。响应于保护控制堆叠(GCS)例外返回状态推送指令,该处理电路系统从该例外返回状态寄存器存储器获得例外返回状态信息,并将该状态信息推送到GCS数据结构。响应于GCS例外返回状态取出指令,该处理电路系统从该GCS数据结构获得GCS保护的例外返回状态信息。在至少一操作状态中,该处理电路系统响应于要修改存储在该例外返回状态寄存器存储器中的该例外返回状态信息的尝试,检测例外返回状态锁定参数是在锁定状态或未锁定状态中,且当其在该锁定状态中时,用信号发送错误。



1. 一种设备,包括:

例外返回状态寄存器存储器,所述例外返回状态寄存器存储器用以存储例外返回状态信息;以及

处理电路系统,所述处理电路系统用以响应于指令而执行处理操作;其中:

所述处理电路系统被配置为响应于保护控制堆叠 (GCS) 例外返回状态推送指令,从所述例外返回状态寄存器存储器获得所述例外返回状态信息,并将所述例外返回状态信息推送到GCS数据结构;

所述处理电路系统被配置为响应于GCS例外返回状态取出指令,从所述GCS数据结构获得GCS保护的例外返回状态信息;

所述设备包括存储器管理电路系统,以响应于确定对应于目标地址的存储器区域被指定为用于存储所述GCS数据结构的GCS区域而拒绝指定所述目标地址的写入请求,并且所述写入请求为指令的GCS存取类型的限制子集的一个限制子集所触发的GCS存储器存取请求之外的请求,指令的GCS存取类型的所述限制子集包括至少所述GCS例外返回状态推送指令;并且

所述处理电路系统被配置为在所述处理电路系统的至少一操作状态中,响应于要修改存储在所述例外返回状态寄存器存储器中的所述例外返回状态信息的尝试,检测例外返回状态锁定参数是在锁定状态或未锁定状态中,并且当检测到所述例外返回状态锁定参数在所述锁定状态中时,用信号发送错误。

2. 根据权利要求1所述的设备,其中:

所述处理电路系统被配置为在多个例外层级的一个中执行所述处理操作,所述多个例外层级与不同的权限层级相关联;并且

所述处理电路系统被配置为在所述处理电路系统的所述至少一操作状态中,响应于采取例外而将所述例外返回状态锁定参数设定到所述锁定状态,其中所述例外是在与采取所述例外之前的目前例外层级相同的例外层级下采取。

3. 根据任一前述权利要求所述的设备,其中所述处理电路系统被配置为在所述处理电路系统的所述至少一操作状态中,响应于所述GCS例外返回状态推送指令而将所述例外返回状态锁定参数设定到所述未锁定状态。

4. 根据任一前述权利要求所述的设备,其中所述处理电路系统被配置为在所述处理电路系统的所述至少一操作状态中,响应于所述GCS例外返回状态取出指令而将所述例外返回状态锁定参数设定到所述锁定状态。

5. 根据任一前述权利要求所述的设备,其中所述处理电路系统被配置为在所述处理电路系统的所述至少一操作状态中,响应于用于基于存储在所述例外返回状态寄存器存储器中的所述例外返回状态信息触发例外返回的例外返回指令,而检查所述例外返回状态锁定参数是否设定到所述锁定状态,其中所述例外返回是在与所述例外返回之前的目前例外层级相同的例外层级下的处理的返回,并且响应于检测到所述例外返回状态锁定参数设定到所述未锁定状态而用信号发送错误。

6. 根据任一前述权利要求所述的设备,其中所述处理电路系统被配置为在所述处理电路系统的所述至少一操作状态中,响应于当所述例外返回状态锁定参数设定到所述锁定状态时要执行请求对所述例外返回状态寄存器存储器中的信息的修改的指令的尝试,用信号

发送错误,无论待写入到所述例外返回状态寄存器存储器的新值是否与先前存储在所述例外返回状态寄存器存储器中的旧值相同或不同。

7. 根据权利要求1至5中任一项所述的设备,其中所述处理电路系统被配置为在所述处理电路系统的所述至少一操作状态中,响应于当所述例外返回状态锁定参数设定到所述锁定状态时要执行请求对所述例外返回状态寄存器存储器中的信息的修改的指令的尝试,检查待写入到所述例外返回状态寄存器存储器的新值是否与先前存储在所述例外返回状态寄存器存储器中的旧值相同或不同,并且当确定所述新值不同于所述旧值时,用信号发送所述错误。

8. 根据任一前述权利要求所述的设备,其中所述处理电路系统被配置为响应于所述GCS例外返回状态取出指令,当检测到从所述例外返回状态寄存器存储器获得的预期的例外返回状态信息与从所述GCS数据结构获得的所述GCS保护的例外返回状态信息之间的不匹配时,用信号发送错误。

9. 根据任一前述权利要求所述的设备,其中所述例外返回状态寄存器存储器包括例外返回地址寄存器,所述例外返回地址寄存器用以存储从例外返回之后待执行的指令的例外返回地址。

10. 根据任一前述权利要求所述的设备,其中所述例外返回状态寄存器存储器包括存储的处理器状态寄存器,所述存储的处理器状态寄存器用以存储从例外返回时待恢复的处理器状态。

11. 根据权利要求10所述的设备,其中所述处理电路系统被配置为响应于采取例外,将来自处理器状态寄存器的处理器状态复制到所述存储的处理器状态寄存器;并且

所述例外返回状态锁定参数是在所述处理器状态寄存器中指定的参数。

12. 根据权利要求11所述的设备,其中所述处理电路系统被配置为响应于例外返回指令,将来自所述存储的处理器状态寄存器的存储的处理器状态复制到所述处理器状态寄存器,以恢复所述例外返回状态锁定参数到所述存储的例外返回状态锁定参数所指示的值,所述存储的处理器状态包括存储的例外返回状态锁定参数。

13. 根据任一前述权利要求所述的设备,其中所述处理电路系统的所述至少一操作状态包括操作状态,其中例外返回状态锁定启用控制参数设定到启用状态,其指示使用所述例外返回状态锁定参数的例外返回状态信息的锁定被启用。

14. 根据权利要求1至12中任一项所述的设备,其中所述处理电路系统的所述至少一操作状态包括操作状态,其中:

例外返回状态锁定启用控制参数设定到启用状态,其指示使用所述例外返回状态锁定参数的例外返回状态信息的锁定被启用;并且

GCS启用参数设定到启用状态,其指示用于支持使用所述GCS数据结构的GCS模式被启用。

15. 根据任一前述权利要求所述的设备,其中所述存储器管理电路系统被配置为响应于确定对应于给定目标地址的存储器区域未指定为所述

GCS区域,而拒绝指定所述给定目标地址的GCS存储器存取请求。

16. 一种方法,包括:

将例外返回状态信息存储在例外返回状态寄存器存储器中;以及

使用处理电路系统,响应于指令执行处理操作;

响应于保护控制堆叠(GCS)例外返回状态推送指令,从所述例外返回状态寄存器存储器获得所述例外返回状态信息,并将所述例外返回状态信息推送到GCS数据结构;

响应于GCS例外返回状态取出指令,从所述GCS数据结构获得GCS保护的例外返回状态信息;

响应于确定对应于目标地址的存储器区域被指定为用于存储所述GCS数据结构的GCS区域而拒绝指定所述目标地址的写入请求,并且所述写入请求为指令的GCS存取类型的限制子集的一个限制子集所触发的GCS存储器存取请求之外的请求,指令的GCS存取类型的所述限制子集包括至少所述GCS例外返回状态推送指令;并且

在所述处理电路系统的至少一操作状态中,响应于要修改存储在所述例外返回状态寄存器存储器中的所述例外返回状态信息的尝试,检测例外返回状态锁定参数是在锁定状态或未锁定状态中,并且当检测到所述例外返回状态锁定参数在所述锁定状态中时,用信号发送错误。

17.一种包括指令的计算机程序,当所述指令由主机数据处理设备执行时,控制所述主机数据处理设备以提供用于执行目标代码的指令执行环境,所述计算机程序包括:

寄存器仿真程序逻辑,所述寄存器仿真程序逻辑用以控制所述主机数据处理设备以模拟用于存储例外返回状态信息的例外返回状态寄存器存储器;以及

处理程序逻辑,所述处理程序逻辑用以控制所述主机数据处理设备以模拟所述目标代码的指令的执行;其中:

所述处理程序逻辑被配置为响应于所述目标代码的保护控制堆叠(GCS)例外返回状态推送指令,控制所述主机数据处理设备,以从模拟的例外返回状态寄存器存储器获得所述例外返回状态信息,并将所述例外返回状态信息推送到GCS数据结构;

所述处理程序逻辑被配置为响应于所述目标代码的GCS例外返回状态取出指令,获得从所述GCS数据结构获得的GCS保护的例外返回状态信息;

所述计算机程序包括存储器管理程序逻辑,所述存储器管理程序逻辑用以响应于确定对应于目标地址的存储器区域被指定为用于存储所述GCS数据结构的GCS区域而拒绝指定所述目标地址的写入请求,并且所述写入请求为指令的GCS存取类型的限制子集的一个限制子集所触发的GCS存储器存取请求之外的请求,指令的GCS存取类型的所述限制子集包括至少所述GCS例外返回状态推送指令;并且

所述处理程序逻辑被配置为在所述处理程序逻辑的至少一模拟的操作状态中,控制所述主机数据处理设备,以响应于要修改存储在所述模拟的例外返回状态寄存器存储器中的所述例外返回状态信息的尝试,检测例外返回状态锁定参数是在锁定状态或未锁定状态中,并且当检测到所述例外返回状态锁定参数在所述锁定状态中时,用信号发送错误。

18.一种存储介质,所述存储介质存储根据权利要求17所述的计算机程序。

例外返回状态锁定参数

[0001] 本技术涉及数据处理的领域。

[0002] 数据处理设备可响应于采取例外而存储例外返回状态信息。例外返回状态信息指示在采取例外的时间时的处理器状态,使得当例外处理完成时,处理器可返回采取例外之前被执行的处理。

[0003] 至少一些示例提供一种设备,其包括:例外返回状态寄存器存储器,该例外返回状态寄存器存储器用以存储例外返回状态信息;以及处理电路系统,该处理电路系统用以响应于指令而执行处理操作;其中:该处理电路系统被配置为响应于保护控制堆叠(GCS)例外返回状态推送指令,从该例外返回状态寄存器存储器获得该例外返回状态信息,并将该例外返回状态信息推送到GCS数据结构;该处理电路系统被配置为响应于GCS例外返回状态取出指令,从该GCS数据结构获得GCS保护的例外返回状态信息;该设备包括存储器管理电路系统,以响应于确定对应于目标地址的存储器区域被指定为用于存储该GCS数据结构的GCS区域而拒绝指定该目标地址的写入请求,并且该写入请求为指令的GCS存取类型的限制子集的一个限制子集所触发的GCS存储器存取请求之外的请求,指令的GCS存取类型的该限制子集包括至少该GCS例外返回状态推送指令;并且该处理电路系统被配置为在该处理电路系统的至少一操作状态中,响应于要修改存储在该例外返回状态寄存器存储器中的该例外返回状态信息的尝试,检测例外返回状态锁定参数是在锁定状态或未锁定状态中,并且当检测到该例外返回状态锁定参数在该锁定状态中时,用信号发送错误。

[0004] 至少一些示例提供一种方法,其包括在例外返回状态寄存器存储器中存储例外返回状态信息;以及使用处理电路系统,响应于指令执行处理操作;响应于保护控制堆叠(GCS)例外返回状态推送指令,从该例外返回状态寄存器存储器获得该例外返回状态信息,并将该例外返回状态信息推送到GCS数据结构;响应于GCS例外返回状态取出指令,从该GCS数据结构获得GCS保护的例外返回状态信息;响应于确定对应于目标地址的存储器区域被指定为用于存储该GCS数据结构的GCS区域而拒绝指定该目标地址的写入请求,并且该写入请求为指令的GCS存取类型的限制子集的一个限制子集所触发的GCS存储器存取请求之外的请求,指令的GCS存取类型的该限制子集包括至少该GCS例外返回状态推送指令;并且在处理电路系统的至少一操作状态中,响应于要修改存储在该例外返回状态寄存器存储器中的该例外返回状态信息的尝试,检测例外返回状态锁定参数是在锁定状态或未锁定状态中,并且当检测到该例外返回状态锁定参数在该锁定状态中时,用信号发送错误。

[0005] 至少一些示例提供一种包括指令的计算机程序,当这些指令由主机数据处理设备执行时,控制该主机数据处理设备以提供用于执行目标代码的指令执行环境,该计算机程序包括:寄存器仿真程序逻辑,该寄存器仿真程序逻辑用以控制该主机数据处理设备以模拟用于存储例外返回状态信息的例外返回状态寄存器存储器;以及处理程序逻辑,该处理程序逻辑用以控制该主机数据处理设备以模拟该目标代码的指令的执行;其中:该处理程序逻辑被配置为响应于该目标代码的保护控制堆叠(GCS)例外返回状态推送指令,控制该主机数据处理设备,以从该模拟的例外返回状态寄存器存储器获得该例外返回状态信息,并将该例外返回状态信息推送到GCS数据结构;该处理程序逻辑被配置为响应于该目标代

码的GCS例外返回状态取出指令,获得从该GCS数据结构获得的GCS保护的例外返回状态信息;该计算机程序包括存储器管理程序逻辑,该存储器管理程序逻辑用以响应于确定对应于目标地址的存储器区域被指定为用于存储该GCS数据结构的GCS区域而拒绝指定该目标地址的写入请求,并且该写入请求为指令的GCS存取类型的限制子集的一个限制子集所触发的GCS存储器存取请求之外的请求,指令的GCS存取类型的该限制子集包括至少该GCS例外返回状态推送指令;并且该处理程序逻辑被配置为在该处理程序逻辑的至少一模拟的操作状态中,控制该主机数据处理设备,以响应于要修改存储在该模拟的例外返回状态寄存器存储器中的该例外返回状态信息的尝试,检测例外返回状态锁定参数是在锁定状态或未锁定状态中,并且当检测到该例外返回状态锁定参数在该锁定状态中时,用信号发送错误。

[0006] 该计算机程序可存储在存储介质上。该存储介质可为非暂时性存储介质。

[0007] 本技术的进一步方面、特征、以及优点将由于结合附图阅读的以下示例描述而显而易见,在这些附图中:

[0008] 图1示出具有处理电路系统及存储器管理电路系统的数据处理设备的示例;

[0009] 图2示出处理电路系统的执行状态的示例;

[0010] 图3示出二阶段地址转换;

[0011] 图4示出处理电路系统的寄存器的示例;

[0012] 图5示出在进入采取的例外时及从该例外返回时执行的指令的示例;

[0013] 图6示出使用保护控制堆叠(guarded control stack,GCS)结构以保护响应于例外进入而检索的返回状态信息;

[0014] 图7示出当使用GCS结构时,在进入采取的例外时及从该例外返回时执行的指令的示例;

[0015] 图8为示出一种检查存储器存取请求是否被允许的方法的流程图;

[0016] 图9为示出在例外进入时执行的步骤的流程图;

[0017] 图10为显示响应于GCS例外返回状态推送指令而执行的步骤的流程图;

[0018] 图11为显示响应于GCS例外返回状态取出指令而执行的步骤的流程图;

[0019] 图12为显示响应于例外返回指令而执行的步骤的流程图;

[0020] 图13及图14为显示响应于请求修改例外返回状态寄存器存储器中存储的例外返回状态信息的指令而执行的步骤的两个替代示例的流程图;并且

[0021] 图15示出模拟器实现方案。

[0022] 一种设备具有:例外返回状态寄存器存储器,该例外返回状态寄存器存储器用以存储例外返回状态信息;以及处理电路系统,该处理电路系统用以响应于指令而执行处理操作。

[0023] 面向返回编程(Return-oriented-programming,ROP)的攻击是数据处理系统上常见的攻击类别。ROP攻击是通过损坏用以从功能呼叫或例外返回的返回状态信息而尝试使程序以非预期方式表现的攻击。通常,软件会将返回状态信息存储到存储器,例如,以促进功能呼叫或例外的嵌套,其中在可以用于内部功能呼叫或例外的返回状态信息在寄存器中覆盖用于(功能呼叫或例外的嵌套集合的)外部功能呼叫或例外的返回状态信息之前,用于外部功能呼叫或例外的返回状态信息存储到存储器以对其进行保存。成功的ROP攻击可导致功能返回或例外返回将程序流程返回到功能被呼叫或例外被采取的点之后的下一指令

之外的指令,这可允许攻击者控制处理电路系统以执行程序设计师所预期的操作序列以外的任意操作。

[0024] 为了保护免于篡改例外返回状态信息的基于ROP的攻击,有用的是提供保护控制堆叠(GCS)数据结构,其用以保护用于从例外返回的返回状态信息。响应于保护控制堆叠(GCS)例外返回状态推送指令,该处理电路系统从该例外返回状态寄存器存储器获得该例外返回状态信息,并将该例外返回状态信息推送到GCS数据结构。响应于GCS例外返回状态取出指令,该处理电路系统从该GCS数据结构获得GCS保护的例外返回状态。可由存储器管理电路系统提供对于GCS数据结构的保护,该存储器管理电路系统支持指定存储器区域作为预期用于存储GCS数据结构的GCS区域。当指定目标地址的写入请求为指令的GCS存取类型的限制子集的一个限制子集所触发的GCS存储器存取请求之外的请求时(指令的GCS存取类型的该限制子集包括至少该GCS例外返回状态推送指令),该存储器管理电路系统响应于确定对应于该目标地址的存储器区域被指定为GCS区域而拒绝该写入请求。因此,可定义不能由所有类型的写入请求写入的存储器的保护区域(例如,一般目的(非GCS特定)存储指令可从指令的GCS存取类型的限制子集排除)。此限制攻击者尝试损坏存储在GCS数据结构中的GCS保护的例外返回状态可用的攻击面。因此,对于例外返回状态,可提供比使用由标准可读取/可写入存储器区域中的软件维持的通用目的堆叠更强的保护。

[0025] 然而,发明人认识到,可能存在例外进入(当在例外返回状态寄存器存储器中检索例外返回状态时)与GCS例外返回状态推送指令的执行之间的时间窗,以及GCS例外返回状态取出指令的执行与例外返回之间的另一时间窗,例外返回状态信息在这些时间窗可能容易受到寄存器中的篡改,如果攻击成功可绕过基于存储在GCS数据结构中的保护的例外返回状态信息的检查。如果攻击者可在例外进入与GCS例外返回状态推送指令的执行之间修改例外返回状态寄存器存储器中的例外返回状态,则错误的例外返回状态可被推送到GCS数据结构,并且因此在GCS数据结构上被误认为是受信任的。类似地,一些攻击可基于在执行GCS例外返回状态取出指令之后但在执行例外返回之前修改例外返回状态寄存器存储器中的例外返回状态。

[0026] 用于解决此问题的一种方法可要求响应于例外进入本身,GCS例外返回状态推送指令的操作替代地以硬件执行,并且响应于触发例外返回的相同指令而执行GCS例外返回状态取出指令的操作,以消除这些时间窗。然而,此方法可能增加单一事件(例外进入及返回)所需触发的操作的数目。响应于单一触发事件所需的动作的此类过载可使处理器微架构设计者实施支持此些操作同时满足电路时序要求的电路硬件更加复杂。再者,将该GCS例外返回状态推送指令与例外进入分离以及将该GCS例外返回状态取出指令与例外返回分离也具有优点,软件有更多弹性确定哪个例外进入/返回事件要保护或不保护,因此可能不想要组合此些操作。

[0027] 在下文讨论的示例中,提供例外返回状态锁定参数,其可设定到锁定状态或未锁定状态。在该处理电路系统的至少一操作状态中,该处理电路系统响应于要修改存储在该例外返回状态寄存器存储器中的该例外返回状态信息的尝试,检测例外返回状态锁定参数是在锁定状态或未锁定状态中,并且当检测到该例外返回状态锁定参数在该锁定状态中时,用信号发送错误。通过提供架构机制以锁定例外返回状态信息以防止其被修改,此可防止在例外进入及例外返回状态推送到GCS数据结构之间以及在GCS保护的例外返回状态信

息的取出及后续的例外返回之间的窗中篡改例外返回状态信息,以改善由例外返回状态信息的GCS型检查所提供的安全性。

[0028] 该处理电路系统可支持在多个例外层级中的一个例外层级中的指令的执行,该多个例外层级与不同的权限层级相关联。在该处理电路系统的该至少一操作状态中,该处理电路系统可响应于采取例外而将该例外返回状态锁定参数设定到该锁定状态,其中该例外是在与采取该例外之前的目前例外层级相同的例外层级下采取。因此,对于在与采取该例外之前执行的处理相同的例外层级中采取的例外,例外进入可导致处理电路系统锁定例外返回状态信息免于修改,以在所检索的例外返回状态信息可被推送到GCS数据结构之前,保护响应于例外进入所检索的例外返回状态信息免于篡改。该例外返回状态锁定参数的设定到锁定状态可响应于采取例外而自动发生,而不需要在例外进入之后执行明确的软件指令以将例外返回状态锁定参数设定到锁定状态。

[0029] 例外返回状态锁定参数不必然响应于例外进入就设定到锁定状态,因为该例外是在比采取该例外之前的目前的例外层级更高权限的例外层级中采取。当在更高权限的层级中采取例外,则可信赖更高权限的例外处理代码适当地管理该例外返回。当在相同例外层级采取例外时,攻击更可能发生。然而,其它实现方案可选择当在更高权限的例外层级采取例外时也锁定例外返回状态信息。

[0030] 在该处理电路系统的该至少一操作状态中,该处理电路系统可响应于该GCS例外返回状态推送指令而将该例外返回状态锁定参数设定到该未锁定状态。因此,一旦该例外返回状态信息已被推送到GCS数据结构,则不再需要防止例外返回状态寄存器存储器中的例外返回状态信息的修改,并且因此通过切换锁定参数到未锁定状态,此允许需要可有效改变例外返回状态寄存器存储器中的信息的信息的后续操作进行。响应于GCS例外返回状态推送指令而未锁定例外返回状态锁定参数也有助于改善安全性,这是通过允许检测当在例外返回状态修改应锁定的时间下执行的指令(例如,例外返回指令)是在例外返回状态修改目前未锁定的时间下的不正确程序流程序列中执行时的情况。

[0031] 在该处理电路系统的该至少一操作状态中,该处理电路系统可响应于该GCS例外返回状态取出指令而将该例外返回状态锁定参数设定到该锁定状态。GCS例外返回状态取出指令可标示窗的开始,其中从GCS数据结构所取出的例外返回状态容易遭受篡改攻击,因此响应于GCS例外返回状态取出指令将例外返回状态锁定有助于给予信赖度:在后续的例外返回,用于例外返回的例外返回状态信息仍与从GCS数据结构获得的相同。

[0032] 在该处理电路系统的该至少一操作状态中,该处理电路系统可响应于用于基于存储在该例外返回状态寄存器存储器中的该例外返回状态信息触发例外返回的例外返回指令(其中该例外返回是在与该例外返回之前的目前例外层级相同的例外层级下的处理的返回),而检查该例外返回状态锁定参数是否设定到该锁定状态,并且响应于检测到该例外返回状态锁定参数设定到该未锁定状态而用信号发送错误。此可帮助保护免于攻击,这些攻击尝试控制程序流程以使例外返回指令被执行,而未先使用GCS例外返回状态取出指令获得GCS保护的例外返回状态信息。

[0033] 可采取不同方法以确定当执行指令时(当例外返回状态锁定参数设定到锁定状态时,其可导致例外返回状态寄存器存储器中的信息的修改),是否应用信号发送错误。

[0034] 在一示例中,在该处理电路系统的该至少一操作状态中,该处理电路系统可响应

于当该例外返回状态锁定参数设定到该锁定状态时要执行请求对该例外返回状态寄存器存储器中的信息的修改的指令的尝试,用信号发送错误,无论待写入到该例外返回状态寄存器存储器的新值是否与先前存储在该例外返回状态寄存器存储器中的旧值相同或不同。因此,在此情况中,当此类修改被锁定时,如果指令尝试修改例外返回状态寄存器存储器中的信息,将用信号发送错误,即使该指令将写入的新值将相同于先前存储在例外返回状态寄存器存储器中的旧值。即使新值相同于旧值,例外返回状态信息并无修改,可较简单的是(从硬件设计者寻求实施用于根据例外返回状态锁定参数控制错误的传信的控制电路逻辑的观点看来)使错误的传信无关于旧与新值之间的任何比较,因为不需要读取目前存储在例外返回状态寄存器存储器中的值以确定是否触发错误。

[0035] 在另一示例中,在该处理电路系统的该至少一操作状态中,响应于当该例外返回状态锁定参数设定到该锁定状态时要执行请求对该例外返回状态寄存器存储器中的信息的修改的指令的尝试,该处理电路系统可检查待写入到该例外返回状态寄存器存储器的新值是否与先前存储在该例外返回状态寄存器存储器中的旧值相同或不同,并且当确定该新值不同于该旧值时,用信号发送该错误。此方法可有用于加速处理某些快速例外程序码路径,对于此类路径,例外处理器如此近,以致于不需要存储/恢复例外返回状态到/从存储器中的堆叠。在此情况下,例外处理器可偏向不执行任何GCS例外返回状态推送/取出指令,因为最初检索的例外返回状态可在整个例外处理器的执行中维持在例外返回状态寄存器存储器中。此可意谓例外返回状态锁定参数在整个例外处理器中可保持在锁定状态。可存在一些合法原因,为何例外处理器可仍然尝试修改例外返回状态寄存器存储器。虽然当锁定参数在锁定状态时如果尝试修改存储于例外返回状态寄存器存储器中的例外返回状态信息,应因此触发错误,但有时尝试修改可实际上写入已存储在例外返回状态寄存器存储器中的相同值,在此情况下,例外返回状态并没有被损坏的风险。例如,虽然没有因为在如上文所论述的快速例外程序码路径中而意图改变例外返回状态信息,但如果例外处理器程序码包括与其它使用情况(其中例外返回状态信息真的需要修改)共享的共同程序码路径,这可能发生,并且无论如何,优选是(通过避免条件式分支以加速性能,即)执行写入到例外返回状态寄存器的指令,但在该指令与该快速例外程序码路径一起使用的情况中,设定用于该指令的操作数,使得其写入先前存储在例外返回状态寄存器中的相同值。因此,在修改指令将不会改变在例外返回状态寄存器存储器中的值的时机,通过在此时机不用信号发送错误,这可通过消除传信错误所导致的延迟而改善性能(否则传信错误将中断目前的处理,并且使另一例外处理器执行)。

[0036] GCS例外返回状态取出指令可以不同方式实施。

[0037] 在一些示例中,GCS例外返回状态取出指令可仅将GCS保护的例外返回状态信息取出到寄存器存储器中(直接到例外返回状态寄存器存储器,或到通用寄存器,后续指令可从其移动GCS保护的例外返回状态信息到例外返回状态寄存器存储器)。接着,当执行后续例外返回指令时,GCS保护的例外返回状态信息可直接用于控制例外返回。在此情况下,不需要软件管理例外返回状态信息的存储/恢复到正常存储器中的软件管理的堆叠结构,因为GCS例外返回状态推送/取出指令的执行意指GCS数据结构足以跟踪例外返回状态,即使嵌套的例外导致例外返回状态寄存器存储器中的例外返回状态被覆盖。

[0038] 在其它示例中,并非用于直接控制例外返回,GCS保护的例外返回状态可用以验证

存储在例外返回状态寄存器存储器中的预期的例外返回状态信息对于后续例外返回是否是安全可用。在此情况下,软件可使用其正常机制管理例外返回状态寄存器存储器中的例外返回状态信息(例如,如果嵌套例外,软件存储/恢复例外返回状态到非GCS存储器中维持的堆叠数据结构)。GCS例外返回状态推送/取出指令额外地存储/恢复例外返回状态到GCS数据结构,以允许通过软件所获得的预期的例外返回状态信息在其用于控制例外返回之前受检查。

[0039] 因此,在此示例中,除了使GCS保护的例外返回状态信息从GCS数据结构取出之外,响应该GCS例外返回状态取出指令,该处理电路系统当检测到从该例外返回状态寄存器存储器获得的预期的例外返回状态信息及从该GCS数据结构获得的该GCS保护的例外返回状态信息之间的不匹配时,也用信号发送错误。因此,使用GCS保护的例外返回状态信息以验证预期的例外返回状态信息,而非直接用以控制例外返回。

[0040] 在另一示例中,虽然GCS保护的例外返回状态信息可用以验证预期的例外返回状态信息是否正确,此验证可能响应于GCS例外返回状态取出指令本身并未发生,但取而代之的是,后续GCS例外返回状态验证指令可验证例外返回状态寄存器存储器中的预期的例外返回状态信息是否匹配GCS例外返回状态取出指令所获得的先前取出的GCS例外返回状态信息。

[0041] 存储在例外返回状态寄存器存储器中的例外返回状态信息可采取各种形式。一般来说,例外返回状态寄存器存储器可包括存储例外返回状态信息的一个或多个寄存器。例如,例外返回状态寄存器存储器可包括例外返回地址寄存器,其用以存储从例外返回之后要执行的指令的例外返回地址;和/或存储的处理器状态寄存器,其用以存储从例外返回时要恢复的处理器状态。响应于GCS例外返回状态推送指令,存储在这些寄存器中的一个或两个中的信息可存储到GCS结构。

[0042] 存储的处理器状态寄存器可为寄存器,其中响应于采取例外,处理电路系统的目前处理器状态从处理器状态寄存器复制到存储的处理器状态寄存器。由于采取例外可能导致目前处理器状态改变,通过将目前处理器状态复制到存储的处理器状态寄存器,使得处理电路系统接着可根据采取例外时的状态自由地更新目前处理器状态,可检索目前处理器状态。在软件中执行的例外处理器可接着读取存储的处理器状态寄存器并将检索的状态存储到存储器,以确保存储的处理器状态被保留,即使另一例外发生而预清空先前的例外的处理。使用GCS例外状态推送指令,也可将该存储的处理器状态推送到GCS数据结构,以实现在对应的例外返回下验证例外返回状态信息。

[0043] 可有用的是,该例外返回状态锁定参数是在该处理器状态寄存器中指定的参数。这意味着当采取例外时,例外返回状态锁定参数自身将自动被存储,因为响应于采取例外,例外返回状态锁定参数可与处理器状态寄存器的其它内容一起复制到存储的处理器状态寄存器。类似地,响应于例外返回指令,从存储的处理器状态寄存器复制存储的处理器状态到处理器状态寄存器导致例外返回状态锁定参数的先前值被恢复。因此,此方法意指,如果在例外进入及GCS例外返回状态推送指令之间的窗或在GCS例外返回状态取出指令及例外返回之间的窗的期间采取新例外,则例外返回状态锁定参数的目前状态(其在那些窗期间应在锁定状态中)被检索且因此可在从新例外返回时被恢复作为存储的处理器状态的正常恢复的部分,以返回到采取新例外的时间时的例外返回状态锁定参数的先前状态。因此,攻

击者不能使用触发新例外作为尝试绕过例外返回状态锁定参数提供的保护的机制。

[0044] 基于例外返回状态锁定参数监督对于例外返回状态寄存器存储器的修改不必用于处理电路系统的所有操作状态中。例如,存储于一个或多个控制寄存器中的一些控制状态信息可用以确定是否应执行上文所描述的例外返回状态锁定参数的各种检查。停用此些检查可有用于向后兼容于针对不支持例外返回状态锁定参数的旧有系统已写的软件。此类旧有软件可能需要软件开发人员的更多警戒,以确保没有漏洞可供攻击者利用来安装ROP攻击。

[0045] 因此,在一示例中,该处理电路系统的该至少一操作状态包括操作状态,其中例外返回状态锁定启用控制参数设定到启用状态,其指示使用该例外返回状态锁定参数的例外返回状态信息的锁定被启用。如果例外返回状态锁定启用控制参数被设定为停用状态,则不需要更新或检查如上文所论述的例外返回状态锁定参数。

[0046] 在一些示例中,支持使用GCS数据结构(例如GCS例外返回状态推送/取出指令、与GCS数据结构相互作用的其它指令、以及经触发以保护GCS数据结构免于未授权存取的各种错误)的架构机制可根据GCS启用参数的状态而完全停用。再者,这可提供支持针对不支持此些GCS型架构机制的旧有系统所写的软件。因此,处理电路系统的至少一操作状态(其中上文所描述的各种措施涉及例外返回状态锁定参数采取)可包括操作状态,其中:例外返回状态锁定启用控制参数设定到启用状态,其指示使用该例外返回状态锁定参数的例外返回状态信息的锁定被启用;并且GCS启用参数设定到启用状态,其指示用于支持使用该GCS数据结构的GCS模式被启用。

[0047] 如上文所论述,响应于确定对应于目标地址的存储器区域被指定为GCS区域且该写入请求为被允许存取GCS区域的指令的GCS存取类型的限制子集的一个限制子集所触发的GCS存储器存取请求之外的请求,该存储器管理电路系统拒绝指定该目标地址的该写入请求。该存储器管理电路系统还可响应于确定该存储器存取请求为该GCS存储器存取请求且对应于该目标地址的该存储器区域并非该GCS区域,而拒绝该存储器存取请求。此减少可用于攻击者的攻击面,因为其意指GCS存储器存取请求仅可用于存取GCS区域,并且不能用于存取用以存储其它数据的正规存储器区域。这意味着,如果GCS存取类型的指令的地址操作数被攻击者修改(例如,被攻击者导致不正确,强迫进行程序流程的不正确序列)以使GCS存储器存取请求存取非GCS区域,此被检测为错误,并且要存取存储器的该请求将被拒绝。这通过避免攻击者能够通过以存储器的另一区域取代GCS数据结构而伪造GCS保护的例外返回状态,来改善安全性。

[0048] 虽然上文描述GCS数据结构是用于保护例外返回状态信息免于ROP攻击,但相同的数据结构也可用于保护功能返回状态信息,例如功能返回地址。可提供机制,其用于检测从GCS数据结构所获得的信息是否是例外返回状态信息或功能返回状态信息(例如,通过区分有效例外返回状态记录及有效功能返回状态记录的编码,使得其可被区分),使得GCS例外返回状态验证指令可导致处理电路系统检查使用作为GCS保护的例外返回状态信息的所获得GCS记录是否对应于有效例外返回状态记录,并且因此通过使用功能返回状态记录以验证用于例外返回的例外返回状态信息,而监督免于导致不正确行为的尝试。

[0049] 在其它示例中,GCS数据结构可用于保护例外返回状态信息,但不用于保护功能返回状态信息(例如,不同的机制可用于保护功能返回状态免于ROP攻击,或者在用于功能返

回状态信息的架构层级处并未提供保护免于ROP攻击(取代地,保护功能返回免于ROP攻击的责任可落在软件开发者上))。因此,例外返回状态及功能返回状态之间不必然共享GCS数据结构。

[0050] 上文讨论的技术可实施在数据处理设备内,该数据处理设备具有针对实施上文讨论的处理电路系统及存储器管理电路系统而提供的硬件电路系统。

[0051] 然而,相同技术也可实施在计算机程序内,该计算机程序是在主机数据处理设备上执行以提供用于目标代码的执行的指令执行环境。即使主机数据处理设备本身不支持该架构,此计算机程序可控制主机数据处理设备以模拟其将提供在实际支持根据某个指令集架构的目标代码的硬件设备上的架构环境。计算机程序可具有处理程序逻辑、寄存器仿真程序逻辑及存储器管理程序逻辑(其仿真处理电路系统的功能)、上文所论述的例外返回状态寄存器存储器及存储器管理电路系统、包括支持GCS例外返回状态推送指令及GCS例外返回状态取出指令、以及例外返回状态锁定参数,如上文所论述。例如,当针对一个指令集架构所编写的遗留代码是在支持不同指令集架构的主机处理器上执行时,此模拟程序可为有用的。再者,由于软件在模拟执行环境上的执行可使软件测试得以与支持新架构的硬件装置的进行中开发平行,模拟可允许针对较新版本的指令集架构的软件开发在支持该新架构版本的处理硬件就绪之前开始。模拟程序可存储在存储介质上,该存储介质可为非暂时性存储介质。

[0052] 数据处理设备的特定示例

[0053] 图1示意地示出数据处理设备2的示例。数据处理设备具有包括若干个管线级的处理管线4。在此示例中,管线级包括用于从指令高速缓冲存储器8提取指令的提取级6;用于解码提取程序指令的解码级10,以产生待由管线的其余级处理的微操作(micro-operation)(经解码指令);用于检查微操作所需的操作数在寄存器文件14中是否可用,并发布用于将给定微操作的所需操作数执行一次的微操作为可用的发布级12;用于执行对应于通过处理读自寄存器文件14的操作数以产生结果值的微操作的数据处理操作的执行级16;以及用于将处理的结果写回到寄存器文件14的写回级18。应了解,此仅为可能的管线架构的一个示例,并且其它系统可具有额外级或不同的级配置。例如,在乱序处理器中,可包括用于将由程序指令或微操作指定的架构寄存器映射到识别寄存器文件14中的物理寄存器的物理寄存器说明符的寄存器重命名级。在一些示例中,在由解码级10解码的程序指令与由执行级所处理的对应微操作之间可有一对一关系。在程序指令与微操作之间也可能有一对多或多对一关系,使得,例如,单一程序指令可分成二个或更多个微操作,或者二个或更多个程序指令可融合以作为单一微操作来处理。

[0054] 执行级16包括若干个处理单元,以用于执行不同类别的处理操作。例如,执行单元可包括用于对读自寄存器14的标量操作数进行算术或逻辑运算的标量算术/逻辑单元(arithmetic/logic unit,ALU)20;用于对浮点值进行运算的浮点单元22;用于评估分支操作的结果及调整该程序计数器的分支单元24,该程序计数器据此表示目前的执行点;以及用于执行载入/存储操作以存取存储器系统8、30、32、34中的数据的载入/存储单元26。提供存储器管理单元(memory management unit,MMU),其为存储器管理电路系统28的示例,用于执行由载入/存储单元26基于数据存取指令的操作数所指定的虚拟地址与识别存储器系统中数据的存储位置的物理地址之间的地址转换。MMU具有转换后备缓冲区(translation

lookaside buffer, TLB) 29, 其用于从存储在存储器系统中的分页表高速缓存地址转换数据, 其中分页表的分页表项目定义地址转换映射及存取权限, 这些存取权限管控例如是否允许管线上执行的给定程序从给定存储器区域读取、写入或执行指令。

[0055] 在此示例中, 存储器系统包括一级数据高速缓冲存储器30、一级指令高速缓冲存储器8、共享二级高速缓冲存储器32、以及主系统存储器34。将理解, 此仅为可能的存储器层级的一个示例, 并可提供高速缓冲存储器的其它布置。显示在执行级16中的处理单元20到26的具体类型仅为一个示例, 并且其它实现方案可具有不同组的处理单元或可包括相同类型的处理单元的多个示例, 使得可平行地处理多个相同类型的微操作。应理解, 图1仅为可能的处理器管线实现方案的一些组件的简化表示, 并且处理器可包括为了简洁起见而未示出的许多其它元件。虽然图1显示存取存储器34的单一处理器核心, 该设备2还可具有一个或多个另外的处理器核心共享对于存储器34的存取, 其中每一核心具有个别的高速缓冲存储器8、30、32。

[0056] 该设备还包括例外处理电路系统40以控制例外的处理。响应于例外信号42, 例外处理电路系统40中断管线4正在执行的处理, 并且控制该管线4以切换到对应于由该例外信号42所指示的例外的类型的例外处理器的处理。举例来说, 例外可为由外部事件触发的中断, 该外部事件并非直接由处理管线4正在执行的软件引起, 例如, 用信号发送外部事件的中断, 例如用户按压包括处理设备的电子装置上的按钮、处理器复位被确立、从网络接口或其它输入/输出装置接收消息等。此外, 例外可为由管线4正在执行的指令直接引起的软件触发例外。例如, 软件触发例外可由软件通过执行指令来自动触发, 执行该指令导致采取“监督器呼叫”中断, 以触发切换到更高权限的执行状态。替代地, 软件触发例外可能是由软件非自动触发的例外, 其由于软件所执行的至少指令发生的错误而发生, 例如未定义的指令错误或地址错误。

[0057] 图2为示出处理电路系统4可在执行指令时操作的不同执行状态 (还称为例外层级) 的图。在此示例中, 存在四个例外层级EL0、EL1、EL2、EL3, 其中例外层级EL0为最低权限的例外层级, 并且例外层级EL3为最高权限的例外层级。一般来说, 当执行在更高权限的例外层级中时, 处理电路系统可存取较低、较少权限的例外层级无法存取的一些存储器位置或寄存器14。

[0058] 在此示例中, 例外层级EL0用于执行应用程序, 这些应用程序由在例外层级EL1处执行的对应的操作系统或虚拟机管理。在相同物理平台上共同存在多个虚拟机的情况下, 则可提供在EL2操作的虚拟机监控程序, 以管理相应的虚拟机。虽然图2显示虚拟机监控程序管理虚拟机且虚拟机管理应用程序的示例, 也可能是虚拟机监控程序直接管理在EL0处的应用程序。

[0059] 虽然并非必需的, 但一些实现方案可实施处理电路系统的操作的不同的硬件分割的安全及非安全域。数据处理系统2可具有实施在处理器及存储器系统内的硬件特征, 以确保与在安全域中操作的软件程序相关联的数据及程序码隔离于在非安全域中操作的程序的存取。举例来说, 可使用一种硬件架构, 例如由英国剑桥的 **Arm[®] Limited** 提供的 **TrustZone[®]** 架构。替代地, 可使用其它硬件强制安全分割架构。安全应用程序 (受信任服务) 可在安全域中的例外层级EL0中操作, 并且安全 (受信任) 操作系统或虚拟机可在安全域中的例外层级EL1中操作。在一些实现方案中, 并不支持安全状态中的EL2, 并且虚拟机监控

程序可仅在非安全EL2中执行。在其它实现方案中,可支持在安全EL2中执行的安全虚拟机监控程序,如图2中的星号所指示。在一些示例中,可提供用于管理非安全域与安全域之间的转变的安全监测器程序在例外层级EL3中执行。其它实现方案可以硬件监督安全域之间的转变,使得可不需要安全监测器程序。

[0060] 如果在目前的例外层级EL_x的处理时采取例外,则可在相同例外层级EL_x中或在更高权限的例外层级EL_y中采取该例外,其中 $y > x$ 。与例外类型相关联的特性可指示该例外的目标例外层级。

[0061] 地址转换

[0062] MMU 28执行的一个任务是虚拟地址 (VA) 与物理地址 (PA) 之间的地址转换。在处理电路系统4上执行的软件使用虚拟地址来指定存储器位置,但这些虚拟地址可通过MMU 28转换成识别要存取的存储器系统位置的物理地址。使用虚拟地址的益处在于其允许管理软件(例如操作系统(OS))控制呈现到软件的存储器的视图。OS可控制什么存储器可见、可见到该存储器的虚拟地址、以及该存储器允许什么存取。此允许OS沙盒(sandbox)应用程序(对另一应用程序隐藏应用程序的资源)且提供潜在的硬件的抽象概念。使用虚拟地址的另一益处是OS可呈现存储器的多个片段的物理区域作为单一连续虚拟地址空间到应用程序。虚拟地址也有益于软件开发人员,其在写其应用程序时将不知道系统的确切存储器地址。利用虚拟地址,软件开发人员不需要关心物理存储器。应用程序知道OS及硬件负责一起工作以执行地址转换。

[0063] 实际上,每一应用程序可使用其自身的虚拟地址集合,其将被映射到物理系统中的不同位置。当操作系统切换于不同应用程序之间,其重新编程该映射。这意味着用于目前应用程序的虚拟地址将映射到存储器中的正确物理位置。

[0064] 虚拟地址经由映射转换成物理地址。虚拟地址与物理地址之间的映射存储在转换表(有时称为分页表)中。转换表存储在存储器中且由软件(通常是OS或虚拟机监控程序)管理。转换表并非静态,并且可随软件改变的需求更新表格。此改变虚拟及物理地址之间的映射。

[0065] 对于当处理电路系统4处于执行状态的特定子集(具体地,当处理电路系统4处于非安全EL0或非安全EL1中时)中时所执行的存储器存取,使用如图3所示的二阶段地址转换(对于其它执行状态,使用阶段1分页表的一阶段地址转换就已足够)。因此,来自非安全EL0及非安全EL1的虚拟地址使用两组表来转换。这些表支持虚拟化,并允许虚拟机监控程序虚拟化由给定虚拟机(VM)所见的物理存储器的视图(该虚拟机对应于访客操作系统及由访客操作系统控制的应用程序)。将OS控制的该组转换称为阶段1(Stage 1)。阶段1表转换虚拟地址成中间物理地址(IPA)。在阶段1中,OS认为IPA为物理地址空间。然而,虚拟机监控程序控制第二组转换,其被称为阶段2。此第二组转换将IPA转换成物理地址。

[0066] 阶段1及阶段2转换表实施为层级表结构,其包括多层级的转换表,分别用于阶段1及阶段2。阶段1及阶段2均可具有至多4层级的分页表,即,层级0(L0)、层级1(L1)、层级2(L2)及层级3(L3)。

[0067] 为了定位针对给定地址映射的物理地址,执行转换表走访,其包括一个或多个转换表查找。转换表走访为将虚拟地址转换为物理地址所需的查找的集合。对于非安全EL1及0转换方案,此集合包括阶段1转换及阶段2转换两者的查找。使用阶段1及阶段2查找的成功

转换表走访所回传的信息为：

[0068] • 所需物理地址 (基于映射到中间地址的阶段1及映射到物理地址的阶段2而转换)。

[0069] • 用于目标存储器区域的存取权限和/或存储器属性,其提供关于如何控制对该存储器区域的存取的信息。此些可包括定义在阶段1表结构中的阶段1存取权限和/或属性以及定义在阶段2表结构中的阶段2存取权限和/或属性。

[0070] 为了横越阶段1及阶段2结构的给定一个,基于在转换表型地址寄存器 (用于阶段1的TTBR,用于阶段2的VTTBR_EL2) 中指定的地址,该走访开始于初始查找的顶层级 (L0) 转换表的读取。每一转换表查找回传一描述符,其指示下列的一个:

[0071] • 该项目是阶段1或阶段2结构的横越的最终项目,其提供正在寻找的地址映射。如果该项目在最后的L3中,则此项目称为页描述符 (D_Page),而如果提供该走访的最终项目的该项目是在较高层级的一个处,则其称为块描述符 (D_Block)。该横越的最终项目含有输出地址 (OA-即,用于阶段1的IPA或用于阶段2的PA),及存取的权限及属性。如果在转换表结构的较高层级处找到块描述符,此意谓块描述符代表比在L3的单个项目所代表的4kB存储器页更大尺寸的存储器区域 (在L1及L2的块描述符所代表的特定尺寸取决于用于索引到L1或L2表中的索引位的数目-在此示例中,L1及L2块描述符分别代表1GB及2MB区域)。

[0072] • 需要查找的额外层级。在此情况下,该项目被称为表描述符 (D_Table),因为其提供用于该查找在表的进一步层级中的转换表型地址。表描述符可任选地还提供可施加到最终转换的其它阶层属性。层级1及2处的转换表项目的编码区分块描述符及表描述符。

[0073] • 描述符无效。在此情况下,存储器存取产生转换错误。

[0074] 寄存器状态

[0075] 图4展示数据处理设备的寄存器14的子集的示例,包括用于存储用于处理例外返回的架构状态的寄存器及用于保护例外返回状态免于篡改的寄存器。寄存器14包括一组通用寄存器 (GPR) 50,其用于存储通用操作数及数据处理指令及载入/存储指令的结果。提供程序计数器 (PC) 寄存器52以存储程序计数器值,其指示代表程序流程的目前点的指令的地址。处理器状态 (PSTATE) 寄存器54存储指示处理电路系统4的目前状态的一个或多个值。例如,PSTATE寄存器54包括目前例外层级EL的指示56。PSTATE寄存器54还可包括用于在解码级10处解码指令的目前指令集的指示、目前的安全状态的指示、处理电路系统4所操作的处理器模式的其它指示、以及用以指示先前处理指令的结果的条件码标志,其可用以评估对于后续条件性指令,测试条件是否满足 (条件性指令控制处理电路系统以在该测试条件满足的条件下执行一功能)。再者,如下文进一步论述,PSTATE寄存器54包括例外返回状态锁定参数 (EXLOCK) 58,用于锁定例外返回状态信息以防止更新。

[0076] 寄存器14包括数个例外返回寄存器,在此示例中,包括存储的处理器状态寄存器 (SPSR) 60及例外连结寄存器 (ELR) 62。在此示例中,SPSR 60及ELR 62为防护的寄存器,其意指此些寄存器的多个单独示例是对应于不同例外层级EL1到EL3提供。当例外被采取以在给定例外层级EL_x中处理时 (其中EL_x为EL1、EL2及EL3中的一个),存储于程序计数器寄存器52中的目前的程序计数器复制到ELR_EL_x 62,并且存储在PSTATE寄存器54中的目前的处理器状态复制到SPSR_EL_x 60 (即,复制到对应于给定的例外层级EL_x的ELR 62及SPSR 60的版本)。程序计数器的复制值可称为例外返回地址。目前的处理器状态的复制值可称为存储的

处理器状态。例外返回地址及存储的处理器状态可统称为例外返回状态信息。从在给定例外层级 $E1x$ 中处理的一例外例外返回时,将存储的处理器状态从 $SPSR_E1x$ 60复制到 $PSTATE$ 寄存器54,并且将例外返回地址从 ELR_E1x 62复制到程序计数器寄存器52,以将处理器恢复到采取例外时其所在的状态。寄存器下标 $E1x$ 表示允许指令写入这些寄存器的最低权限例外层级为例外 $E1x$ 。因此,例如,当在 $EL2$ 或 $EL3$ 中时, $SPSR_EL2$ 将可写入,但在 $EL0$ 或 $EL1$ 中时不可写入。

[0077] 寄存器14还包括保护的堆叠控制寄存器($GCSPR$) 64,其同样于每一例外层级受到防护,使得相关联例外层级 $EL1$ 、 $EL2$ 、 $EL3$ 的每一个存在版本 $GCSPR_E1x$ 。 GCS 控制寄存器64用于存储堆叠指标,其指向存储器中被称为保护控制堆叠(GCS)的数据结构,保护控制堆叠用于保护在例外时存储到 $SPSR$ 60及 ELR 62的例外返回状态。此更详细地讨论于下文。再者,寄存器包括 GCS 控制寄存器($GCSCR$) 66,其也于每一例外层级受到防护,以提供对应于例外层级 $EL1$ 、 $EL2$ 、 $EL3$ 的版本 $GCSCR_E1x$ 。当处理电路系统4在一给定例外层级 $E1x$ 中操作时,对应的堆叠指标及 GCS 控制寄存器 $GCSPR_E1x$ 及 $GCSCR_E1x$ 提供用于控制 GCS 数据结构的使用的堆叠指标及控制状态。 GCS 控制寄存器66包括至少一 GCS 启用参数68,用于控制用于保护 GCS 数据结构的使用的架构措施是否被启用或停用;以及例外返回状态锁定启用参数($EXLOCKEN$) 69,用于控制用以防止例外返回寄存器60、62的修改的例外返回状态锁定参数($EXLOCK$) 58的使用是否被启用或停用。应了解, GCS 控制寄存器66还可存储其它状态信息,用于控制在 GCS 模式中时处理电路系统的行为,在 GCS 模式中, GCS 数据结构的使用如启用参数68所指示般启用。

[0078] 虽然图4显示了寄存器60、62、64、66被防护以提供对应于处理电路系统4的不同例外状态的这些寄存器的多个版本的示例,此并非必需的,并且在其它示例中,可提供这些寄存器的任一个的单个版本,在例外状态之间共享。在此情况中,当在例外状态之间切换时,软件可切换存储在共享寄存器中的值,使得在给定的例外状态中存储在共享寄存器中的值可不同于在不同的例外状态中存储在共享寄存器中的值。还有可能,寄存器60、62、64、66的子集被防护,而寄存器60、62、64、66的另一子集在例外状态之间共享。

[0079] 图5示出在不支持使用 GCS 的系统中例外进入及返回时执行的指令的示例。图5的左手侧示出例外进入序列。响应于例外处理电路系统40采取例外,例外处理电路系统40触发处理电路系统40分别从程序计数器寄存器52及 $PSTATE$ 寄存器54复制目前的程序计数器值及目前的处理器状态到与将采取例外的例外状态 $E1x$ 相关联的例外连结寄存器62及 $SPSR$ 60。该例外的例外返回状态的复制是以硬件触发,因此不需要执行任何特定指令以引起例外返回状态的复制。例外处理电路系统40接着控制处理电路系统以执行例外处理例程。举例来说,响应于例外进入初始执行的例外处理例程可为一般例外封套数据处理器,其在分支到用于处理例外的特定原因的更特定处理器之前执行一些一般控制功能。因此,在分支到特定处理器之前,为了确保即使另外的例外发生而导致新的例外返回状态复制到寄存器60、62时存储的例外返回状态被保存,例外处理例程执行指令72以推送寄存器状态信息(包括来自 $SPSR$ 60及 ELR 62的至少例外返回状态信息)到存储器中的软件维持的数据结构,例如堆叠。用以保存例外返回状态的数据结构被组织的特定方式可以软件特定方式选择,并且可不存在此例外返回状态将如何被保存的架构定义。已将该例外返回状态存储到存储器之后,例外处理软件程序码执行称为分支指令74的功能,分支指令分支到处理该例外的原

因的特定处理器程序码。

[0080] 在从该例外返回之后,该例外处理软件程序码执行指令76,以从存储器中的该数据结构读取先前存储的寄存器状态(例如,从堆叠数据结构取出寄存器状态);指令77,以将寄存器状态恢复到例外返回寄存器60、62(例如,这可为系统寄存器修改指令,允许写入到系统控制寄存器);以及例外返回指令78,以基于存储在寄存器60、62中的例外返回状态而触发例外返回。因此,例外返回指令78使先前存储的处理器状态从SPSR_E1x 60复制到PSTATE寄存器54,并且先前存储的例外返回地址是从ELR_E1x 62复制到程序计数器寄存器52。此导致分支到由先前存储的例外返回地址指示的指令,其将以如SPSR_E1x中的先前存储的处理器状态值中所指示的处理器状态执行。

[0081] 当例外返回状态信息在由指令72写入及由指令76读取之间保持在存储器中时,攻击者可通过篡改例外返回状态信息而尝试安装面向返回编程(ROP)攻击。如果攻击者成功损坏例外返回状态,这可允许例外返回返回到不同于采取例外的程序点的程序点;或将PSTATE寄存器54所表示的处理器状态返回到不同于采取例外时的状态(例如,具有更大权限的状态),此两者可导致不期望的行为,其可能有危害安全信息或软件程序码的潜在风险。

[0082] 图6示出用于保护免于对于例外返回状态的此类ROP攻击的技术。在例外进入时,执行额外指令,以推送针对该例外在寄存器60、62中检索的例外返回状态到称为保护控制堆叠(GCS)数据结构80的保护数据结构。基于与目前例外状态E1x相关联的GCS堆叠指标寄存器64,确定例外返回状态要推送到的地址。由分页表(由MMU 28使用以控制地址转换)所指定的存取权限限定用以存储GCS数据结构80的地址区域作为特殊“GCS”存储器类型,其存取仅受限于由GCS存取指令的限制子集,并且无法使用不在该限制子集中的其它类型的指令来存取。因此,当该指令并非特殊GCS存取指令中的一个时,篡改正在执行的程序的通用载入/存储指令的地址操作数的攻击者无法使该指令存取GCS数据结构80。例如,GCS存储器类型可在由用以提供虚拟地址空间与中间地址空间之间的映射的对应的阶段1转换表项目直接或间接地指定的阶段1存取权限内指示(例如,GCS存储器类型权限可在阶段1转换表项目本身的编码中指示,或在由阶段1转换表项目所参照的寄存器中指示)。相反地,如果GCS存取指令尝试存取未指定为GCS存储器类型的存储器区域,则可用信号发送错误,以防止攻击者欺骗GCS存取指令存取GCS数据结构80以外的区域。

[0083] 在例外返回的情况下,不同于如图5所讨论地从存储器中的任何软件特定数据结构软件控制地恢复例外返回状态(预期随后由例外返回状态指令78使用,以控制例外返回),执行额外的GCS取出指令,以从基于GCSPR_E1x64中的GCS堆叠指标确定的GCS数据结构80上的位置取出保护的例外返回状态,比较保护的例外返回状态与预期的例外返回状态,并且如果检测到不匹配,用信号发送错误。这提供一种手段,以检测预期的返回状态是否在被指令76取出之前已在存储器中被篡改。如果预期的例外返回状态匹配从GCS数据结构80获得的保护的例外返回状态,则例外返回指令78可基于预期的例外返回状态来触发例外返回。

[0084] 因此,如图7所示,例外进入及返回程序码被扩展,以包括例外进入时的额外的GCS例外状态推送指令81及例外返回时的GCS例外状态验证(取出及比较)指令82,以提供由指令72推送到存储器且由指令76读回的例外返回状态的额外保护。虽然图7显示单一指令82

从GCS数据结构80取出保护的例外返回状态且验证寄存器60、62中的预期的例外返回状态与保护的例外返回状态,其它示例可将此分成两个不同的指令,一个用于从存储器取出保护的例外返回状态,并且另一个用于验证预期的例外返回状态是否匹配取出的例外返回状态。再者,在一些示例中,GCS例外返回状态取出指令82可仅从GCS数据结构80取出例外返回状态,但不验证其与例外返回状态寄存器60、62中的例外返回状态(例如,GCS例外返回状态取出指令82可直接取出GCS保护的例外返回状态到例外返回状态寄存器60、62,例外返回指令78可从寄存器60、62使用GCS保护的例外返回状态,以直接控制例外返回-在此情况中,软件将不需执行指令72、76以推送/取出例外返回状态到正常非GCS存储器中的分开的软件管理堆叠)。

[0085] 提供用于将例外返回状态推送到GCS结构80且将例外返回状态从GCS结构取出的指令81、82被视为GCS存取指令,允许存取阶段1存取权限中指定为存储器的GCS区域的存储器区域。

[0086] 应了解,当GCS启用参数68设定成指示GCS模式启用的启用状态时,可执行上文讨论的各种GCS相关功能。当GCS启用参数68被设定为停用状态以指示GCS模式停用时(例如,这可用于与未设计用于GCS模式的遗留代码的兼容性),则GCS存取指令可被处理为不具有架构功能的操作指令(如果执行GCS存取指令,导致传信错误),并且当GCS模式启用时,可省略可针对某些非GCS存取指令执行的额外检查(参见下文讨论的示例性流程图)。

[0087] GCS模式可用于支持用于维持针对给定例外所检索的有效例外返回状态的副本的存储器中的保护区域,该副本可用以直接控制例外返回,或者用以验证将用以控制该例外返回的“预期的”例外返回状态。

[0088] 然而,因为GCS例外状态推送指令80分隔于应于例外进入70的例外返回状态的硬件触发检索,存在时间窗90(在图7中标记为“窗A”),其中攻击者可尝试修改例外返回状态寄存器60、62的内容,使得响应于指令81被推送到GCS数据结构80的例外返回状态不同于响应于采取例外所检索的例外返回状态。类似地,在例外返回时,在执行GCS例外返回状态取出指令82及执行例外返回指令78之间存在时间窗92(标记为“窗B”),其中例外返回状态信息可在寄存器60、62中被修改,这可允许例外返回被触发到程序计数器值及处理器状态,其并非获得从GCS 80或与从GCS 80获得的状态相验证。

[0089] 为了解决此问题,例外返回状态锁定参数(EXLOCK) 58提供于PSTATE寄存器54中。EXLOCK 58可设定为锁定状态及未锁定状态。在一个示例中,EXLOCK=1在锁定状态中,并且EXLOCK=0在未锁定状态中。其它示例可使用相反映射,或EXLOCK 58可为多位参数,其编码例外返回状态锁定参数及其它信息,其中EXLOCK 58的状态的第一子集可视为锁定状态,并且EXLOCK 58的状态的第二子集可视为未锁定状态。因此,存在各种方式,其中EXLOCK 58可经编码以指示锁定及未锁定状态。

[0090] 当EXLOCK在未锁定状态中时,则可允许例外返回状态寄存器的修改(假如满足任何其它检查,例如目前例外层级56为足够的权限,以更新尝试被修改的例外返回状态寄存器60、62的特定防护版本)。当EXLOCK 58在锁定状态中时,则禁止对存储在寄存器60、62中的例外返回状态中的信息进行修改,即使尝试使用的是如果EXLOCK 58在未锁定状态中执行时允许修改例外返回状态信息的指令。至少对于在采取该例外之前的目前的相同的例外层级中采取的例外,可设定EXLOCK参数58,以锁定例外返回状态寄存器60、62免于如图7所

示的窗A 90与窗B 92期间的修改,以防止例外返回状态在例外进入时被检索及推送到GCS数据结构80之间以及在与从GCS数据结构80恢复的保护的例外返回状态相验证及用于例外返回78之间被篡改。可认为在比目前例外层级更高权限的例外层级中采取的例外在采取例外时不需要设定锁定参数,因为在采取例外之后执行的更高权限的程序码可受信任以适当地管理其返回状态。

[0091] 因此,提供架构机制以防止例外返回状态在存储到GCS数据结构80之前及从GCS数据结构80恢复之后被修改。采取例外时,该状态被锁定且不可修改。PSTATE.EXLOCK 58可锁定对ELR 62及SPSR 60的写入存取(至少对于尝试改变存储在寄存器60或62中的值的写入),并且在采取例外时设定。在执行推送程序81以将例外返回状态存储到GCS缓冲器80时,PSTATE.EXLOCK未锁定,允许对寄存器60、62的未来写入。在执行取出程序82以从GCS缓冲器80恢复例外返回状态时,PSTATE.EXLOCK重新锁定,预防未来写入及确保从GCS恢复的状态是不可变的。当锁定时,尝试写入到状态(ELR/SPSR)导致传信错误(其可导致采取另一例外)。替代地,仅如果写入到寄存器60、62中的一个或两个的值不同于该寄存器中的目前值,可发生此错误。这允许用于返回状态不变的情境中的快速程序码路径。

[0092] 可执行EXLOCK参数85的额外检查,使得如果发生下列的任一个,则可触发错误:

- [0093] • 当EXLOCK参数85目前处于锁定状态中时的例外进入70;
- [0094] • 当EXLOCK参数85目前在未锁定状态中时的GCS例外状态推送程序81;
- [0095] • 当EXLOCK参数85目前在锁定状态中时的GCS例外状态取出程序82;以及
- [0096] • 当EXLOCK参数85目前处于未锁定状态中时的例外返回78。

[0097] 此些事件中的任一个可指示,攻击者已损坏控制流程以遗漏GCS保护指令中的一个或导致程序码以不同于预期的顺序执行,这可为尝试ROP攻击的信号。

[0098] 例外返回状态锁定启用控制参数EXLOCKEN 69可设定为启用状态或停用状态(其再次可以类似于上文针对EXLOCK 58所论述的那些方式编码),其中启用状态指示使用例外返回状态锁定参数58的上文所讨论的例外返回状态锁定特征启用,并且停用状态指示此些例外返回状态锁定特征停用。如下文所示,如果使用设定成停用状态的GCS启用参数68将GCS模式停用或EXLOCKEN 69设定为停用状态,则可停用基于EXLOCK参数58的各种检查(及将EXLOCK参数58设定到锁定或未锁定状态的动作)。

[0099] 图8为流程图,展示当由存储在用于目前例外层级E1x的GCS控制寄存器66中的GCS启用参数68启用GCS模式时,针对发出的存储器存取由MMU 28执行的存取权限检查。在步骤100,发出存储器存取请求,其指定对应于目标存储器区域的目标虚拟地址。在步骤102,基于如图3所示的阶段1及阶段2地址转换结构中定义的相关的阶段1(及如果需要,阶段2)地址映射,MMU 28起始地址转换。可使用任何已知的地址转换技术来定义用于转换的地址映射。在步骤104,MMU 28确定阶段1权限信息(由提供目标虚拟地址的地址映射的目标阶段1转换表项目直接或间接地指定)是否指定目标存储器区域为GCS区域。举例来说,阶段1权限信息可编码在目标阶段1转换表项目本身中,或存储于由目标阶段1转换表项目参照的寄存器字段中。其它示例可使用阶段2转换表项目(经由参照寄存器而直接或间接)指定的阶段2权限信息定义GCS区域,阶段2转换表项目对应于通过目标阶段1转换表项目映射到目标VA的中间地址。此外,可针对不同的数据结构(与转换表结构不同)限定哪些存储器区域为GCS区域。

[0100] 如果目标存储器区域为GCS区域,则在步骤106处,MMU 28确定存储器存取请求是否是由允许存取GCS区域的指令的GCS存取类型的限制子集中的一个触发的GCS存储器存取请求。如果否,则在步骤107处,MMU 28确定存储器存取请求是否为写入请求。如果存储器存取请求为写入请求,则在步骤108处,用信号发送错误,以防止其它一般载入/存储指令类型能够存取GCS结构80。如果存储器存取请求为读取请求,则不需要用信号发送错误,并且该方法继续进行到步骤110(其它示例可选择针对对GCS存储器区域的非GCS读取请求也用信号发送错误,但此为任选的,因为读取请求将不会损坏GCS保护的例外返回状态信息)。

[0101] 如果存储器存取请求为GCS存储器存取请求,则在步骤110处,MMU 28确定存储器存取请求是否满足任何其它存取权限。例如,这可基于在对应于目标区域的阶段1或阶段2转换表项目中指定的其它参数,或其它安全检查,例如基于处理电路系统4的安全状态、例外层级或其它操作模式。如果未满足任何其它所需存取权限,则在步骤108中再次用信号发送错误。如果存储器存取请求需要满足的任何其它存取权限都确定满足,则在步骤112处可允许该存储器存取请求。

[0102] 如果在步骤104处,MMU 28确定目标存储器区域并非GCS区域,则在步骤114处,MMU 28确定存储器存取请求是否是由指令的GCS存取类型的限制子集中的一个触发的GCS存储器存取请求。这是与步骤106处相同的检查,但与步骤106相比有相反结果。在步骤114处,如果存储器存取请求为GCS存储器存取请求,则在步骤108处用信号发送错误。此确保GCS存储器存取请求无法用以存取存储器的非GCS区域(这将避免软件开发人员被诱导针对非GCS存取使用指令的GCS存取类型(这将增加攻击者可用的攻击面)),并且允许检测指令的GCS存取类型的地址操作数被攻击者修改而指向GCS之外的区域。通过限制指令的GCS存取类型存取GCS区域且指令的非GCS存取类型存取非GCS区域,可改善安全。

[0103] 如果在步骤114处,确定存储器存取请求并非GCS存储器存取请求,则在步骤116处,MMU 28确定存储器存取请求是否满足任何其它存取权限。举例来说,这可包括基于安全状态、例外层级或操作模式的检查,如上文所论述。如果满足所有所需存取条件,在步骤118处,允许该存储器存取请求。如果违反任何存取权限,则在步骤108用信号发送错误。

[0104] 虽然图8显示单一步骤108响应于数个不同的检查失败而用信号发送错误,但应了解,一些实现方案可根据错误的原因而用信号发送不同的错误类型。

[0105] 图8显示如果接收到存储器存取请求,同时由用于目前例外层级E1x的GCS控制寄存器66中的GCS模式启用参数68启用GCS模式,执行的检查。如果目前停用GCS模式,则可省略步骤104、106、107、114,并且可基于与如步骤110、116处所描述的GCS数据结构80的使用不相关的其它存取权限而检查所执行的检查。

[0106] 如图8所示,由支持专用GCS存储器区域类型的定义(例如使用转换表)的MMU 28在架构层级提供用于GCS数据结构80的保护,该定义指示在该区域中的地址受到额外保护,因为被要求由指令的GCS存取类型的限制子集存取而非由其它载入/存储指令存取。指令的GCS存取类型可至少包括上文在图7中论述的GCS例外返回状态推送指令81及GCS例外返回状态取出/验证指令82。还可提供指令的GCS存取类型的其它类型,例如独立的GCS例外返回状态取出指令,其从GCS数据结构80取出例外返回状态,而没有还验证其与预期的例外返回状态。可提供指令的其它GCS存取类型,用于推送及恢复功能返回状态,以保护例如用于功能返回的返回地址。

[0107] 图9为示出响应于例外进入而执行的步骤的流程图。在步骤120处,例外处理电路系统40基于例外信号42检测将采取例外,并且触发例外进入程序。响应于例外进入,在步骤122处,程序计数器及目前处理器状态是从寄存器52、54分别复制到对应于采取该例外的例外层级E1x的例外连结寄存器62及存储的处理器状态寄存器60的版本。在步骤124,处理电路系统4或例外处理电路系统40将程序计数器寄存器52更新以指示待从例外进入程序码执行的第一指令的地址,并且更新PSTATE寄存器54以指示将采取例外的处理器状态,以导致到例外进入程序码的分支,其接着将在PSTATE寄存器54所指示的新状态中执行。在步骤126,处理电路系统4确定(基于GCS启用参数68)GCS模式目前是否启用,并且确定(基于GCS例外返回状态锁定启用参数EXLOCKEN 69)例外返回状态的锁定是否启用(将使用的参数68、69的版本为存储在对应于目前例外层级E1x的GCS控制寄存器66的防护版本的一个中的版本)。再者,处理电路系统4确定例外是否要在与采取该例外之前的目前例外层级相同的例外层级(E1x)中采取。当启用GCS模式且启用例外返回状态的锁定,并且该例外是在与采取该例外之前的目前例外层级相同的例外层级中采取时,则在步骤128,处理电路系统4将例外返回状态锁定参数EXLOCK 58设定成锁定状态。否则,在步骤129处,EXLOCK 58设定为未锁定状态。虽然图9显示仅针对在相同的例外层级EL中采取的例外将EXLOCK 58设定到锁定状态的示例,但其它示例也可针对在更高权限的例外层级采取的例外将EXLOCK 58设定到锁定状态。

[0108] 图10为显示例外返回状态推送程序的流程图,以将保护的例外返回状态推送到GCS数据结构80。在步骤130,指令解码器10解码GCS例外返回状态推送指令,同时GCS启用参数68在启用状态中,指示启用GCS模式(如果已停用GCS模式,则GCS例外返回状态推送指令可被解译为不操作指令或未定义指令,并且不执行图10的剩余步骤)。在步骤132,处理电路系统4是通过指令解码器10来控制,以检查EXLOCKEN 69是否指示例外返回状态的锁定启用(如果GCS控制寄存器66的防护版本是提供用于不同的例外层级,在步骤132处用于检查的参数69的版本为对应于目前例外层级的版本)。如果如此,则在步骤134,控制处理电路系统4以检查例外返回状态锁定参数(EXLOCK 58)是否目前在未锁定状态中。如果EXLOCK 58在未锁定状态中,则在步骤136用信号发送错误,因为对于例外返回状态锁定启用时的有效的例外进入序列,如图7所示,当例外返回状态推送到GCS数据结构80时,例外返回状态锁定参数58应目前被锁定(对于在更高权限的例外层级采取的例外,其中在采取例外时EXLOCK 58并未锁定,如图9所述,根本不需要执行GCS例外返回状态推送指令,因为更高权限的例外处理器可受信任以管理例外返回状态,因此,对于那些更高权限的例外处理器,当执行GCS例外返回状态推送指令时EXLOCK 58要被锁定的要求并不导致问题)。如果EXLOCK 58目前在未锁定状态用信号发送错误可提供反制措施对抗攻击,这些攻击基于篡改程序流程以导致GCS推送程序在紧随例外进入之后以外的情况下被执行。因此,在步骤134处的检查降低攻击者控制任意的例外返回状态(其不是响应于例外进入所检索的例外返回状态)推送GCS数据结构80的风险。

[0109] 如果在步骤132发现例外返回状态锁定被停用(EXLOCKEN 69设定为停用状态)或在步骤134发现EXLOCK 58处于锁定状态中,则在步骤138,处理电路系统4从例外返回状态寄存器60、62获得例外返回状态信息,并且在步骤140,处理电路系统4发出一个或多个存储器存取请求,以推送获得的例外返回状态信息到GCS数据结构80上的位置,该位置是基于存

储在对应于目前例外层级E1x的GCS指标寄存器GCSPR_E1x 64中的目前堆叠指标来确定。

[0110] 在步骤140,处理电路系统4检查EXLOCKEN 69是否指示例外返回状态的锁定经启用,并且如果例外返回状态的锁定经启用,则前进到步骤144(或替代地,在步骤132处执行的相同检查也可使步骤144有条件地执行,取决于例外返回状态的锁定经启用或停用,因此可不需要再次检查EXLOCKEN参数69)。如果例外返回状态的锁定经启用,则在步骤144,例外返回状态锁定参数58设定为未锁定状态(如果例外返回状态的锁定经停用,则跳过此步骤)。无论例外返回状态的锁定是否经启用或停用,在步骤146,处理电路系统4更新对应于目前执行状态(例如,例外层级E1x的GCS指标寄存器64中的堆叠指标,以考虑到该堆叠的例外返回状态的推送(例如,堆叠指标可递减或递增一数量,该数量对应于基于寄存器60、62的内容所形成的例外返回状态记录的大小)。

[0111] 图11为展示用于从GCS数据结构80取出例外返回状态所执行的步骤的流程图。在步骤150,指令解码器10解码GCS例外返回状态取出指令,同时GCS模式由GCS启用参数68指示为启用(再次,如果停用GCS模式,则GCS例外返回状态取出指令可处理为NOP(不操作指令)或未定义指令,并且可不执行图11的剩余步骤)。

[0112] 响应于GCS例外返回状态取出指令,在步骤152,处理电路系统4检查EXLOCKEN 69是否设定到指示例外返回状态的锁定启用的启用状态(如果GCS控制寄存器66的防护版本是提供用于不同的例外层级,在步骤152处用于检查的参数69的版本为对应于目前例外层级的版本)。如果启用例外返回状态的锁定,则在步骤154,处理电路系统4检查EXLOCK参数58是否设定为锁定状态。如果EXLOCK 58目前在锁定状态中,则在步骤156用信号发送错误,因为此指示从期望的使用偏离,其中例外返回状态取出指令不应在EXLOCK锁定时的时间执行。此错误可用信号发送可能的攻击,其尝试损坏程序流程,以导致来自GCS数据结构80的保护的例外返回状态的不正确验证。

[0113] 如果在步骤152处确定例外返回状态的锁定被启用或在步骤154处确定EXLOCK 58目前在未锁定状态中,则在步骤158处理电路系统4发出一个或多个存储器存取请求,以从GCS数据结构80上的一位置取出GCS保护的例外返回信息,该位置是基于对应于目前例外层级E1x的GCS指标寄存器66中的堆叠指标而确定。在步骤160,寄存器GCSPR_E1x 66中的GCS堆叠指标被更新以考虑从GCS数据结构80的例外返回状态的取出(例如,堆叠指标在相反方向上递增或递减(但通过对应的数量)到图10的步骤146所做的更新)。

[0114] 步骤162、164及166为可响应于GCS例外返回状态取出指令而选择性执行的步骤,在一实现方案中,在步骤158处,GCS例外返回状态取出指令还控制处理电路系统4验证目前存储在例外返回寄存器60、62中的预期的例外返回状态信息是否匹配从GCS数据结构80取出的GCS保护的例外返回状态信息。在步骤162,处理电路系统4读取对应于目前例外层级E1x的例外连结寄存器62及SPSR 60中的目前值所表示的预期的例外返回状态信息。用语“预期的”例外返回状态信息用以指在执行GCS例外返回状态取出指令时在寄存器62、60中的例外返回状态的目前值(其可或可不同于对应的例外进入发生时写入到这些寄存器的值)。在步骤164,处理电路系统4确定从寄存器60、62获得的预期的例外返回状态信息是否匹配从存储器的受保护GCS区域中存储的GCS数据结构80读取的GCS保护的例外返回状态信息,并且如果不匹配,则在步骤166用信号发送错误。

[0115] 如果在步骤164处预期的例外返回状态信息匹配GCS保护的例外返回状态信息,或

如果步骤162、164、166未实施,因为GCS例外返回状态取出指令未触发验证(但例如反而直接取出GCS保护的返回状态信息到例外返回状态寄存器62、60),则在步骤168,处理电路系统4检查EXLOCKEN 69是否指示例外返回状态的锁定被启用,并且如果例外返回状态的锁定被启用,前进到步骤170(或替代地,在步骤152处执行的相同检查还可导致步骤170有条件地被执行,取决于例外返回状态的锁定是否被启用或停用)。如果例外返回状态的锁定经启用,则在步骤170,例外返回状态锁定参数58设定为锁定状态(如果例外返回状态的锁定经停用,则跳过此步骤)。

[0116] 图12为示出响应于例外返回指令而执行的步骤的流程图。在步骤180处,指令解码器10解码例外返回指令,并且响应地产生用于控制处理电路系统4(特定来说,执行级16)的控制信号,以执行后续步骤。在步骤182处,处理电路系统4确定GCS启用参数68是否指示GCS模式启用,EXLOCKEN参数69是否指示例外返回状态的锁定也启用(再次,如果GCS控制寄存器66的防护版本是提供用于不同的例外层级,在步骤182处用于检查的参数68、69的版本为对应于目前例外层级的版本),及例外返回是否是要在与目前例外层级相同的例外层级中处理的返回。如果GCS模式及例外返回状态的锁定两者都经启用,并且例外返回导致返回在目前例外层级中执行处理,则在步骤184处,处理电路系统4检查EXLOCK参数58是否目前在未锁定状态中。如果EXLOCK 58目前在未锁定状态中,则在步骤186处用信号发送错误,因为此指示例外返回可能未遵循先前的例外返回状态取出指令82,并且因此,使用在寄存器60、62中指定的可能不安全的例外返回状态来执行例外返回可能不安全。

[0117] 如果GCS模式目前停用,或EXLOCKEN参数69被设定成停用例外返回状态的锁定,或例外返回是到比目前例外层级更低权限的例外层级,或EXLOCK参数58目前在锁定状态中,则例外返回可进行,接着在步骤188中,将例外返回地址从ELR_E1x 62复制到程序计数器寄存器52,并且将存储的处理器状态从SPSR_E1x 60复制到PSTATE寄存器54。应注意,因为EXLOCK参数58是在例外时存储的处理器状态的部分并且在例外返回时恢复,步骤188可改变例外返回状态锁定参数58到采取对应的例外的时间时由EXLOCK 58指示的相同状态。因此,在例外返回之后EXLOCK 58是在锁定状态或未锁定状态中将取决于采取对应的例外时的EXLOCK的状态。

[0118] 图13为流程图,展示用于控制是否可允许寄存器60、62中的例外返回状态信息的修改的步骤。在步骤200,指令解码器10解码请求与目标例外层级相关联的例外连结寄存器62或存储的处理器状态寄存器60中的一个或两个中的例外返回状态信息(例如,例外返回地址或存储的处理器状态)的修改的指令(该目标例外层级可或可不为目前例外层级56,因为可能是在较高权限的例外层级中执行的指令要修改与较低权限的例外层级相关联的例外返回状态)。在步骤202,处理电路系统4确定GCS模式是否由启用参数68指示为启用且EXLOCKEN参数69指示停用例外返回状态的锁定。如果GCS模式停用或例外返回状态的锁定停用,则在步骤204处,假设满足任何其它检查(例如,检查目前例外层级是否具有足够的权限以允许修改指定的寄存器),可允许对例外返回状态信息的修改。如果在步骤202处,确定GCS模式及例外返回状态的锁定都经启用,则在步骤206处,处理电路系统4确定例外返回状态锁定参数EXLOCK 58是否目前在锁定状态中。如果EXLOCK 58在未锁定状态中,则在步骤204,再次,假设满足任何其它检查,则可允许修改。如果在步骤206处,EXLOCK参数58在锁定状态中,则在步骤208处用信号发送错误,以防止例外返回状态信息的修改(且因此监督免

于尝试在如图7所示的窗A 90或窗B 92中篡改例外返回状态)。

[0119] 在图13的示例中,步骤208处的用信号发送错误无关于待写入到寄存器60、62的一个中的例外返回状态信息的值是否与存储在该寄存器中的先前值相同或不同。因此,即使在步骤200所遇到的指令的执行事实上将不会导致例外返回状态的变化,将用信号发送错误。此方法可更简单地以硬件实施,因为其不需要提供电路逻辑以使用信号发送错误取决于寄存器60、62的读取值及与待写入的新值进行比较。

[0120] 替代方案展示于图14中。图14的步骤200、202、204、206及208与图13中的对应步骤相同。图14包括步骤210处的额外检查,其检查待写入到寄存器60、62中的一个的新值是否不同于该寄存器中的例外返回状态信息的旧值。如果新值与旧值不同,在步骤208处用信号发送错误。如果新值与旧值相同,则在步骤204,尝试的“修改”可被允许,因为其不会改变例外返回状态信息。此方法可加速一些程序码路径,例如快速例外路径,其使用非常短的例外处理例程(以在采取例外之后几乎立即地从例外返回),并且因此不需要存储/恢复例外返回状态到存储器,减少受攻击者篡改的风险。对于此类快速例外程序码路径,可不使用GCS例外返回状态推送/取出指令,并且因此EXLOCK可在整个例外例程中保持设定。然而,如果该例外处理例程执行尝试修改寄存器60、62中的例外返回状态的系统寄存器修改指令(不太可能,但有机会),则在系统寄存器修改指令所写入的新值实际上相同于旧值而使得不存在损坏例外返回状态的风险的情况下,可避免EXLOCK锁定时将触发的对应错误。此可帮助改善此类程序码路径的性能。

[0121] 虽然上文所论述的流程图展示以某些依序顺序执行的步骤,但应了解,相同功能也可以不同顺序、或至少部分并行地执行。

[0122] 图15示出可使用的模拟器实施方案。虽然稍早所述的实施方案以用于操作支持所关注技术的特定处理硬件的设备及方法来实施本发明,但也可能根据本文所述的实施方案提供指令执行环境,其是通过使用计算机程序实施。此类计算机程序常称为模拟器,因为其提供硬件架构的基于软件的实施方案。模拟器计算机程序的种类包括仿真器、虚拟机、模型、以及二进制转换器(包括动态二进制转换器)。一般来说,模拟器实施方案可在可选地运行主机操作系统320、支持模拟器程序310的主机处理器330上运行。在一些布置中,在硬件与所提供的指令执行环境和/或相同的主机处理器上提供的多个相异指令执行环境之间可有多层模拟。历史上,已需要强大的处理器来提供模拟器实施方案,其以合理速度执行,但此种方法在某些情况下可为有正当理由的,例如当因为兼容性或再使用原因因此需要执行另一处理器原生的程序码时。例如,模拟器实施方案可提供具有不为主机处理器硬件所支持的额外功能性的指令执行环境,或提供一般与不同的硬件架构相关联的指令执行环境。模拟的综述于“一些高效的架构模拟技术(Some Efficient Architecture Simulation Techniques)”中给出,Robert Bedichek, Winter 1990USENIX Conference, 页数53到63。

[0123] 在先前已参照特定硬件架构或特征来描述实施方案的情况下,在模拟实施方案中,可通过合适的软件架构或特征提供等效功能。例如,可在模拟实施方案中将特定电路系统实施为计算机程序逻辑。类似地,存储器硬件(例如寄存器或高速缓冲存储器)可在模拟实施方案中实施为软件数据结构。在先前描述的实施方案中提及的硬件元件的一个或多个存在于主机硬件(例如,主机处理器330)上的布置中,一些模拟实施方案可(在适当处)利用主机硬件。

[0124] 模拟器程序310可存储在计算机可读存储介质(其可为非暂时性介质)上,并提供程序接口(指令执行环境)给目标代码300(其可包括应用程序、操作系统、以及虚拟机监控程序),该程序接口与通过模拟器程序310建模的硬件架构的程序接口相同。因此,目标代码300的包括如上文描述的GCS存取指令的程序指令可在指令执行环境内使用模拟器程序310执行,使得实际上不具有上文讨论的设备2的硬件特征的主机计算机330可模仿此些特征。类似地,基于EXLOCK参数58的检查可使用模拟器程序310仿真(例如,使用处理程序逻辑312或指令解码程序逻辑314的部分)。

[0125] 因此,模拟器程序310可具有处理程序逻辑312,其模拟上述的处理电路系统4的状态。例如,处理程序逻辑312可响应于目标代码300的模拟执行期间发生的事件而控制执行状态EL0-EL3的转变。指令解码程序逻辑314解码目标代码300的指令及将这些指令映射到主机设备330的原生指令集中的对应指令集。寄存器仿真程序逻辑316将由目标代码请求的寄存器存取映射成对维护在主机设备330的主机硬件上的对应数据结构的存取,例如通过存取主机设备330的寄存器或存储器332中的数据。存储器管理程序逻辑318以与上述硬件实施的实施方案中描述的MMU 28对应的方式实施地址转换、分页表走访及存取控制检查,但还具有映射模拟的物理地址(其通过基于针对目标代码300所限定的分页表在地址转换中的阶段2映射而获得)到用于存取主机存储器332的主机虚拟地址的额外功能。此些主机虚拟地址可本身使用由主机所支持的标准地址转换机制来转换成主机物理地址(将主机虚拟地址转换成主机物理地址是在受到模拟器程序310控制的范围之外)。例外处理程序逻辑320控制在目标代码300执行期间产生的模拟例外的处理(包括由于上文所描述的EXLOCK型检查所触发的错误)。

[0126] 在本申请中,用语“被配置为……”用以意指设备的元件具有能够实行该经定义作业的配置。在此上下文中,“配置”意指硬件或软件的互连的布置或方式。例如,该设备可具有专用硬件,其提供经定义的作业,或者处理器或其它处理装置可被编程以执行该功能。“被配置为”并不意味着设备元件需要以任何方式改变以提供所定义的作业。

[0127] 虽然本文已参照附图详细地描述本发明的说明性实施方案,但应了解,本发明不限于这些精确实施方案,并且所属领域的技术人员可于其中实行各种变化与修改,而不脱离如随附权利要求书所定义的本发明的范围。

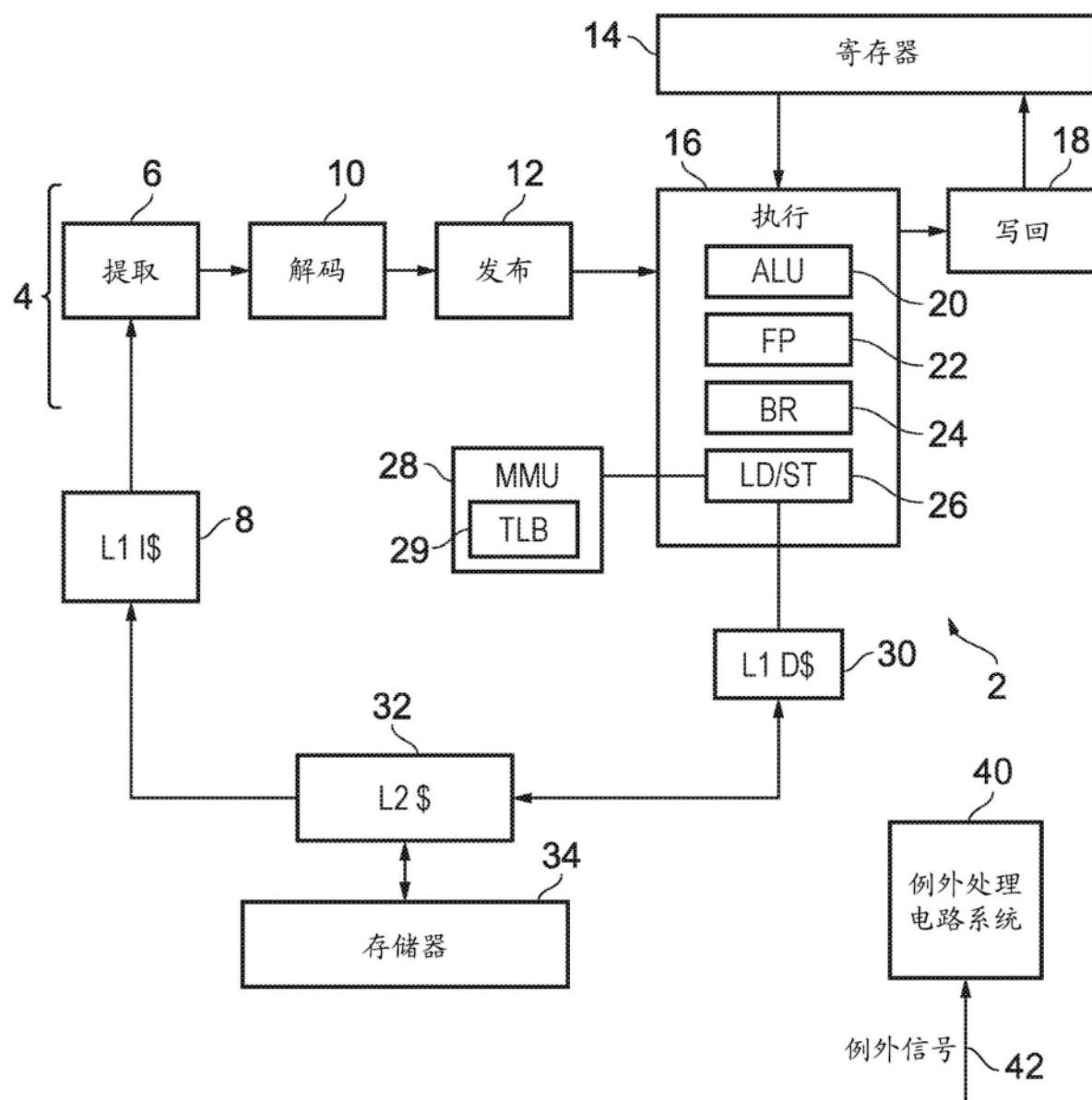


图1

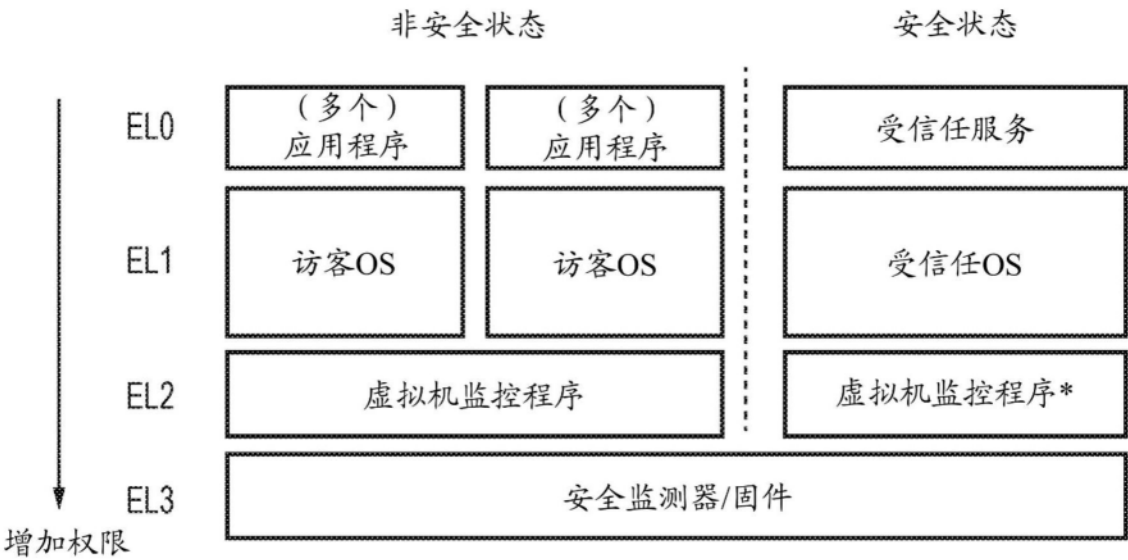


图2

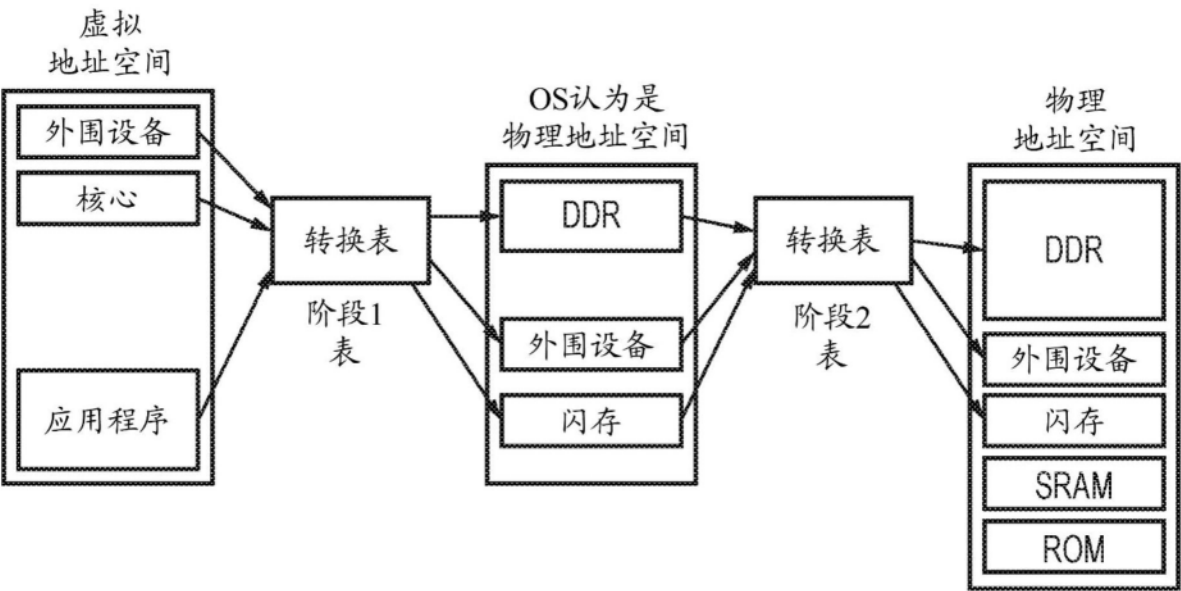


图3

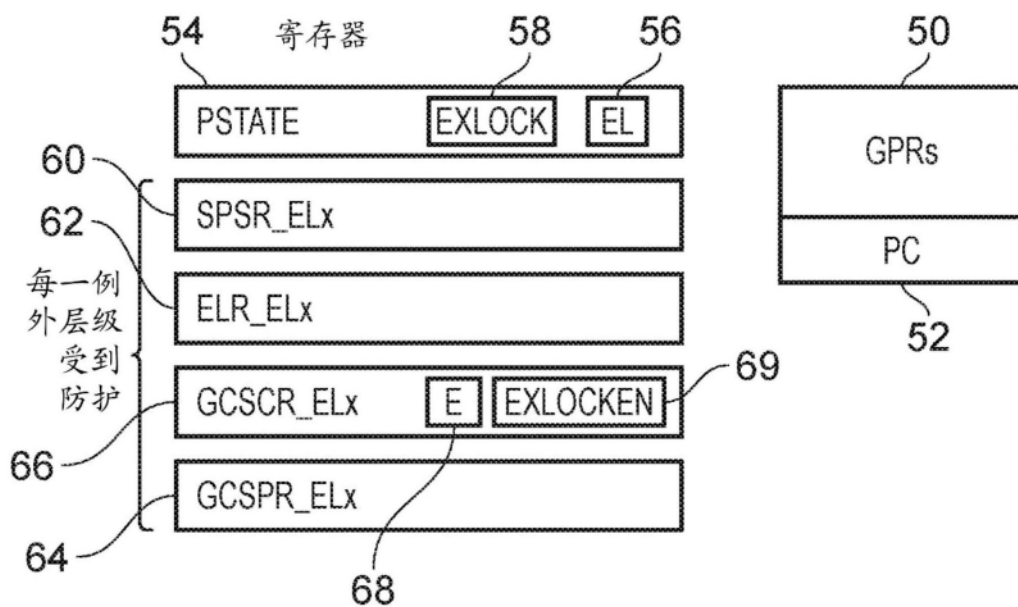


图4

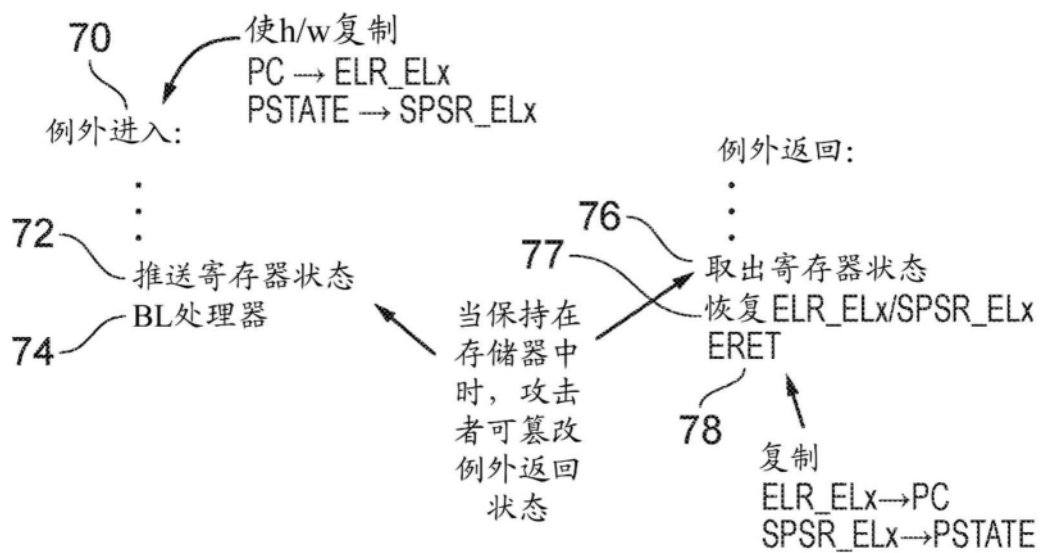


图5

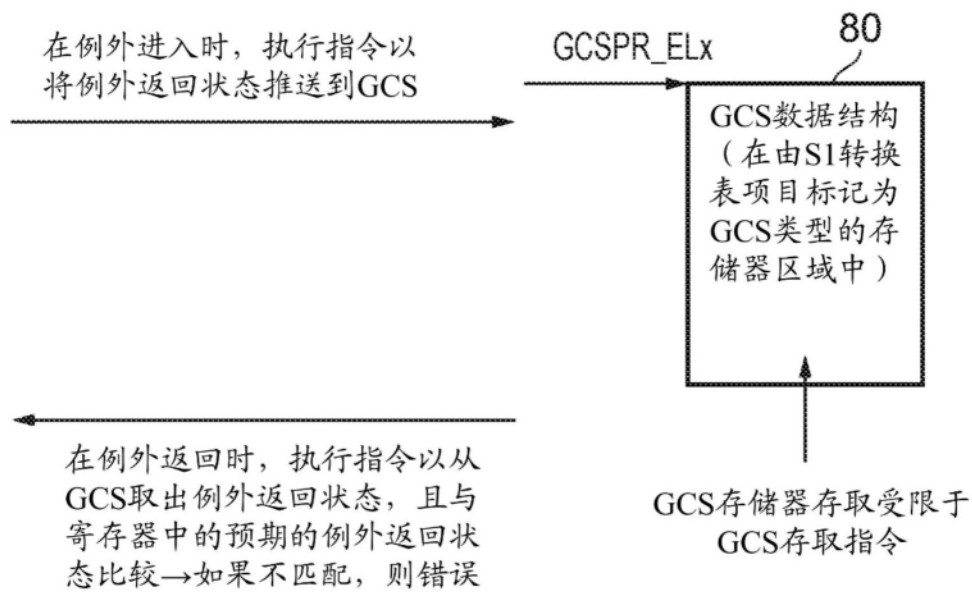


图6

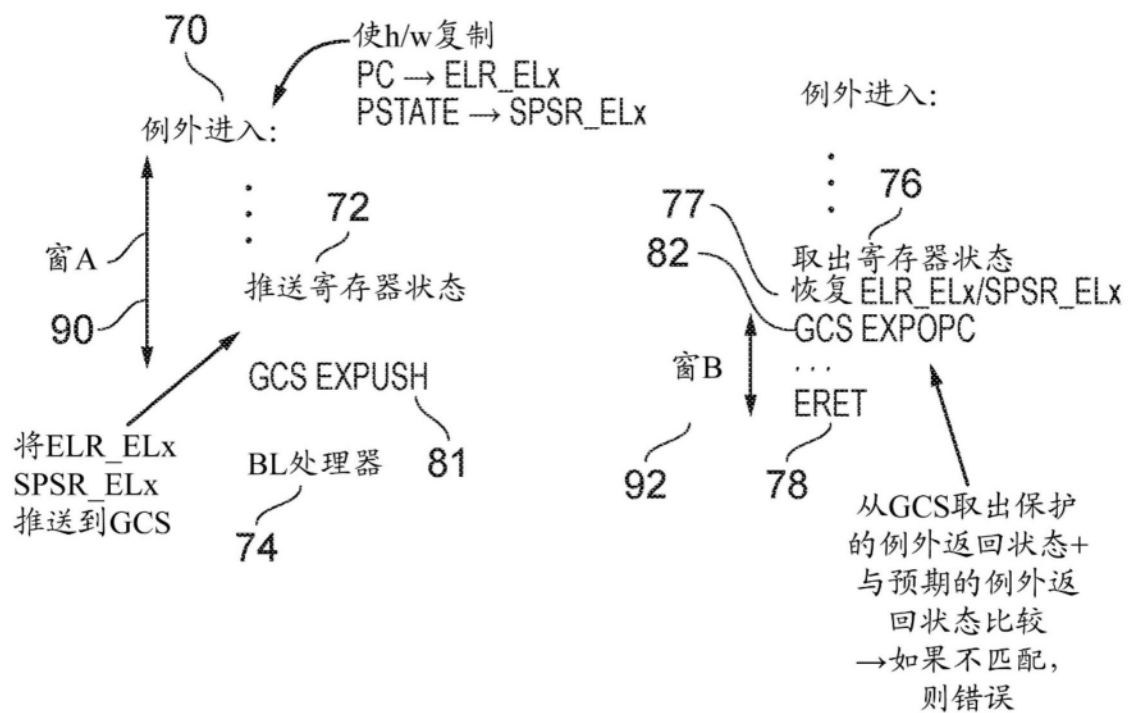


图7

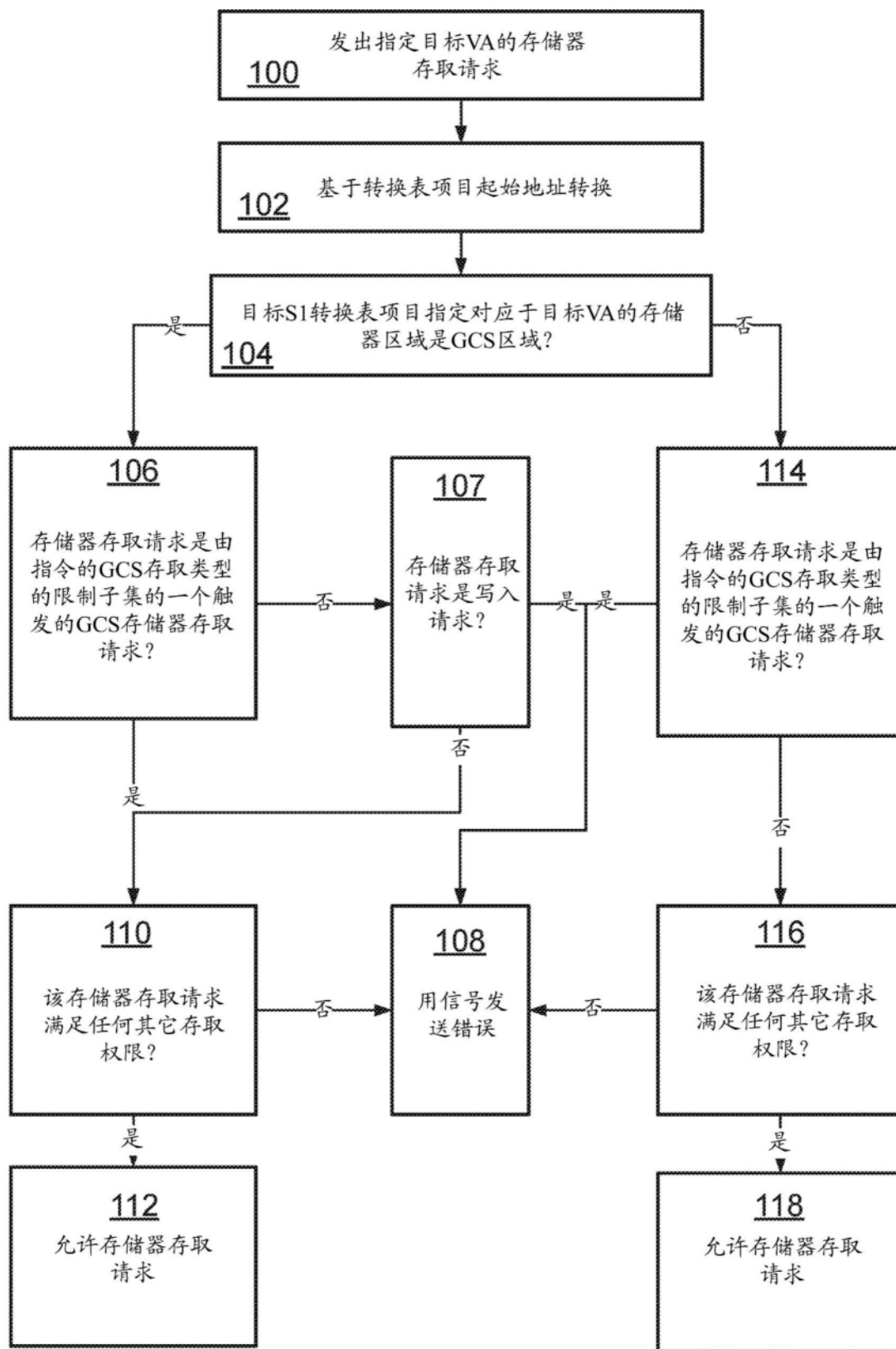


图8

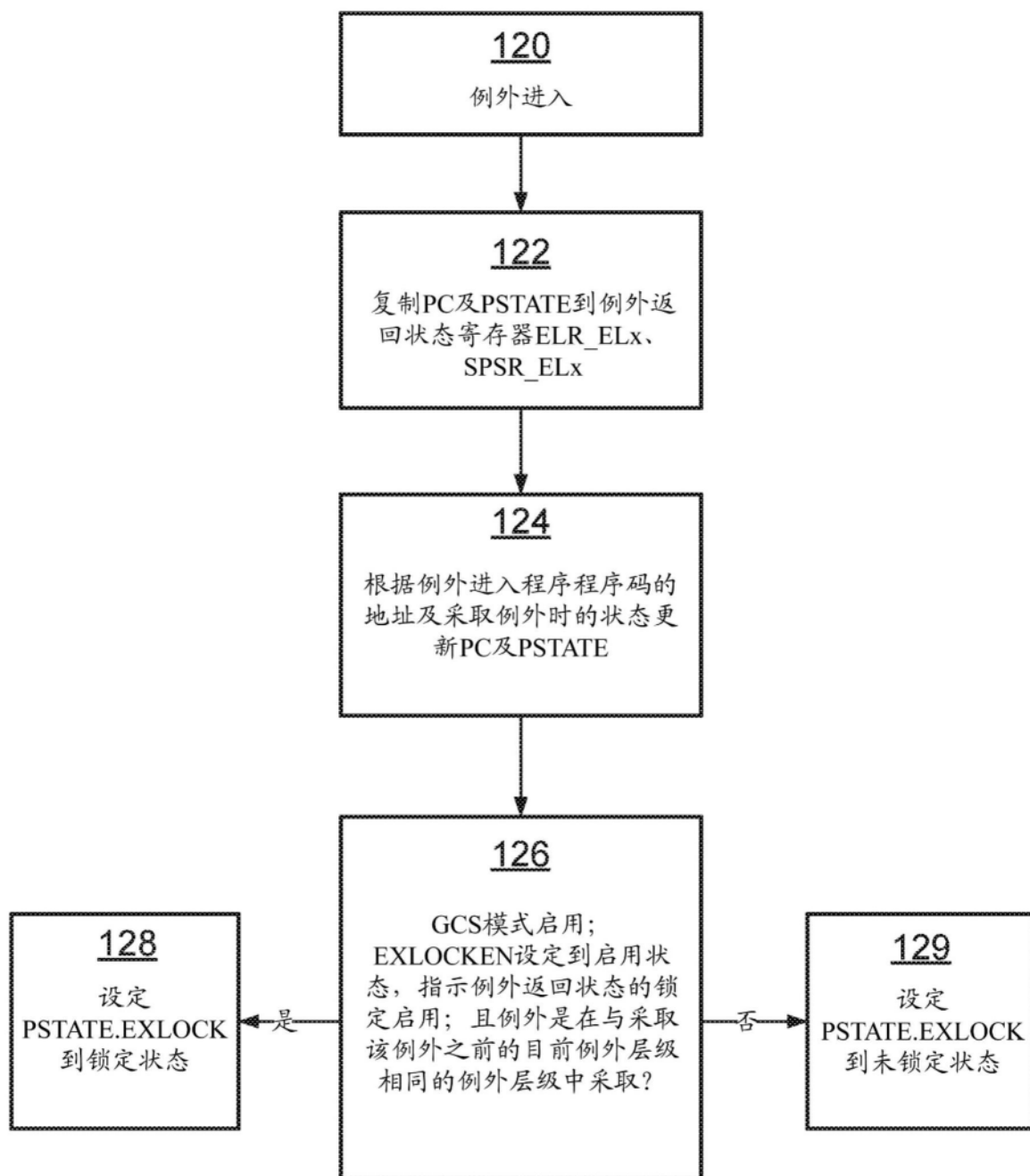


图9

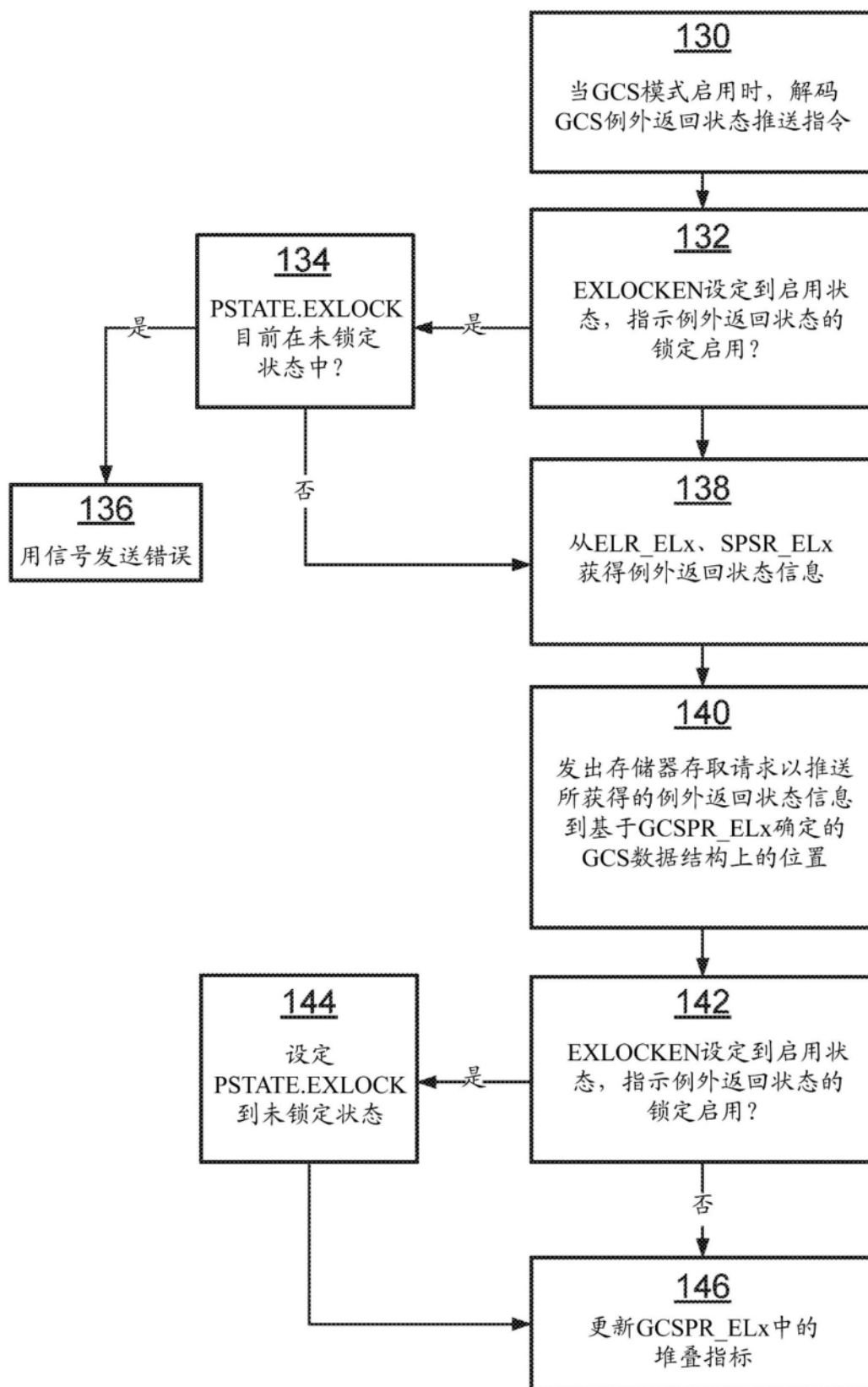


图10

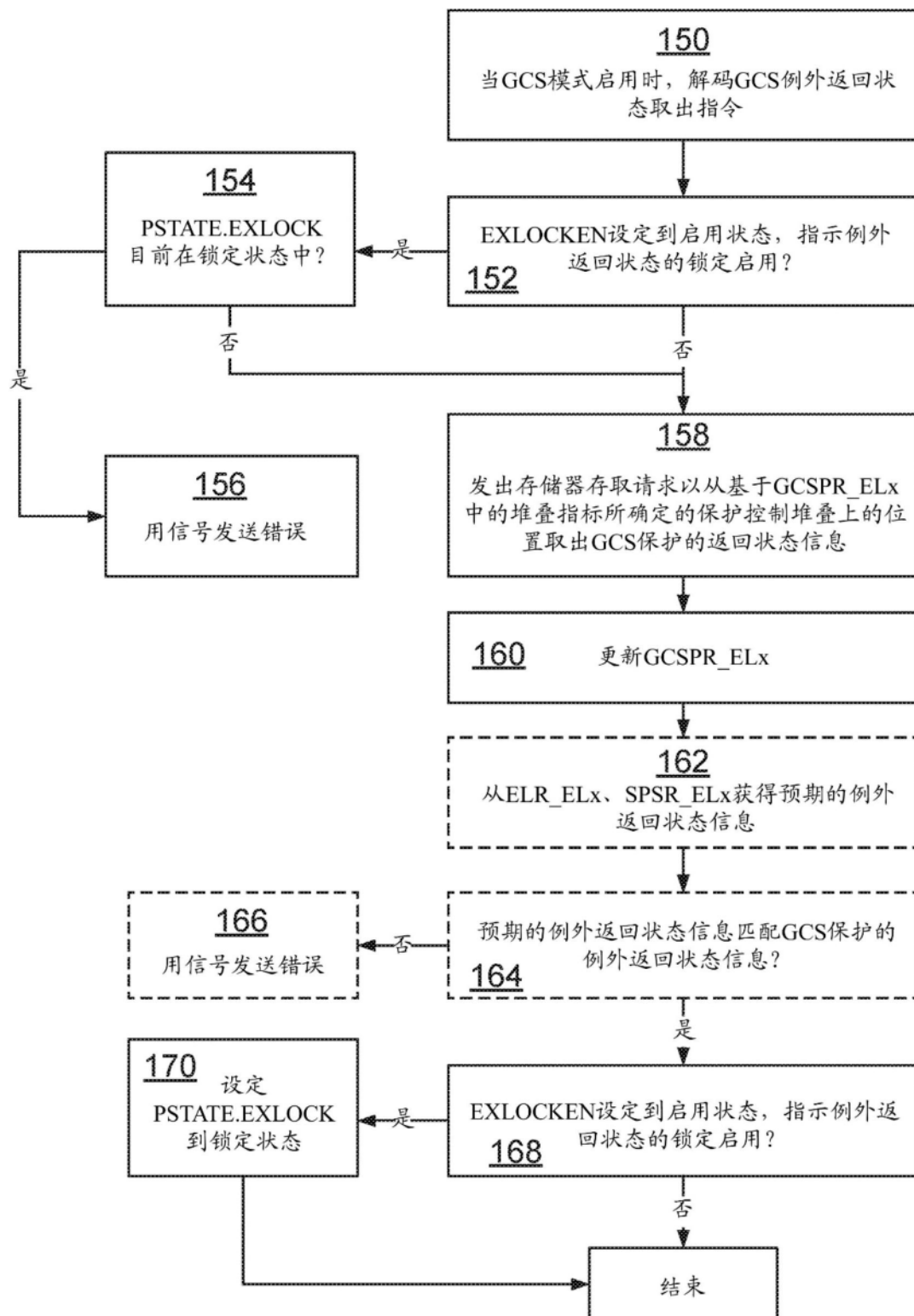


图11

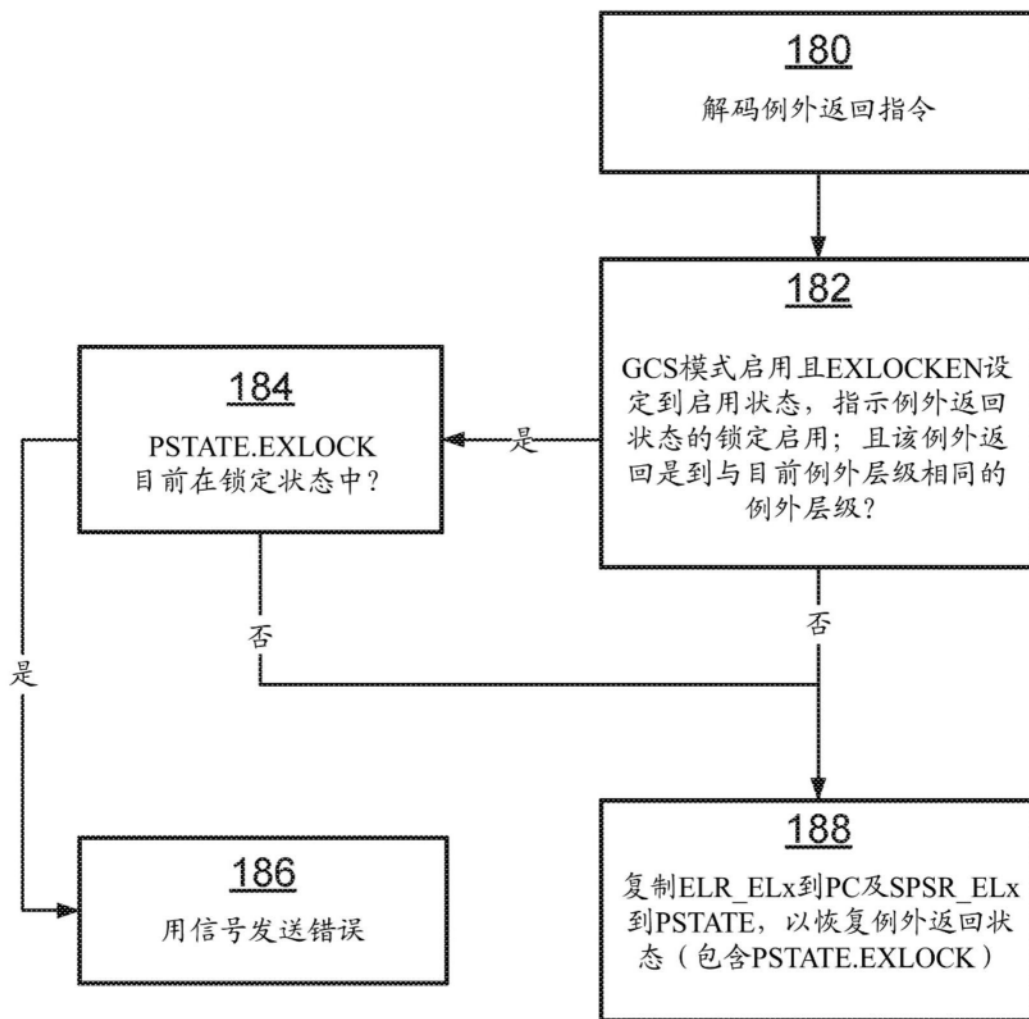


图12

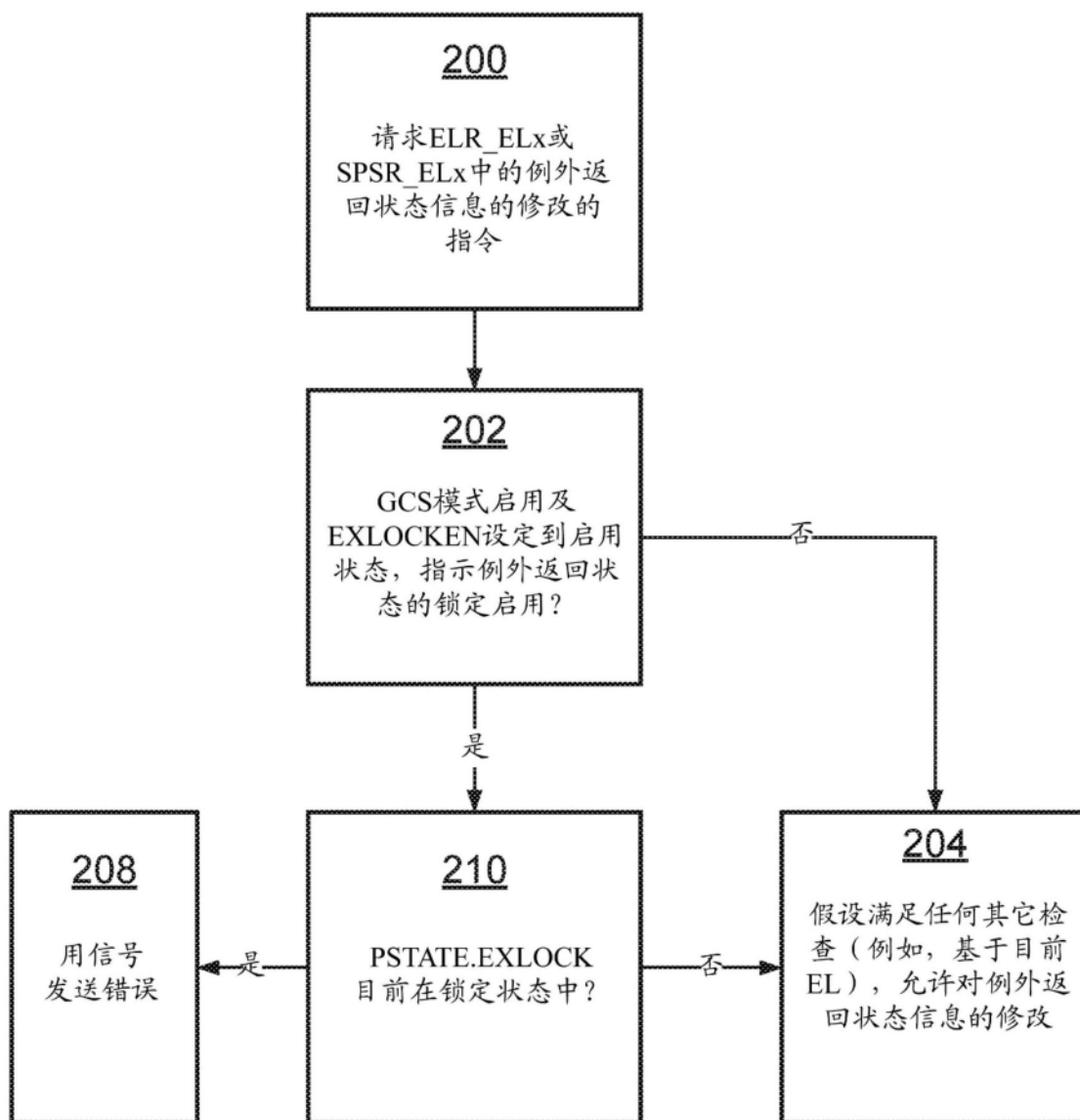


图13

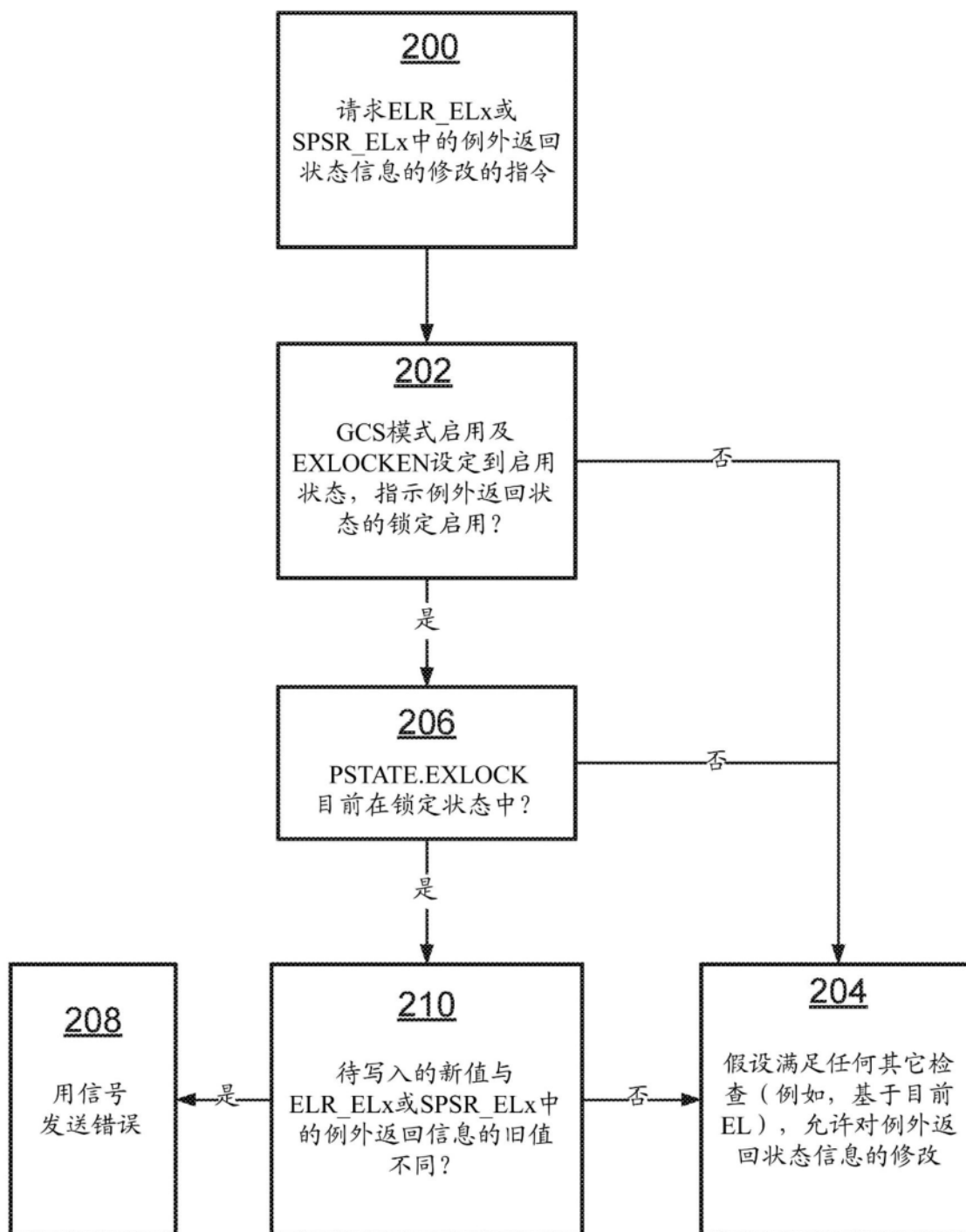


图14

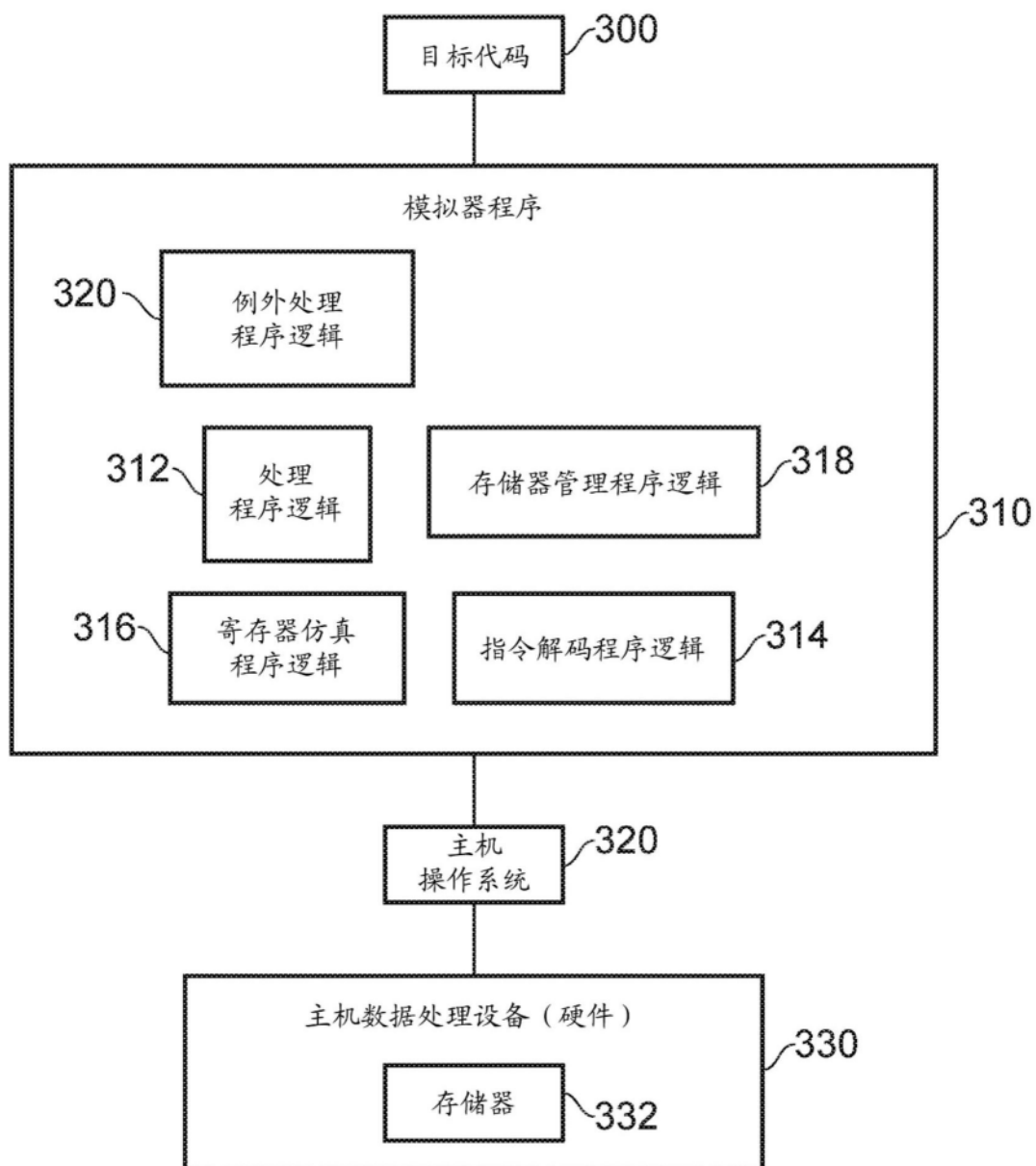


图15