



(12)发明专利申请

(10)申请公布号 CN 109564608 A

(43)申请公布日 2019.04.02

(21)申请号 201780046239.4

(22)申请日 2017.06.30

(30)优先权数据

62/368,223 2016.07.29 US

15/245,041 2016.08.23 US

(85)PCT国际申请进入国家阶段日

2019.01.25

(86)PCT国际申请的申请数据

PCT/US2017/040495 2017.06.30

(87)PCT国际申请的公布数据

W02018/022256 EN 2018.02.01

(71)申请人 高通股份有限公司

地址 美国加利福尼亚

(72)发明人 S·K·德

(74)专利代理机构 永新专利商标代理有限公司

72002

代理人 张立达 王英

(51)Int.Cl.

G06F 21/53(2006.01)

G06F 21/56(2006.01)

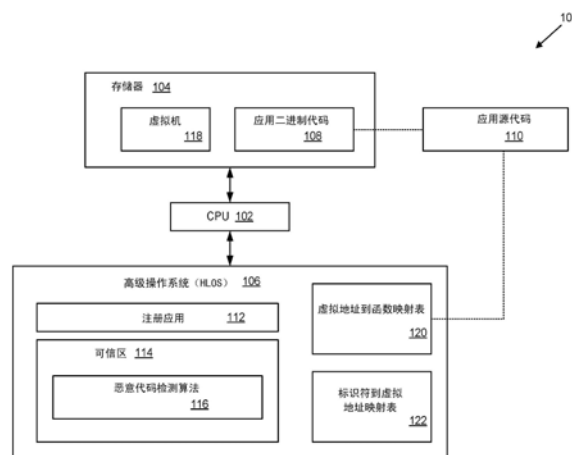
权利要求书3页 说明书16页 附图16页

(54)发明名称

对用于应用二进制代码的经更新的版本的目标应用功能的虚拟存储器地址进行更新

(57)摘要

公开了用于对用于应用二进制代码的经更新的版本的目标应用功能的虚拟存储器地址进行更新的系统、方法以及计算机程序。所述方法包括：存储与向高级操作系统注册的应用二进制代码相关联的虚拟地址映射表。所述虚拟地址映射表包括在所述应用二进制代码中被映射到对应的目标应用功能的多个虚拟地址。响应于接收到所述应用二进制代码的经更新的版本，选择伪二进制代码模板，其与所述虚拟地址映射表中的所述多个虚拟地址中的一个或多个虚拟地址相关联。将所述伪二进制代码模板与所述应用二进制代码的所述经更新的版本中的二进制指令进行匹配。确定与所匹配的二进制指令相对应的所述新虚拟地址。利用所述新虚拟地址来更新所述虚拟地址映射表。



1. 一种用于对针对应用二进制代码的经更新的版本的目标应用功能的虚拟存储器地址进行更新的方法,所述方法包括:

存储与向高级操作系统注册的应用二进制代码相关联的虚拟地址映射表,所述虚拟地址映射表包括在所述应用二进制代码中被映射到对应的目标应用功能的多个虚拟地址;

响应于接收到所述应用二进制代码的经更新的版本,选择与所述虚拟地址映射表中的所述多个虚拟地址中的一个或多个虚拟地址相关联的伪二进制代码模板;

将所述伪二进制代码模板与所述应用二进制代码的所述经更新的版本中的二进制指令进行匹配;

确定与匹配的二进制指令相对应的新虚拟地址;以及

利用所述新虚拟地址来更新所述虚拟地址映射表。

2. 根据权利要求1所述的方法,其中,所述将所述伪二进制代码模板与所述应用二进制代码的所述经更新的版本中的所述二进制指令进行匹配包括:对所述应用二进制代码的所述经更新的版本中的关注区域进行搜索。

3. 根据权利要求2所述的方法,其中,所述关注区域是由与用于所述目标应用功能的所述虚拟地址相关联的基础/偏移元数据定义的。

4. 根据权利要求3所述的方法,其中,所述基础/偏移元数据被存储在所述虚拟地址映射表中。

5. 根据权利要求1所述的方法,其中,所述虚拟地址映射表包括:与用于所述目标应用功能的所述多个虚拟地址中的所述一个或多个虚拟地址相关联的基础/偏移元数据。

6. 根据权利要求5所述的方法,其中,所述基础/偏移元数据定义了所述应用二进制代码的所述经更新的版本中的关注区域,以针对匹配的二进制指令进行搜索。

7. 根据权利要求1所述的方法,其中,所述伪二进制代码模板包括伪代码指令,所述伪代码指令等效于与所述目标应用功能中的一个或多个目标应用功能相关的二进制代码指令。

8. 一种用于对针对应用二进制代码的经更新的版本的目标应用功能的虚拟存储器地址进行更新的系统,所述系统包括:

用于存储与向高级操作系统注册的应用二进制代码相关联的虚拟地址映射表的单元,所述虚拟地址映射表包括在所述应用二进制代码中被映射到对应的目标应用功能的多个虚拟地址;

用于响应于接收到所述应用二进制代码的经更新的版本,选择与所述虚拟地址映射表中的所述多个虚拟地址中的一个或多个虚拟地址相关联的伪二进制代码模板的单元;以及

用于将所述伪二进制代码模板与所述应用二进制代码的所述经更新的版本中的二进制指令进行匹配的单元;

用于确定与匹配的二进制指令相对应的新虚拟地址的单元;以及

用于利用所述新虚拟地址来更新所述虚拟地址映射表的单元。

9. 根据权利要求8所述的系统,其中,所述用于将所述伪二进制代码模板与所述应用二进制代码的所述经更新的版本中的所述二进制指令进行匹配的单元包括:用于对所述应用二进制代码的所述经更新的版本中的关注区域进行搜索的单元。

10. 根据权利要求9所述的系统,其中,所述关注区域是由与用于所述目标应用功能的

所述虚拟地址相关联的基础/偏移元数据定义的。

11. 根据权利要求10所述的系统, 其中, 所述基础/偏移元数据被存储在所述虚拟地址映射表中。

12. 根据权利要求8所述的系统, 其中, 所述虚拟地址映射表包括: 与用于所述目标应用功能的所述多个虚拟地址中的所述一个或多个虚拟地址相关联的基础/偏移元数据。

13. 根据权利要求12所述的系统, 其中, 所述基础/偏移元数据定义了所述应用二进制代码的所述经更新的版本中的关注区域, 以针对匹配的二进制指令进行搜索。

14. 根据权利要求8所述的系统, 其中, 所述伪二进制代码模板包括伪代码指令, 所述伪代码指令等效于表示所述目标应用功能中的一个或多个目标应用功能的二进制代码指令。

15. 一种计算机程序, 其包含在非暂时性计算机可读介质中并且由处理器执行用于对针对应用二进制代码的经更新的版本的目标应用功能的虚拟存储器地址进行更新, 所述计算机程序包括被配置为进行以下操作的逻辑单元:

存储与向高级操作系统注册的应用二进制代码相关联的虚拟地址映射表, 所述虚拟地址映射表包括在所述应用二进制代码中被映射到对应的目标应用功能的多个虚拟地址;

响应于接收到所述应用二进制代码的经更新的版本, 选择与所述虚拟地址映射表中的所述多个虚拟地址中的一个或多个虚拟地址相关联的伪二进制代码模板; 以及

将所述伪二进制代码模板与所述应用二进制代码的所述经更新的版本中的二进制指令进行匹配;

确定与匹配的二进制指令相对应的新虚拟地址; 以及

利用所述新虚拟地址来更新所述虚拟地址映射表。

16. 根据权利要求15所述的计算机程序, 其中, 所述被配置为将所述伪二进制代码模板与所述应用二进制代码的所述经更新的版本中的所述二进制指令进行匹配的逻辑单元包括: 被配置为对所述应用二进制代码的所述经更新的版本中的关注区域进行搜索的逻辑单元。

17. 根据权利要求16所述的计算机程序, 其中, 所述关注区域是由与用于所述目标应用功能的所述虚拟地址相关联的基础/偏移元数据定义的。

18. 根据权利要求17所述的计算机程序, 其中, 所述基础/偏移元数据被存储在所述虚拟地址映射表中。

19. 根据权利要求15所述的计算机程序, 其中, 所述虚拟地址映射表包括: 与用于所述目标应用功能的所述多个虚拟地址中的所述一个或多个虚拟地址相关联的基础/偏移元数据。

20. 根据权利要求19所述的计算机程序, 其中, 所述基础/偏移元数据定义了所述应用二进制代码的所述经更新的版本中的关注区域, 以针对匹配的二进制指令进行搜索。

21. 根据权利要求15所述的计算机程序, 其中, 所述伪二进制代码模板包括伪代码指令, 所述伪代码指令等效于表示所述目标应用功能中的一个或多个目标应用功能的二进制代码指令。

22. 一种用于对针对应用二进制代码的经更新的版本的目标应用功能的虚拟存储器地址进行更新的系统, 所述系统包括:

处理设备, 其被配置为执行应用二进制代码;

高级操作系统(HLOS),其包括与所述应用二进制代码相关联的虚拟地址映射表,所述虚拟地址映射表包括在所述应用二进制代码中被映射到对应的目标应用功能的多个虚拟地址;以及

控制器,其用于响应于接收到所述应用二进制代码的经更新的版本来更新所述虚拟地址映射表,所述控制器包括被配置为进行以下操作的逻辑单元:

选择与所述虚拟地址映射表中的所述多个虚拟地址中的一个或多个虚拟地址相关联的伪二进制代码模板;

将所述伪二进制代码模板与所述应用二进制代码的所述经更新的版本中的二进制指令进行匹配;

确定与匹配的二进制指令相对应的新虚拟地址;以及

利用所述新虚拟地址来更新所述虚拟地址映射表。

23.根据权利要求22所述的系统,其中,所述控制器包括通过以下操作将所述伪二进制代码模板与所述应用二进制代码的所述经更新的版本中的所述二进制指令进行匹配:对所述应用二进制代码的所述经更新的版本中的关注区域进行搜索。

24.根据权利要求23所述的系统,其中,所述关注区域是由与用于所述目标应用功能的所述虚拟地址相关联的基础/偏移元数据定义的。

25.根据权利要求24所述的系统,其中,所述基础/偏移元数据被存储在所述虚拟地址映射表中。

26.根据权利要求22所述的系统,其中,所述虚拟地址映射表包括:与用于所述目标应用功能的所述多个虚拟地址中的所述一个或多个虚拟地址相关联的基础/偏移元数据。

27.根据权利要求26所述的系统,其中,所述基础/偏移元数据定义了所述应用二进制代码的所述经更新的版本中的关注区域,以针对匹配的二进制指令进行搜索。

28.根据权利要求22所述的系统,其中,所述伪二进制代码模板包括伪代码指令,所述伪代码指令等效于表示所述目标应用功能中的一个或多个目标应用功能的二进制代码指令。

29.根据权利要求22所述的系统,其中,所述HLOS被配置为:在所述应用二进制代码的所述经更新的版本的执行期间,当与所述目标应用功能相对应的所述新虚拟地址中的一个或多个虚拟地址被执行时,检测所述目标应用功能。

30.根据权利要求22所述的系统,其中,所述应用二进制代码包括安全网页应用和网页浏览器中的一项。

对用于应用二进制代码的经更新的版本的目标应用功能的虚拟存储器地址进行更新

[0001] 相关申请的优先权要求和交叉引用

[0002] 本专利申请依据35 U.S.C. §119(e) 要求享受于2016年7月29日提交的、名称为“KERNEL-BASED DETECTION OF TARGET APPLICATION FUNCTIONALITY USING VIRTUAL ADDRESS MAPPING”的美国临时申请序列号No.62/368,223的优先权,故将其全部内容以引用方式并入本文。

背景技术

[0003] 存在在硬件平台上运行的各种高级应用,该硬件平台并不在系统或者平台层处显示出任何明显的活动,因此并不提供检测应用执行的有用功能和行为信息的时机。

[0004] 常见的例子是高级网页浏览器应用在其在设备上执行期间受安全漏洞(例如,跨站脚本)危害,安全漏洞并不在系统和平台级别处留下任何指示性踪迹。通过探测系统库、平台、SOC硬件或者观察设备级别活动无法确定这样的活动正发生在高级应用上。

[0005] 因此,为了对在设备上运行的各种第三方应用具有更好的平台级别控制,并且为了检测这些正在执行的高级应用的功能和行为活动中的一些,存在对开发如下机制的需求:该机制能够实现将高级应用功能和行为表达并且传送为平台的HLOS或者内核能够理解的形式。这将允许平台对正在执行的应用的行为具有更好的理解,并且允许平台作出决策和采取动作,以处理正在执行的应用的各种不同的情况。

[0006] 作为一个例子,可以使用该信息来作出防止第三方网页浏览器应用上的网页安全漏洞的平台级别决策。其它领域的示例使用是:一旦使用本公开内容中的该机制在HLOS或者内核层检测到应用的特定功能或者行为性质,平台就作出如增加/减小各种SOC部件(DDR、总线、CPU、缓存)的频率或者引入高功率或者低功率模式的决策。通常,利用本公开内容,平台通过检测和识别出正被应用执行的功能,来获得对在设备上执行的各种第三方应用进行各种控制的时机。这允许SOC和平台供应商从平台级别提供用于各种第三方应用的更好的方案,该平台原本对这些第三方应用没有控制。

发明内容

[0007] 公开了用于对用于应用二进制代码的经更新的版本的目标应用功能的虚拟存储器地址进行更新的系统、方法以及计算机程序。所述方法包括:存储与向高级操作系统注册的应用二进制代码相关联的虚拟地址映射表。所述虚拟地址映射表包括在所述应用二进制代码中被映射到对应的目标应用功能的多个虚拟地址。响应于接收到所述应用二进制代码的经更新的版本,选择伪二进制代码模板,其与所述虚拟地址映射表中的所述多个虚拟地址中的一个或多个虚拟地址相关联。将所述伪二进制代码模板与所述应用二进制代码的所述经更新的版本中的二进制指令进行匹配。确定与匹配的二进制指令相对应的新虚拟地址。利用所述新虚拟地址来更新所述虚拟地址映射表。

[0008] 一种系统包括被配置为执行应用二进制代码的处理设备、包括虚拟地址映射表的

高级操作系统 (HLOS) 以及控制器。所述虚拟地址映射表与所述应用二进制代码相关联,并且包括在所述应用二进制代码中被映射到对应的目标应用功能的多个虚拟地址。所述控制器响应于接收到所述应用二进制代码的经更新的版本,来更新所述虚拟地址映射表。所述控制器包括被配置为进行以下操作的逻辑单元:选择与所述虚拟地址映射表中的所述多个虚拟地址中的一个或多个虚拟地址相关联的伪二进制代码模板;将所述伪二进制代码模板与所述应用二进制代码的所述经更新的版本中的二进制指令进行匹配;确定与匹配的二进制指令相对应的新虚拟地址;以及利用所述新虚拟地址来更新所述虚拟地址映射表。

附图说明

[0009] 在各图中,除非另外指出,否则类似的附图标记贯穿各个视图指代类似的部分。对于具有诸如“102A”或者“102B”之类的字母字符名称的附图标记而言,字母字符名称可以对在同一图中的两个类似的部分或者元素进行区分。当旨在附图标记包含在所有图中具有相同的附图标记的所有部分时,可以省略用于附图标记的字母字符名称。

[0010] 图1是用于使用安全存储器中的虚拟地址映射来检测目标应用功能的系统的实施例的框图。

[0011] 图2示出了目标应用功能到对应的应用二进制代码的示例性映射。

[0012] 图3示出了虚拟地址到函数映射表 (VAFMT) 的示例性实施例。

[0013] 图4是示出用于检测图1的系统中的恶意代码活动的方法的实施例的流程图。

[0014] 图5示出了用于动态地识别虚拟机代码空间的边界的VAFMT的另一实施例。

[0015] 图6示出了与VAFMT相结合地使用的标识符到虚拟映射表 (IVAMT) 的实施例。

[0016] 图7示出了与垃圾回收过程相结合地使用的VM代码空间的部分。

[0017] 图8示出了针对图1的虚拟机中的垃圾回收功能的示例性兴趣点以及用于VAFMT中的功能兴趣点的虚拟地址,所述虚拟地址用于在包含虚拟机的应用二进制文件的执行期间检测垃圾回收活动的执行。

[0018] 图9示出了用于虚拟机堆的外部/内部边界的虚拟地址的示例性映射。

[0019] 图10是示出用于在虚拟机实施例中检测图1的系统中的恶意代码活动的方法的实施例的流程图。

[0020] 图11示出了包括用于特定缓冲分配器函数的虚拟地址的VAFMT的实施例,所述特定缓冲分配器函数用于确定被动态分配的缓冲器的虚拟地址,所述被动态分配的缓冲器包含特定数据结构类型的对象以及在该缓冲器中分配的对象成员/字段的值。

[0021] 图12是示出用于响应于接收到应用二进制代码的经更新的版本而自动地更新VAFMT的系统的实施例的组合框图/流程图。

[0022] 图13示出了图12的具有经更新的虚拟地址和元数据的VAFMT。

[0023] 图14示出了图12的VAFMT中的功能兴趣点与伪二进制代码模板的示例性匹配。

[0024] 图15示出了图14的伪二进制代码模板与应用二进制代码的经更新的版本中的经匹配的区域示例性匹配。

[0025] 图16是示出用于响应于接收到应用二进制代码的经更新的版本而更新VAFMT的方法的实施例的流程图。

具体实施方式

[0026] “示例性”一词在本文中用于意指“用作例子、实例或说明”。在本文中被描述为“示例性的”任何方面未必被解释为比其它方面优选或者有优势。

[0027] 在该描述中,术语“应用”还可以包括具有可执行内容的文件,例如:目标代码、脚本、字节代码、标记语言文件和补丁。另外,本文中所提及的“应用”还可以包括本质上不可执行的文件,例如,可能需要打开的文档或者需要访问的其它数据文件。

[0028] 术语“内容”还可以包括具有可执行内容的文件,例如:目标代码、脚本、字节代码、标记语言文件和补丁。另外,本文中所提及的“内容”还可以包括本质上不可执行的文件,例如,可能需要打开的文档或者需要访问的其它数据文件。

[0029] 如该描述中使用的,术语“部件”、“数据库”、“模块”、“系统”等旨在指代计算机相关实体,任一硬件、固件、硬件和软件的组合、软件、或者执行中的软件。例如,部件可以是但不限于在处理器上运行的进程、处理器、对象、可执行文件、执行的线程、程序和/或计算机。通过说明的方式,在计算设备上运行的应用和计算设备都可以是部件。一个或多个部件可以位于进程和/或执行的线程内,并且部件可以被本地化在一个计算机上和/或分布在两个或更多个计算机之间。另外,这些组件可以从具有存储在其上的各种数据结构的各个计算机可读介质执行。部件可以通过本地和/或远程进程的方式进行通信,例如根据具有一个或多个数据分组的信号(例如,来自一个部件的数据,该部件通过信号的方式与本地系统、分布式系统中或/或跨越诸如互联网之类的具有其它系统的网络的另一部件交互)。

[0030] 图1示出了用于从内核或者操作系统(O/S)层检测应用二进制文件的期望或者目标高级功能的系统100的实施例。如图1的实施例所示,系统100包括处理设备(例如,中央处理单元(CPU)102)、存储器104和高级操作系统(HLOS)106。存储器104存储可以由CPU 102执行的一个或多个应用。存储器104可以存储与参考应用源代码110相对应的应用二进制代码108,参考应用源代码110与安装在计算设备上的应用相关联。在这点上,系统100可以在任何期望的计算设备或者系统中实现,例如,包括个人计算机、膝上型计算机、工作站、服务器、或者便携式计算设备(PCD)(例如,蜂窝电话、智能电话、便携式数字助理(PDA)、便携式游戏控制台、导航设备、平板计算机、可穿戴设备(例如,智能手表)、或者其它电池供电的便携式设备)。

[0031] 在一个实施例中,内核或者O/S层包括高级操作系统(HLOS)106。如图1中所示,HLOS 106包括注册应用112的列表、安全存储器(例如,可信区114)以及用于每个注册应用的应用二进制代码108的专门配置的虚拟地址映射表。注册应用112的列表标识在系统100上安装的、已经向HLOS 106注册的以用于安全控制和/或支持的应用。例如,应用(例如,网页应用、浏览器应用等)的应用二进制代码108可以向HLOS 106注册,并且在列表112中进行标识。如本领域中已知的,可信区114包括安全存储器或者区域,其被配置为保证被加载到存储器和/或被执行的代码和/或数据在安全、机密性、完整性等方面受保护。用于注册应用的应用二进制代码108可以具有一个或多个虚拟地址映射表,其由HLOS 106和/或可信区114中的算法用于通过跟踪预定的虚拟地址点的执行来识别期望或者目标高级应用功能。

[0032] 应当理解的是,可以将系统100应用于各种应用域,在这些应用域中,在内核层跟踪和检测高级应用功能是有利的。例如,在一个示例性实施例中,内核可以控制决策,例如,响应于对正在执行的应用的特定功能或者行为性质的检测,增加和/或减小各种片上系统

(SoC) 部件 (例如, 中央处理单元 (CPU)、缓存、双倍数据速率 (DDR) 存储器、一个或多个总线等) 的频率, 或者设置高功率和/或低功率模式以及启用/禁用特定硬件特征。以这种方式, HLOS 106 和内核通过检测和识别出正被应用执行的功能, 而具有实现对在设备上执行的各种第三方应用的各种控制的时机。应当理解的是, 这可以允许 SoC 和平台供应商从平台/HLOS/内核级别提供用于各种第三方应用的改进的方案, 否则该平台可能对这些第三方应用没有控制。

[0033] 在示例性应用域中, 系统 100 可以提供针对网页应用、网页浏览器、JavaScript (Java 脚本) 代码等的恶意攻击或者其它漏洞的实时安全保护。如本领域中已知的, JavaScript 是在许多网站和网页应用中使用的编程语言, 基于 JavaScript 的攻击是针对网络安全的头号威胁之一。随着越来越多的网页活动从台式计算机转移到移动设备, JavaScript 攻击正变成对便携式计算设备的主要威胁。

[0034] 大多数恶意 JavaScript 攻击利用 JavaScript 语言的特性以及网页标准和规范对漏洞的约束。通过恶意 JavaScript 的基于网页的漏洞的常见例子包括以下各项: 跨站脚本 (即, XSS/CSS)、跨站请求伪造 (即, CSRF/XSRF)、偷渡下载、用户意图劫持、点击劫持、分布式拒绝服务 (DDoS)、JavaScript 隐写术以及各种形式的混淆 JavaScript。由于需要高级网页行为和功能知识来尝试检测恶意行为, 所以通常在浏览器软件架构内构建当前网页和 JavaScript 安全方案。

[0035] 然而, HLOS、内核和设备平台内的内置网页安全机制是受限的, 因为基于网页/JavaScript 的漏洞可能没有关于平台活动 (例如, 系统呼叫、设备使用等) 的可见的指示。许多基于网页/JavaScript 的攻击是面向外部的, 并且仅危害用户的在线资产、活动、身份等。换言之, 可能仅在网页浏览器/应用软件内检测到可见的活动模式, 并且因此针对网页漏洞的大多数安全机制几乎总是构建在网页浏览器应用内。

[0036] 在这点上, 系统 100 中的应用二进制代码 108 的示例性实施例可以包括网页应用、浏览器应用或者在其中 HLOS 106 通过跟踪预定的虚拟地址点来检测高级应用功能的其它应用。如图 1 中进一步所示, 系统 100 还可以包括位于可信区 114 中的一个或多个恶意代码检测算法 116。恶意代码检测算法 116 可以接收与虚拟地址点的执行以及在虚拟地址映射表中标识的其相关联的功能含义相关的数据。基于该数据, 算法 116 可以检测例如恶意代码和行为、恶意 JavaScript 代码和执行等, 并且发起用于解决安全威胁或者以其它方式阻碍恶意攻击的合适的方法。在一个实施例中, 当检测到安全威胁时, 系统 100 可以自动地解决威胁或者提示用户进行合适的动作。

[0037] 如图 1 的实施例中所示, 由 HLOS 106 使用的虚拟地址映射表可以包括虚拟地址到函数映射表 120 以及标识符到虚拟地址映射表 122。应当理解的是, HLOS 106 和映射表 120 和 122 包括集成化平台机制, 通过该机制, 系统 100 可以确定来自正在执行的应用二进制代码 108 的期望或者目标高级功能信息。高级功能信息可以由在可信区 114 中实现的算法和/或模型 (例如, 恶意代码检测算法 116) 用于检测恶意行为。

[0038] 如以下更详细地描述的, 系统 100 可以支持用于执行应用二进制代码 108 的两种不同的执行模型。第一执行模型涉及本机二进制执行 (例如, 来自 C/C++ 代码)。第二执行模型涉及托管 (managed) 运行时执行 (例如, 由虚拟机 118 进行的执行)。在一个实施例中, 虚拟机 118 可以执行来自 JavaScript 源的动态即时 (JIT) 或者解释代码。在托管运行时执行实施例

中,虚拟机118可以包括二进制代码108的部分,其中,虚拟机118在该二进制代码108内在该部分中运行。然而,应当理解的是,在其它实施例中,可以存在单独的VM和二进制工作负载。

[0039] 在图2-4中示出了本机二进制执行模型的示例性实施例。对于本机二进制执行而言,注册应用112的列表中的每个应用具有对应的VAFMT 120,其由HLOS 106维护。VAFMT 120可以位于可信区114中。VAFMT 120包括不同的感兴趣的虚拟地址,其与它们相关联的高级功能相映射。在一个实施例中,每个相关联的高级功能可以被表示为算法116理解的宏名称。然而,应当理解的是,可以实现用于表示相关联的高级功能的其它机制,例如包括指向算法116中的函数或者函数名称的指针,以使得在特定虚拟地址处检测到的活动与算法116中需要被触发的功能直接对应。二进制图像中的特定应用函数(以及函数内的特定点)的虚拟地址可以被称为“兴趣点”。在一个实施例中,虚拟地址兴趣点可以包括在例如以下各项之内、起始处、结束处的点或者以下各项之间的多个特定点、或者用于对已知网页/JavaScript攻击的分析和检测的其它合适的信息:敏感源/宿例程、危险网页应用编程接口(API)、特定网页功能、缓冲器的起始/结束、或者攻击者可以利用的任何其它对象。在其它实施例中,虚拟地址兴趣点可以包括在JavaScript解释器、即时(JIT)编译器、或者运行时环境(例如,用于存储JavaScript源代码、字节码/JIT代码的虚拟机堆的分配/解除分配函数等)的实现中的点。

[0040] 图2和3示出了VAFMT 120的示例性实施例。图2示出了应用源代码110内的某些期望或者目标功能点到应用二进制代码108内的对应的虚拟地址点的逻辑映射200。在图2和3中,示出了虚拟地址,但是没有示出二进制目标代码。在该实施例中,应用源代码110包括用于“documentWrite(文档编写)”函数的C++代码。源代码中的点201被映射到二进制代码中的虚拟地址202。源代码中的点203被映射到二进制代码中的虚拟地址204。源代码中的点205被映射到二进制代码中的虚拟地址206。图3示出了在VAFMT 120中的列302之下的二进制代码202、204和206中的虚拟地址到代码在那些虚拟地址处表示的相应的功能含义的逻辑映射300。如图3中所示,VAFMT 120可以包括多个虚拟地址(列302)以及功能兴趣点的对应描述(列304)。由二进制代码点202表示的虚拟地址(0x3273fa94)被映射到与EVAL_FUNCTION相对应的功能点。由二进制代码点204表示的虚拟地址(0x3473fac8)对应于表示DOCUMENT_WRITE_FUNCTION_START的功能兴趣点。由二进制代码中的206表示的虚拟地址(0x3473fad4)被映射到具有宏含义DOCUMENT_WRITE_1的功能点。

[0041] 图11示出了包括自定义虚拟地址表的VAFMT 120的实施例,自定义虚拟地址表具有用于特定缓冲分配器函数的虚拟地址,该缓冲分配器函数可以用于确定动态分配的缓冲器(包括特定数据结构类型的对象(例如,类、结构体、联合体))的起始和结束的虚拟地址。在缓冲器中分配的对象的成员/字段的值可以使用偏移和长度字段来确定,其中,还可以在该表中针对作为兴趣点的特定字段/成员来维护偏移和长度字段。缓冲器分配函数的虚拟地址可以用于通过例如跟踪来自自由分配器函数的虚拟地址覆盖的区域中的系统存储器分配器函数的执行,来检测所分配的缓冲器的大小和地址。一旦已知缓冲器起始和结束虚拟地址,偏移和长度字段就可以用于确定用于特定数据结构类型的对象的特定成员/字段的值。

[0042] 如图1中虚线所示,应用源代码110不需要存储在系统100中。相反,其可以位于离线或者脱离设备的地方,并且可用作参考或者开源代码。用于特定版本的参考源代码可以

用作参考和指导来确定浏览器或者网页应用的实际商业二进制文件中的感兴趣的虚拟地址。可以根据开源项目的相匹配的代码修订版/版本来编译等效二进制文件。经编译的二进制文件可以用于检测基于该版本/修订版的应用二进制文件的期望或者目标虚拟地址以及函数/点。可以使用类似的编译器和链接器选项。此外,在应用代码中的各个点处的断点可以用于对虚拟地址和它们的功能映射点的确定。二进制代码识别和相似度提取方法可以用于通过使用来自用于开源项目的已知的、经编译的函数的参考二进制文件,来识别给定应用二进制文件中的功能。对于具有经稍微修改的版本的二进制文件(或者来源于源代码库(source base)的二进制文件,其具有与已知参考开源项目不同的一些源代码)而言,可以编写调用重要的网页函数和API的测试代码。来自各个测试用例的虚拟地址访问序列可以用于汇聚为目标虚拟地址点集合。应当理解的是,其它机制可以用于从应用二进制代码中提取功能。

[0043] 图4是示出用于检测本机二进制执行模型中的恶意代码活动的方法400的实施例的流程图。在框402处,生成用于应用的VAFMT 120。如上所述,VAFMT 120包括多个感兴趣的虚拟地址,该多个感兴趣的虚拟地址被映射到对应的高级应用功能。在框404处,可以在计算设备(例如,便携式计算设备)上安装应用。在框406处,可以将应用注册用于由HLOS 106提供的安全支持(例如,注册应用112)。在框408处,可以启动应用,并且作为响应,CPU 102可以执行应用二进制代码108。当注册应用112运行时,HLOS 106可以拦截应用的运行过程(框410)。在框412处,HLOS 106可以使用对应的VAFMT 120来在功能兴趣点被执行时检测和记录它们。在框414处,可以将所记录的点提供给恶意代码检测算法116以检测 and 解决恶意攻击。恶意代码检测算法116可以包括基于签名的算法、模式匹配算法或者采用机器学习或者其它技术。以这种方式,恶意代码检测算法116可以使用VAFMT 120来提供其作为输入接收的虚拟地址的含义。

[0044] 由于VAFMT 120在HLOS 106的控制之下,由HLOS 106执行的应用二进制代码108的虚拟地址的任何转换/随机化(例如,地址空间布局随机化(ASLR))可以应用于VAFMT 120中的虚拟地址,以保持它们与正在执行的申请的有效虚拟地址同步。在一个实施例中,从JavaScript代码以及利用VAFMT 120的应用执行收集的信息可以提供高级网页/JavaScript功能信息,其可以被提供给恶意代码检测算法116。在检测到任何恶意行为(框416)时,HLOS 106可以暂停应用/渲染器/JavaScript进程,并且为用户打开对话框,该对话框就潜在的危险进行警告,并且询问用户用于继续进行的指令。如果用户仍然想要继续进行,那么HLOS 106可以恢复浏览器进程。如果用户不想继续进行,那么HLOS 106可以请求用户关闭标签或者导航到某个其它网站,或者HLOS 106可以结束用于该执行实例(浏览器标签)的进程。

[0045] 当应用二进制代码110版本改变时,可以经由例如空中(OTA)更新来对VAFMT 120进行更新。这些更新确保HLOS 106准备好经更新的二进制文件用于任何注册应用112。经更新的二进制文件可以产生用于相同的兴趣点的新的虚拟地址。

[0046] 应当理解的是,HLOS 106和映射表120和122还可以被配置为支持涉及例如虚拟机118(图1)的托管运行时执行模型。在这点上,上述集成化平台机制使得系统100能够确定来自正在执行的应用二进制代码108的期望或者目标高级功能信息。在图5-10中示出了托管运行时执行模型的示例性实施例。

[0047] 在涉及托管运行时或者虚拟机执行的实施例中,可以借助于另一表(例如,标识符到地址映射表(IVAMT) 122),从虚拟机(VM)堆的不同部分中读取JavaScript源和/或用于JavaScript源的字节代码/即时(JIT)二进制文件。IVAMT 122包括用于VM堆的重要边界的虚拟存储器地址。其还可以包括其它类型的条目,在这些条目中,可以维护用于虚拟机118或者应用二进制108的各个功能点的虚拟地址。应当理解的是,IVAMT 122通常可以用于特定功能点的虚拟地址,其可以在应用执行期间更新和/或动态地确定。在这点上,IVAMT 122可以将功能点映射到虚拟地址。在另一方面,VAFMT 120可以将静态定义的虚拟地址映射到功能含义。因此,在应用执行期间VAFMT 120可以不改变,但是可以通过例如到计算设备的空中(OTA)更新来进行更新。还应当理解的是,其它各种各样的表可以与VAFMT 120和IVAMT 122相关联。所述各种各样的表可以包括各种宏或者参数名称,其映射到它们的不是虚拟地址的参数值或者设置。

[0048] 在图9的实施例中,标识用于示例性VM堆结构900的各个外部和/或内部边界的虚拟存储器地址901。如图9中所示,VM堆结构900可以包括标识各个外部和/或内部边界的多个数据字段,其包括例如始于字段912、至字段914、代码字段902、映射字段904、大对象字段906、旧数据字段908以及旧指针字段910。VM堆是VM托管存储器区域,其被分配在本机系统堆中。如本领域中已知的,在VM堆中,VM执行例如对以下各项的抽象:存储器管理、分配和解除分配代码(例如,JavaScript源)、字节代码、中间代码、JIT二进制文件、在执行期间创建的对象以及用于程序的执行的所有其它相关联的内务信息以及内部数据结构(例如,JavaScript平台)。如图9中进一步所示,根据VM所存储的事物的类型,VM堆区域可以包括各个子区域(例如,910、908、906、904、902、912和914)。子区域912和914可以用于包含第一次创建的对象,并且任何垃圾回收活动互换来自子区域912至914的活动对象,并且反之亦然。在一个实施例中,子区域902可以用于保存JavaScript源、字节代码、中间代码和JIT二进制/汇编代码。子区域904可以用于保存与由VM在程序(例如,JavaScript程序)的执行期间创建的对象相关联的某些内部数据结构。子区域906可以用于保存比预定尺寸(例如,1MB)大的任何种类的项(代码、对象)。子区域908和910可以保存已经幸免于多个垃圾回收周期的对象和数据,其中,子区域908关注具有常数值值的对象,以及子区域910关注指向其它对象的对象。

[0049] 在操作中,HLOS 106可以随着针对VM堆的存储器分配改变,而识别并且动态地更新IVAMT 122中的虚拟存储器地址901。应当理解的是,JavaScript虚拟机118保存该堆中的源,直到函数是活动的为止。托管运行时或者虚拟机执行模型可以涉及识别来自VM堆的JavaScript源和/或字节/JIT代码。可以针对任何新的写来跟踪具有JavaScript源的VM堆对象,并且可以识别由虚拟机118接收的JavaScript源。可以将所识别的JavaScript源提供给可信区114中的算法116,该算法116从JavaScript代码中提取各个特征,并且使用它们来检测任何恶意行为。从JavaScript代码中提取的特征的例子包括以下或者其它特征:文档对象模型(DOM)修改以及敏感函数;评估的数量;字符串的数量;脚本长度;字符串修改函数;用于去混淆(de-obfuscation)的“内置”等。可信区115可以将所提取的特征提供给恶意代码检测算法116,以确定任何恶意活动。

[0050] 在某些实施例中,当仅有JIT二进制/字节代码可用时,可以从它们中提取特征,并且然后将特征发送给恶意代码检测算法116。例如,HLOS 106可以维护表示高级JavaScript

工件 (artifact) 的字节代码/JIT代码序列的库。可以记录来自VM代码空间中的JavaScript函数的字节代码/JIT代码流与这些工件的任何匹配,并且将其传递给恶意代码检测算法116,以对恶意特性进行确定。

[0051] 图5和6示出了在托管运行时或者虚拟机执行期间使用的IVAMT 122和VAFMT 120的示例性实施例。图5示出了与VM代码空间的分配相关的目标功能到对应的应用二进制代码108的逻辑映射500。在该实施例中,应用源代码110包括用于“AllocateVMCodeSpace”函数的代码。如图5中所示,源代码110中的第一点可以被映射到二进制代码108中的虚拟地址502。源代码110中的第二点可以被映射到二进制代码108中的虚拟地址504。在示例实现中,当VM在执行期间获得其需要执行的新的JavaScript源代码,并且确定在当前VM堆代码空间中不存在太多空间(902)时,可以调用函数AllocateVMCodeSpace。该函数可以获得新的JavaScript代码的大小,并且确定VM堆代码空间在大小上需要被增加的量,以使得VM可以保存JavaScript源、相关联的字节代码或者中间代码和/或JIT二进制文件。基于所确定的大小,AllocateVMCodeSpace函数可以使用系统分配器函数(例如,mmap()、malloc()、calloc()或者realloc()),来增加本机平台的堆中的VM堆代码空间的所分配的空间。mmap()函数是兼容POSIX的Unix系统调用,其将在从文件描述符指定的其它对象的一偏移处起始(优选地,在地址起始处)的字节序列映射到存储器中。mmap()函数返回对象被映射的实际地方。Malloc()、realloc()、calloc()和free()包括C标准库中的用于以C/C++编程语言执行针对动态存储器分配的手动存储器管理的一组函数。可以将用于二进制代码108中的兴趣点的虚拟地址502和504直接放置在VAFMT 120中的列302中。由虚拟地址表示的不同兴趣点的功能含义可以作为宏名称列在VAFMT 120的列304中。检测算法116(图1)可以具有对由VAFMT 120的列304中的宏所表示的功能的清楚的理解。针对VAFMT 120中的特定行的宏名称(列304中)可以明确地标识当处理器(例如,CPU 102)执行应用在虚拟地址点(列302中)处的二进制指令时正在被执行的功能。以这种方式,通过知晓执行统计结果、用于兴趣点的虚拟地址的计数和分布,检测算法116充分地理解正在由高级应用二进制文件执行的功能。应当理解的是,映射可以直接在虚拟地址302与由宏(304)表示并且由执行处理或者检测的检测算法116理解的功能含义之间,从而消除知晓该虚拟地址兴趣点处的实际二进制指令。

[0052] 利用虚拟地址和宏含义表示的兴趣点可以离线确定,并且然后被填充于用于特定应用二进制文件的VAFMT 120中。许多类型的应用可以具有可用的匹配参考源代码。例如,匹配参考源代码可以可用于从普及的开源项目(例如,基于blink/Chromium的浏览器、基于Webkit的浏览器、安卓平台中的各种虚拟机(比如Dalvik、ART、RenderScript))开发的常用应用。对于具有可用匹配参考源代码的应用而言,各种离线机制可以用于确定用于商业应用二进制文件中的兴趣点的虚拟地址,以用于源代码中的用于那些兴趣点的对应表达式/语句。

[0053] 将对用于兴趣点的虚拟地址的离线确定的示例性实施例进行描述。可以在匹配参考源代码中识别源代码110中的实现感兴趣的功能的某些重要且有用的函数。可以将源代码110内的各个点手动地确定为形成将一起表示特定唯一的功能的唯一点集合。应当理解的是,这可以等效于源代码110内的用于该功能的采样点集合,该采样点集合唯一地表示完整的源代码110的总体功能。可以对源代码110进行编译、汇编以及链接到参考应用,该参考

应用等效于实际的商业第三方应用。二进制文件(参考和商业第三方二者)都可以源自于相同的源代码110,并且使用类似的构建技术(例如,编译、汇编、链接)和工具链。如本领域已知的,开源应用可以使用可免费获得的GCC或者LLVM工具链。编译器、汇编器和链接器工具可以用于生成参考二进制应用,并且可以记下与源代码中的重要点相对应的虚拟地址点。由于用于兴趣点的虚拟地址可以包括对二进制应用从其构建(编译、汇编、链接)的源代码110中的兴趣点的直接映射,所以参考二进制文件可以离线用于与商业二进制文件进行比较,以识别商业第三方二进制文件中的虚拟地址兴趣点。还应当理解的是,其它离线或者其它技术可以用于确定商业第三方二进制文件中的兴趣点的虚拟地址。在一个实施例中,图2示出了源代码110中的不同兴趣点(201、203、205)可以如何直接被映射到二进制文件108中的对应虚拟地址(202、204、206)。

[0054] 图6示出了图5中的VAFMT 120与示例性IVAMT 122之间的逻辑映射600。VAFMT 120包括二进制应用中的固定且已知的兴趣点的虚拟地址,该二进制应用的执行是感兴趣的并且正在被跟踪。每当二进制应用改变时,可以更新这些虚拟地址。IVAMT 122包括当二进制应用执行时被创建或者更新的特定点的虚拟地址,其可以是动态的,并且表示动态项的虚拟地址(例如,运行时缓冲器起始或者结束点)。VAFMT 120中的左侧列(302)包括虚拟地址,而右侧列(304)可以指示在该虚拟地址处的在二进制代码108中存在的功能描述。以这种方式,VAFMT 120将虚拟地址映射到功能含义。通常,IVAMT 122包括相反的内容。在这种情况下,功能含义或者宏名称是已知的,并且系统确定在二进制应用的执行实例中功能含义或者宏名称604被实现或者可用的虚拟地址602。IVAMT 122中的虚拟地址可以包括在运行时被确定的动态值。对于动态分配的缓冲器(或者虚拟机堆或者其子空间)的起始和结束被确定的情况而言,可以从VAFMT 120获取用于二进制应用中的正在进行动态的缓冲器/堆空间分配的函数内的兴趣点的虚拟地址。这些函数的执行可以通过检测VAMFT 120中的虚拟地址的执行来确定。此外,缓冲器/堆空间分配的起始/结束虚拟地址可以通过检测从这些函数调用的系统存储器分配函数来确定。可以在IVAMT (122) 中更新缓冲器/堆空间分配的这些确定的起始/结束虚拟地址。

[0055] 图7示出了垃圾回收对VM堆代码空间的影响以及可以如何在虚拟机118的垃圾回收活动存在的情况下一贯地确定JavaScript源。应当理解的是,垃圾回收是托管运行时或者虚拟机的不可缺少的活动,因为对新对象的分配以及对无用(dead)(即没有在使用中)对象的解除分配可以由运行时或者虚拟机118来明确地处理。从托管VM堆中取回(reclaim)无用(未使用的)对象的活动被称为垃圾回收。在这点上,当取回不需要的脚本对象或者其它对象时,可以重新组织VM堆,并且将现有对象来回移动以及进行压缩,以为新对象分配腾出空间。图7示出了这样的垃圾回收活动对VM堆代码空间704a的影响。VM堆代码空间704a包括JavaScript对象JS1、JS2、JS3、JS4。在垃圾回收事件之后,可以通过移除被垃圾回收器检测为不需要的或者无用的JavaScript脚本JS3,并且因此从VM堆代码空间704b中回收(删除),从而对它们进行压缩。然而,VM堆中的对象的任何这种移动(例如,移除、压缩等)改变确定JavaScript对象位于何处的虚拟地址起始和结束位置。在示例性方法中,可以通过在每次垃圾回收活动之后,重新运行图5和6中示出的用于VM堆和该堆内的各个空间(图9)的虚拟地址确定机制,来改变虚拟地址,从而在脚本对象在垃圾回收期间被移动的情况下,利用新值来更新虚拟地址。如图8中所示,内核可以跟踪在垃圾回收期间发生的对象移动以及它们

移动的距离。通过跟踪对象移动的地址偏移,可以更新JavaScript对象在VM堆代码空间中的起始和结束的虚拟地址值。以类似的方法,可以通过跟踪图9中示出的对VM堆的各个子空间的分配/解除分配/移动,来更新IVAMT 122中的用于VM堆的各个代码空间的虚拟地址。

[0056] 图10是示出用于检测托管运行时或者虚拟机执行模型中的恶意代码活动的方法1000的实施例的流程图。应当理解的是,图10中的框1002、1004、1006、1008和1010中表示的步骤或者功能通常可以与以上结合图4的方法所描述的框402、404、406、408和410相对应。在框1012处,方法1000检测用于VM堆分配器/取消分配器函数在被执行时的兴趣点虚拟地址。如框1014处所示,当该执行被检测到是在VM堆分配器/解除分配器函数内时,方法1000可以检测进入内核的系统分配器/解除分配器函数的入口VM,并且记录系统存储器分配/解除分配。基于此,方法1000可以计算并且确定VM堆的起始/结束虚拟地址。通过实现用于VM堆的特定分配区域(例如,代码空间、大对象空间等)的类似机制,可以确定用于VM堆内的特定子区域(例如,代码空间、大对象空间等)的起始/结束虚拟地址。如框1016处所示,一旦在框1014处确定用于存储JavaScript源代码对象的VM堆空间,方法1000就可以使用脚本对象头部签名/模式(具有二进制形式),来确定JavaScript对象在VM堆内的起始。JavaScript对象的长度可以从头部中提取的,并且用于提取整个JavaScript源代码。如框1018处所示,JavaScript源代码可以用于提取由检测算法116用于检测例如恶意行为的感兴趣的特定特征。在框1020处,可以基于例如在框1018中从JavaScript源中提取的特征,确定JavaScript代码的恶意行为。

[0057] 如上所述, VAFMT 120可以以离线方式初始地配置,并且被提供给计算系统100(图1)。在一个实施例中,当使得应用二进制代码108的新版本可用于计算系统100时, VAFMT 120可以类似地以离线方式进行更新,并且经由例如通信网络(被称为“空中(OTA)更新”)被提供给计算系统100。对于被频繁地更新的二进制应用而言,以这种方式更新VAFMT 120可能是缺点。应当理解的是,在应用二进制代码108的经更新版本中的二进制代码的相对大的部分可能保持不变。VAFMT 120中识别的功能兴趣点304可以包括应用二进制代码108和/或二进制代码的可能逐个版本没有改变的相对有限部分。

[0058] 例如,编译器操作和/或设置可以不是经常改变,以及二进制代码中的各个模块可以维护在各模块之间类似或者预定的偏移。图12-16示出了可以在计算系统100中实现的用于当安装了应用二进制代码108的新的或者经更新的版本时自动地更新VAFMT 120中的虚拟地址的各种机制。

[0059] 应当理解的是,这些机制可以减少针对用于各种类型的应用和/或用例的VAFMT 120的OTA更新的需求。例如,在网页安全应用的背景下,这些机制可以消除针对用于对基于相同的起源代码库的网页浏览器应用的最频繁类型的更新中的许多更新的OTA更新的需求。现有的网页浏览器应用可以在每周或者每月的基础上来更新二进制应用代码。用于新二进制版本的虚拟地址可能改变,即使当源代码还没有针对与功能兴趣点304相关的特定模块进行改变时。在这种情况下,在该应用中的除了功能兴趣点304之外的部分中存在源代码改变或者在该应用的其它部分中访问的变量类型和数据结构类型(例如,C++类、C-结构体、联合体等)方面存在改变的情况下,虚拟地址可能改变。此外,在编译器、汇编器和链路器选项中的某些种类的改变可能导致该应用的其它部分中的虚拟改变。

[0060] 图12示出了可以在计算系统100中实现的用于当安装了应用二进制代码108的新

的或者经更新的版本时自动地更新VAFMT 120的示例性机制的实施例。如图12中所示,可以利用元数据1200和一个或多个伪二进制代码模板1202来增补VAFMT 120。如以下更详细地描述的,元数据1200和伪二进制代码模板1202可以使得HLOS 106能够在利用新版本更新应用二进制代码108时确定用于功能兴趣点304的新虚拟地址302。

[0061] 应当理解的是,伪二进制代码模板1202包括操作语句序列,其将符号表示用于存储器中的存储位置以及用于本地变量的伪寄存器。伪二进制代码模板1202可以使用指示其目的的各种类别的伪寄存器。在一个实施例中,ArgumentReg#可以表示将参数传递给子例程的伪寄存器。ReturnReg可以包括当从子例程调用返回时的返回地址。ProgCounter可以包括由处理器的程序计数器指向的当前地址。ReturnValueReg#可以表示用于将来自子例程调用的值返回到调用器代码的寄存器。操作可以包括处理器中具有可以是变量或者存储位置的输入和输出的汇编操作的接近表示(close representation)。例如,AddWord变量可以指示大小为4字节或者1字的操作数的加法运算。LoadWord变量可以指示从具有预定大小(例如,4字节或者1字)的存储器中加载值。LoadByte变量可以指示从具有预定大小(例如,1字节)的存储器中加载值。branchEQ可以包括条件分支,其中,如果先前比较操作导致正在比较的操作数相等,则该条件分支进行分支到目标被作为操作数而提供。寻址模式或者地址计算可以独立于加载或者存储操作。在一个实施例中,利用基址寄存器和偏移的加载操作可以被划分为两种操作:加法运算,其通过将常数偏移值加到伪寄存器上来计算最终地址;之后是实际的加载操作,其使用包含所计算的最终地址的伪寄存器。这可以完成以保存具有最通用形式的表示,因为经更新的应用二进制文件可以使用各种形式的寻址模式。作为常数的操作参数可以由对有效的常数范围进行编码所需要的比特数量表示。

[0062] 例如,常数“Const8bits”可以用作操作的操作数,其指示该操作数是可以由8比特编码的任何有效值,并且因此,确定所允许的值的动态范围。一些操作数可以是硬编码常数(例如,“#8”指示值“8”)。直接分支操作的操作数可以被表示为从当前程序计数器的偏移(例如,“ProgCounter+#Const20bits”或者“ProgCounter+#12”)。伪二进制代码模板1202可以使用这些或者其它操作语句来实现感兴趣的功能。应当理解的是,操作语句可以用于识别新的经更新的二进制文件中的区域,其经由例如匹配功能或者模块来实现提取功能。匹配模块被配置为理解伪二进制代码模板1202和应用的实际二进制文件的格式和表示二者。匹配模块可以在操作窗口内执行逐个操作的比较,以检测匹配或者使用控制数据流和控制数据流内的操作来进行比较。

[0063] 可以使用各种匹配技术。伪二进制代码模板1202中的操作语句可以使用静态单赋值(SSA)表示,其中,一次仅分配特定的伪寄存器变量,从而披露操作语句之间的真实依赖关系。SSA表示可以实现应用的经更新的二进制文件中的功能区域的改进的匹配。术语“伪”指代以下事实:表示不是二进制可执行文件并且不使用处理器的实际的汇编指令、寄存器和寻址模式,并且不被汇编到二进制代码中。伪二进制代码模板1202提供功能参考,其中匹配模块使用其作为模板模式和指南来检测应用的经更新的二进制文件中的感兴趣的功能。应当理解的是,伪二进制代码模板1202的实际格式和表示是依赖的实现,并且可以使用各种其它替代方案。在其它实施例中,一些实现可以使用实际的汇编指令表示,或者类似于二进制应用在其上运行的CPU 102的汇编表示的表示。

[0064] 如上所述,HLOS 106可以维护注册应用112的列表。对于每个注册应用而言,HLOS

106维护包括用于功能兴趣点304的虚拟地址302的表(例如, VAFMT 120、IVAMT 122)。如图12中所示, VAFMT 120中的一个或多个虚拟地址302可以与伪二进制代码模板1202相关联。在图12的实施例中, 伪二进制代码模板1202与用于功能兴趣点304的特定集合的虚拟地址302的集合相关联, 功能兴趣点304的特定集合表示唯一功能(documentWrite函数)。伪二进制代码模板1202包括一般等效于覆盖documentWrite函数的二进制代码的伪代码指令。在一个实施例中, 伪二进制代码模板1202可以不使用处理器指令集架构(ISA), 并且不需要被汇编到实际的二进制代码中。伪二进制代码模板1202可以使用类似于汇编操作的操作语句, 并且使用伪寄存器和符号参考来进行存储。虽然使用这样的操作语句序列, 但是伪二进制代码模板1202可以实现感兴趣的功能(例如, 在以上例子中的“documentWrite”函数的功能), 其中, 其表示的该功能与在应用的实际二进制文件中实现的感兴趣的功能(例如, documentWrite函数)相同或者等效。应当理解的是, 计算系统100可以包括任何数量的伪二进制代码模板1202。不同的伪二进制代码模板1202的数量可以使得: 在VAFMT 120中捕获的所有不同的功能(虽然不同的功能兴趣点集合)都具有至少一个代表性伪二进制代码模板1202, 其用于当安装新应用二进制代码时更新其覆盖的函数点的虚拟地址。

[0065] 在一个实施例中, 伪二进制代码模板1202可以包括通用形式的目标汇编指令、或者一个或多个伪寄存器、以及从通用基础(例如, 全局堆或者栈、符号/变量名称)的存储器访问偏移(其表示存储器中的特定参考点)。元数据1200通常包括使用例如字节偏移的无虚拟地址的表示。用于虚拟地址(0x3473fac8)的元数据1200包括字节偏移($\text{BASE2} = \text{BASE0} + 74709704$)。用于虚拟地址(0x3473fad4)的元数据1200包括字节偏移($\text{BASE12} + 12$)。用于虚拟地址(0x3473fae8)的元数据1200包括字节偏移($\text{BASE12} + 32$)。应当理解的是, 该元数据可以形成与唯一地表示“document_write”功能的三个虚拟地址兴趣点的集合相对应的唯一集合。

[0066] 伪二进制代码模板1202可以初始以离线方式生成, 被提供给计算系统100, 并且存储在设备的安全存储装置中。应当理解的是, 当在例如由功能兴趣点304覆盖的区域中的代码和/或数据结构中存在明显的改变时, 可以仅需要更新伪二进制代码模板1202。这些类型的改变可以是相对不频繁的(例如, 每6个月一次)。可以经由OTA更新来实现该类型或者其它类型的更新。这可以能够实现从例如每周/每月基础的虚拟地址的OTA更新到仅在每6个月一次进行伪二进制代码模板1202的OTA更新的显著减少。

[0067] 可以检测用于现有的注册应用的新二进制版本的更新或者重新安装。作为响应, 元数据1200和伪二进制代码模板1202可以用于自动地更新VAFMT 120。如图12中所示, 伪二进制代码模板1202可以用于对新应用中的二进制代码的区域1206进行模式匹配, 其中, 由伪二进制代码模板1202表示的功能兴趣点304(并且因此该特定伪二进制代码模板表示的虚拟地址兴趣点)位于该区域1206中。元数据1200可以用于关注于在应用二进制代码108的经更新的版本中搜索区域1206。可以进行初始尝试, 以通过使用来自用于唯一功能的功能兴趣点304的原始基础(BASE0)的相对OFFSET, 对所关注的区域1206进行搜索(例如, 在基础BASE2之前以及之后的预定百分比)。应当理解的是, 在许多类型的频繁更新中, 这些相对偏移保持在附近。如图12中进一步所示, 当检测到匹配时, 可以从新二进制文件中获取新虚拟地址, 并且可以对VAFMT 120进行更新以反映新虚拟地址。如果一个或多个功能兴趣点304无法产生新二进制文件中的匹配, 那么计算系统100可以发起OTA更新, 或者在其它实施例

中,基于特定功能的重要性,从VAFMT 120中删除特定的感兴趣的功能以及相关联的虚拟地址。

[0068] 图13示出了来自图12的具有经更新的虚拟地址的VAFMT 120(由灰色方框表示)。与DOCUMENT_WRITE_FUNCTION_START兴趣点304相对应的虚拟地址302已经被更新为新虚拟地址(0x3133b61c)。与DOCUMENT_WRITE_1兴趣点304相对应的虚拟地址302已经被更新为新虚拟地址(0x3133b62c)。与DOCUMENT_WRITE_2兴趣点304相对应的虚拟地址302已经被更新为新虚拟地址(0x3133b62c)。如图12中进一步所示,还可以对与虚拟地址相对应的元数据1200进行更新。如图13中所示,用于新虚拟地址(0x3133b61c)的元数据1200已经被更新为“BASE2=BASE0+74709000”。这示出了在应用的经更新的二进制文件中的两个感兴趣的功能之间(即,在“KERNEL_ALLOCATOR_FUNCTION”与“DOCUMENT_WRITE_FUNCTION”之间)的轻微相对位置改变。该改变可以是相对轻微的。例如,该改变可以是在它们之间的74709704字节的总原始距离中减少704字节。因此,在已经以两种感兴趣的功能之间的基础偏移元数据(即74709704字节)之前和之后的某一容忍度关注了搜索的情况下,通过使得搜索区域变窄,而允许有效的匹配。用于新虚拟地址(0x3133b62c)的元数据1200已经被更新为BASE2+16。用于新虚拟地址(0x3133b640)的元数据1200已经被更新为BASE2+36。

[0069] 图14和15示出了与和DOCUMENT_WRITE函数相关的功能兴趣点304的集合相关联的伪二进制代码模板1202的示例性实施例。功能兴趣点304的集合包括DOCUMENT_WRITE_FUNCTION_START模块、DOCUMENT_WRITE_1模块以及DOCUMENT_WRITE_2模块。如图14中所示,在该集合中的功能兴趣点304中的每个功能兴趣点与特定伪代码指令直接相关联,特定伪代码指令形成伪二进制代码模板1202内的“伪二进制指令兴趣点”。基于经更新的应用二进制文件中的与“伪二进制兴趣点”直接匹配的特定二进制指令,伪二进制代码模板1202内的这些“伪二进制指令兴趣点”包括当前VAFMT 120中的虚拟地址兴趣点与应用二进制文件的经更新的版本中的新虚拟地址兴趣点的一对一映射。如图14中所示,DOCUMENT_WRITE_FUNCTION_START模块与保存前两个调用器保存的伪寄存器(CallSave0、CallSave1)以及返回寄存器(ReturnReg)的“入栈”操作相关联。其之后是AddWord操作,其计算随后的LoadWord操作所需要的地址。AddWord操作将应当适合放入8比特的常数值与程序计数器相加,并且将结果保存在伪寄存器reg0中。随后的LoadWord操作直接使用reg0中的地址作为要加载值的地址。在用于应用的实际二进制文件中,具有8比特常数的AddWord可以被直接包括在LoadWord指令中作为寻址模式的一部分。“Const8bits”允许具有适合放入8比特的任何常数的选项。将所加载的值保存在伪寄存器reg1中,并且将其用作将值加载在伪寄存器reg2中的第二LoadWord操作的地址。对于由DOCUMENT_WRITE_FUNCTION_START表示的功能兴趣点而言,“入栈”操作是该伪二进制代码模板1202中的“伪二进制指令兴趣点”。

[0070] DOCUMENT_WRITE_1模块与逻辑左移16比特的值的操作相关联,16比特的值被保存在伪寄存器(reg0)中并且保存在伪寄存器reg1中。然后将其与常数值“4”相加,并且将得到的值保存在伪寄存器reg2中,然后该得到的值用作一地址,其中值从该地址被加载在伪寄存器(reg3)中。应注意的是,对于实际的二进制加载指令而言,寻址模式可以直接执行与常数值4的加法,并且因此,AddWord和LoadWord可以由单加载指令表示。还可以将reg3中的值加到程序计数器值(PC)上,以创建伪寄存器reg4中的最终地址,该最终地址是如下的地址:字节值从该地址被加载到用于作为第一参数传递给所调用的例程的第一参数寄存器

“ArgumentReg0”中。在其之后,存在去往处于一偏移(其是可以适合放入20比特的值)处的地址的直接分支。然而,在直接分支指令之前,存在AddWord指令,该AddWord指令保存在直接分支对应用的不同部分进行控制之后要返回的地址(通过正确地设置ReturnReg)。“逻辑左移”操作是用于该伪二进制代码模板1202中的由DOCUMENT_WRITE_1表示的功能兴趣点的“伪二进制指令兴趣点”。

[0071] DOCUMENT_WRITE_2模块与AddWord操作相关联,该AddWord操作将可以适合放入8比特的常数值与程序计数器相加,并且将结果保存在伪寄存器reg0中。然后将伪寄存器reg0用作一地址,其中,值从该地址被加载在伪寄存器(reg2)中。其之后是另一AddWord操作,该AddWord操作将伪寄存器(reg2)和程序计数器的当前值相加并且将结果保存在伪寄存器reg1中。然后伪寄存器reg1用作一地址,其中,值从该地址被加载到ArgumentReg0中,ArgumentReg0用于通过直接分支指令将值传递给随后的子例程调用。应注意的是,对于实际的二进制加载指令而言,寻址模式可以直接执行与常数值的加法,并且因此,在应用的实际二进制文件中,AddWord和LoadWord可以由单加载指令表示。在LoadWord操作之后,存在去往处于一偏移(其是适合放入20比特的值)处的地址的直接分支。然而,在直接分支指令之前,存在AddWord指令,该AddWord指令保存在直接分支对应用的不同部分进行控制之后要返回的地址(通过正确地设置ReturnReg)。对子例程的调用之后跟随两组比较和去往伪二进制代码模板1202内的附近位置的分支。比较都是在第一子例程返回值寄存器(ReturnValueReg0)上完成的,以检查由子例程返回的特定值(“0”和“1”),并且基于所返回的值,分别使用BranchEQ和BranchNE操作本地地进行分支。分支目标地址被提供为从当前程序计数器值的常数偏移。将Const8bits操作数与程序计数器相加的AddWord操作是用于该伪二进制代码模板1202中的由DOCUMENT_WRITE_2表示的功能兴趣点的“伪二进制指令兴趣点”。应注意的是,应用的实际二进制文件可以具有伪二进制代码模板中的该地址计算操作(AddWord)以及LoadWord操作,其匹配到单个实际二进制指令(如“ldr r1,[pc,#80]”),并且在这种情况下,在“伪二进制指令兴趣点”全部匹配或者作为其子部分匹配的实际二进制指令变成确定应用的二进制文件的新版本中的经更新的虚拟地址的指令。

[0072] 图15示出了伪二进制代码模板1202中的伪代码指令与应用二进制代码108的经更新的版本1204中的所匹配的区域1206中的等效的对应二进制代码的匹配。在操作中,当伪二进制代码模板1202与区域1206匹配时,二进制代码中与功能兴趣点304匹配的对应指令的虚拟地址变成新虚拟地址,并且在VAFMT 120中进行更新。可以基于新虚拟地址来计算新基础和偏移,并且可以对元数据1200进行更新。

[0073] 图16示出了在计算系统100中实现的用于当安装应用二进制代码108的新的或者经更新的版本时自动地更新VAFMT 120的方法1600的实施例。在框1602处,可以将用于向HLOS 106注册的应用的虚拟地址映射表120存储在计算系统100中,如上所述。可以将VAFMT 120存储在HLOS 106中的安全存储器中。如图12中所示,VAFMT 120可以包括多个虚拟地址302的集合,其映射到用于注册应用的应用二进制代码108中的对应目标应用功能(功能兴趣点304)的。响应于接收到应用二进制代码108的经更新的版本1204(框1604),可以确定与虚拟地址映射表120中的多个虚拟地址302的集合中的一个或多个集合相关联的对应伪二进制代码模板1202(框1606)。如上所述,在一个实施例中,伪二进制代码模板1202连同初始VAFMT 120一起可以初始通过到系统100的空中(OTA)更新,或者通过下载代码/日期并且将

其安装到系统100的任何其它方式,来获取。这些伪二进制代码模板1202和VAFMT 120二者可以被存储在系统100中可由HLOS 106和内核访问的位置中。实际的存储地址是相互依赖的实现。各个级别的安全保护或者安全存储器配置可以被考虑用于存储位置,这取决于实现选择。当例如现有模板中的一个或多个模板无法在应用的经更新的二进制文件中发现任何匹配时,可以对伪二进制代码模板1202进行更新。由于感兴趣的区域中的应用代码中的大规模改变或者上述其它种改变,可能发生不匹配。在这样的情况期间,经更新的伪二进制代码模板1202和经更新的VAFMT 120可以是OTA下载的,并且被安装在系统100中。在决策框1608处,伪二进制代码模块1202用于对应用二进制代码108的经更新的版本1204进行搜索,并且将伪代码指令与等效二进制指令进行匹配。当发现匹配时,在框1610处,确定与二进制指令相对应的新虚拟地址。在框1612处,可以利用新虚拟地址以及对应的经更新的基础/偏移元数据1200,来对虚拟地址映射表120进行更新。

[0074] 如图16中所示,可以针对所有不同的伪二进制代码模板1202,重复框1606、1608、1610和1612,直到所有的伪二进制代码模板1202被匹配并且VAFMT 120中的所有虚拟地址被更新为止。在决策框1611处,方法1600可以确定所有伪二进制代码模块1202是否已经被处理。如果“是”,方法1600可以在框1613处结束。如果“否”,在框1606处,可以选择新的伪二进制代码模板1202。在决策框1608处,当在应用的经更新的二进制文件中针对特定伪二进制代码模板1202识别匹配的序列时,方法1600可以重复到下一伪二进制代码模板1202,以进行匹配。如果在某一重复处,在应用的经更新的二进制文件中不存在针对伪二进制代码模板1202的匹配,那么首先确定是否可以从VAFMT 120中删除感兴趣的功能(由伪二进制代码模板1202表示)(决策框1607)。如果可以删除其(其可以由于不同的原因(包括功能的重要性为低)造成的),那么可以从VAFMT 120中删除用于该感兴趣的功能的所有虚拟地址兴趣点(框1605),并且重复继续进行到框1606,以针对用于下一伪二进制代码模板1202的匹配进行搜索。然而,如果该功能(并且因此伪二进制代码模板1202)是重要的,并且不应当被删除(框1609),那么自动更新机制失败,在这种情况下,可以执行用于虚拟地址和/或伪二进制代码模板1202的完整的空中(OTA)更新。这可以表示在应用的经更新的二进制文件中存在重大改变/修改的情况(例如,这可以以较低的频率发生,6个月一次)。

[0075] 应当理解的是,可以将本文描述的方法步骤中的一个或多个步骤作为计算机程序指令(例如,上述模块)存储在存储器中。可以由任何适当的处理器与对应的模块结合或者合作来执行这些指令,以执行本文所描述的方法。

[0076] 在本说明书中描述的各过程或者各过程流中的某些步骤自然而然地在其它步骤之前,以便本发明如所描述地运作。然而,本发明并不限于所描述的步骤的次序,如果这样的次序或者顺序不改变本发明的功能。也就是说,应认识到的是,在不脱离本发明的范围和精神的情况下,一些步骤可以在其它步骤之前、之后或者与其并行地(与其基本上同时)执行。在一些实例中,可以在不脱离本发明的情况下,省略或者不执行某些步骤。此外,诸如“之后”、“然后”、“接下来”等的词并不旨在限制这些步骤的次序。这些词仅用于引导读者通读示例性方法的描述。

[0077] 此外,本领域技术人员在编程时能够基于例如本说明书中的流程图以及相关关联的描述,编写计算机代码,或者识别合适的硬件和/或电路,来实现所公开的发明。

[0078] 因此,特定程序代码指令集或者详细的硬件设备的公开并不被视为对于充分地地理

解如何实现和使用本发明而言是必需的。在以上描述中并且结合可以示出各个过程流的图,更加详细地解释了所发明的、所要求保护的计算机实现的过程的功能。

[0079] 在一个或多个示例性方面中,所描述的功能可以用硬件、软件、固件、或其任意组合来实现。如果用软件来实现,所述功能可以被存储在计算机可读介质上或作为计算机可读介质上的一个或多个指令或代码进行传输。计算机可读介质包括计算机存储介质和通信介质二者,通信介质包括促进将计算机程序从一个地方传输到另一个地方的任何介质。存储介质可以是能够由计算机访问的任何可用介质。通过举例而非限制的方式,这种计算机可读介质可以包括RAM、ROM、EEPROM、NAND闪存、NOR闪存、M-RAM、P-RAM、R-RAM、CD-ROM或其它光盘存储、磁盘存储或其它磁存储设备、或者可以用于携带或存储具有指令或数据结构形式的期望程序代码并且可以被计算机访问的任何其它介质。

[0080] 另外,任何连接被适当地称为计算机可读介质。例如,如果利用同轴电缆、光纤电缆、双绞线、数字用户线(“DSL”)或无线技术(例如,红外线、无线电和微波)从网站、服务器或其它远程源发送软件,则同轴电缆、光纤电缆、双绞线、DSL或无线技术(例如,红外线、无线电和微波)被包括在介质的定义中。

[0081] 如本文中所使用的,磁盘(disk)和光盘(disc)包括压缩光盘(“CD”)、激光光盘、光盘、数字多功能光盘(“DVD”)、软盘和蓝光光盘,其中磁盘通常磁性地复制数据,而光盘则用激光来光学地复制数据。上述各项的组合也应当包括在计算机可读介质的范围之内。

[0082] 对于本领域技术人员而言,在不脱离其精神和范围的情况下,本发明涉及的替代实施例将变得显而易见。因此,虽然已经示出并且详细地描述了所选择的方面,但是将理解的是,可以在不脱离本发明的精神和范围(如以下权利要求所定义的)的情况下在其中进行各种替换和改变。

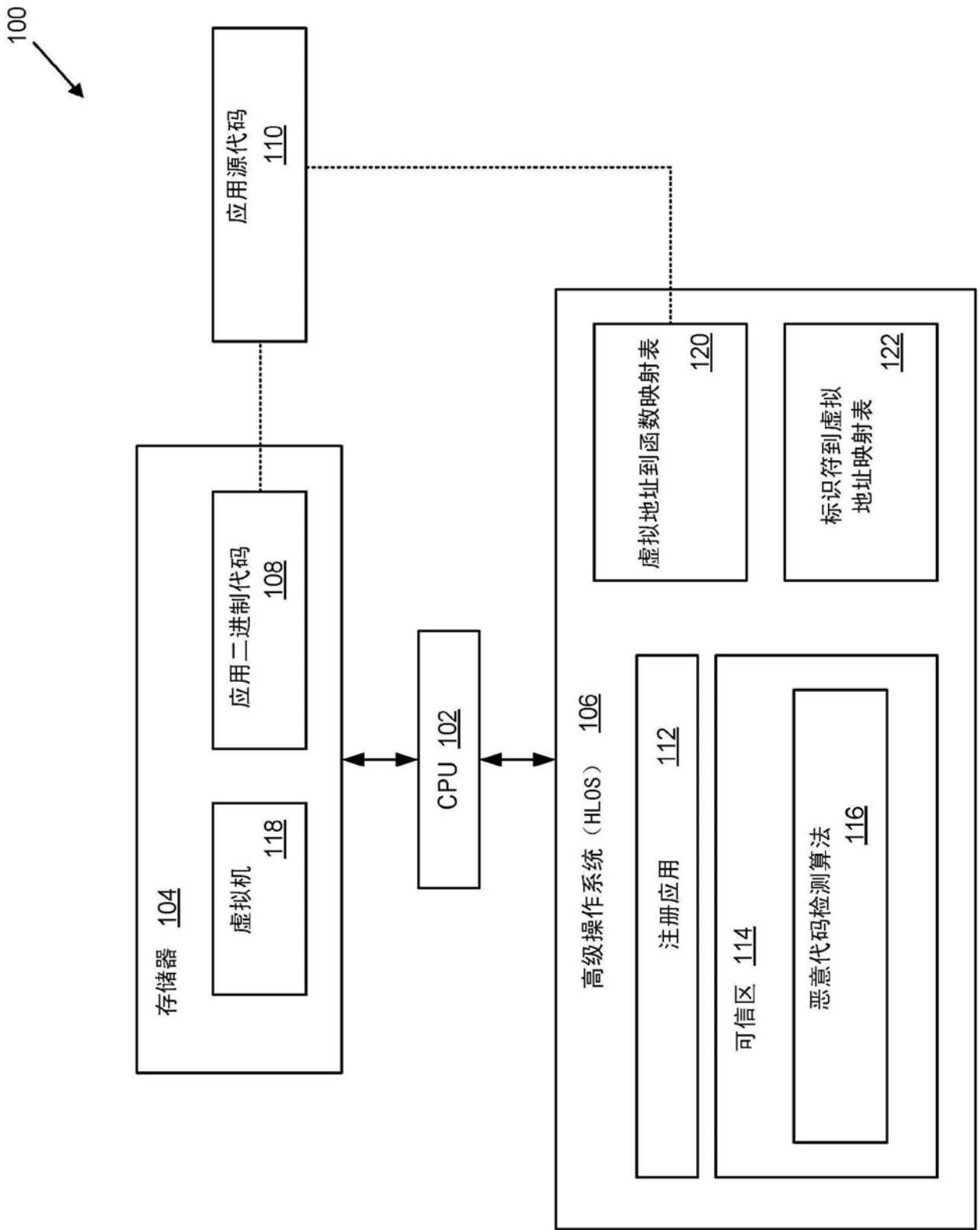


图1

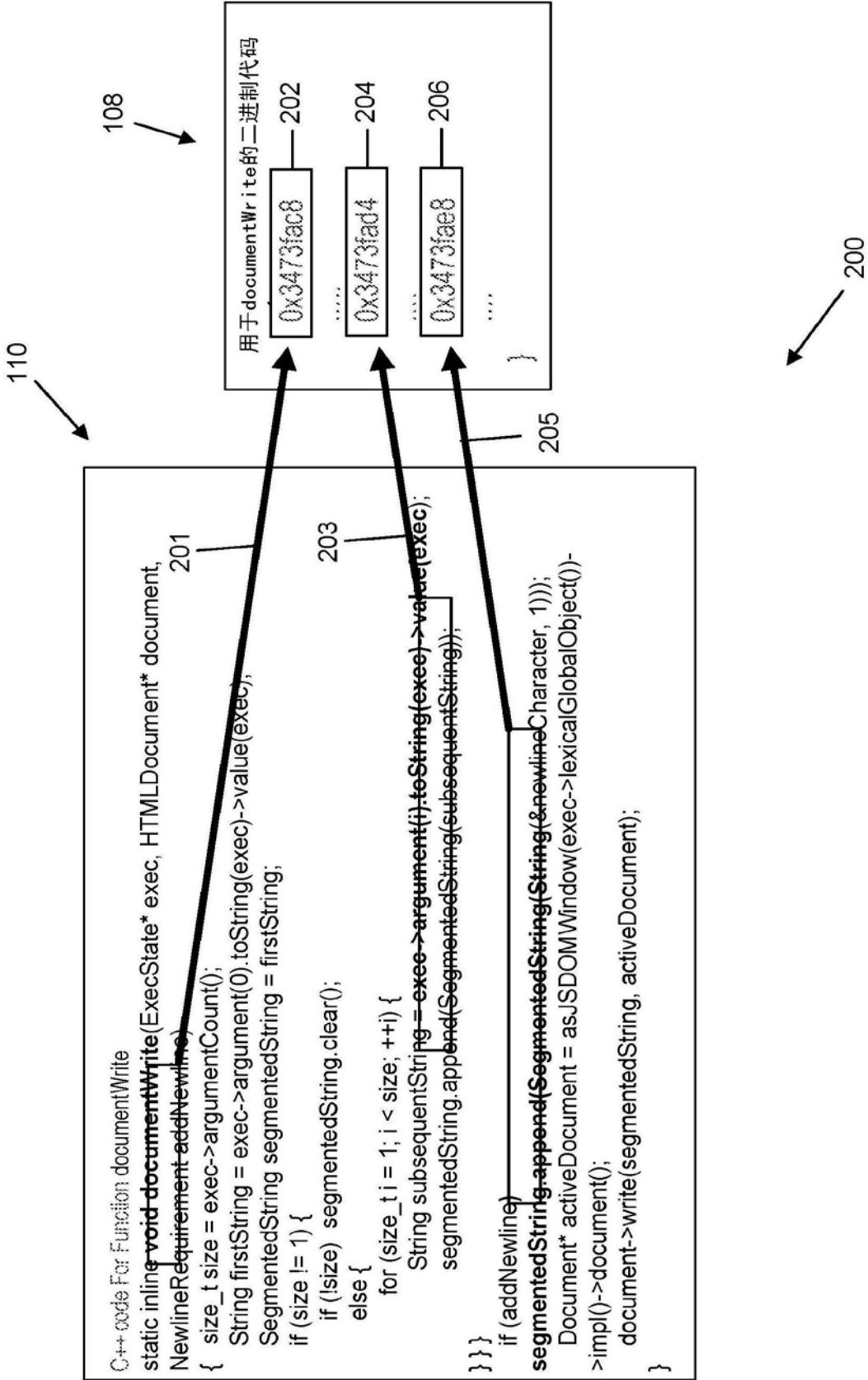


图2

300

虚拟地址到函数映射表			120
虚拟地址	302	功能兴趣点	304
0x3273fa94		EVAL_FUNCTION	
0x3473fac8		DOCUMENT_WRITE_FUNCTION_START	
0x3473fad4		DOCUMENT_WRITE_1	
0x3473fae8		DOCUMENT_WRITE_2	
0x29b93420		ONCLICK_FUNCTION	
0x59d782b4		DOCUMENT_COOKIE_FUNCTION	
0x59d78264		SETTIMEOUT_FUNCTION_START	

108

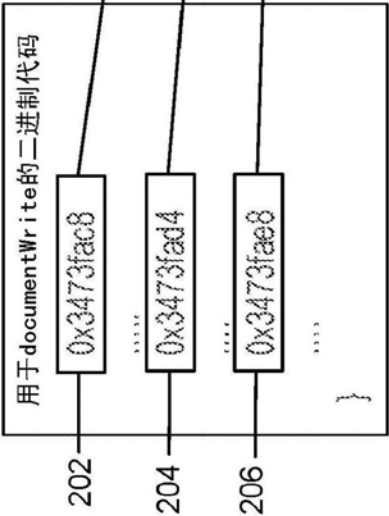


图3

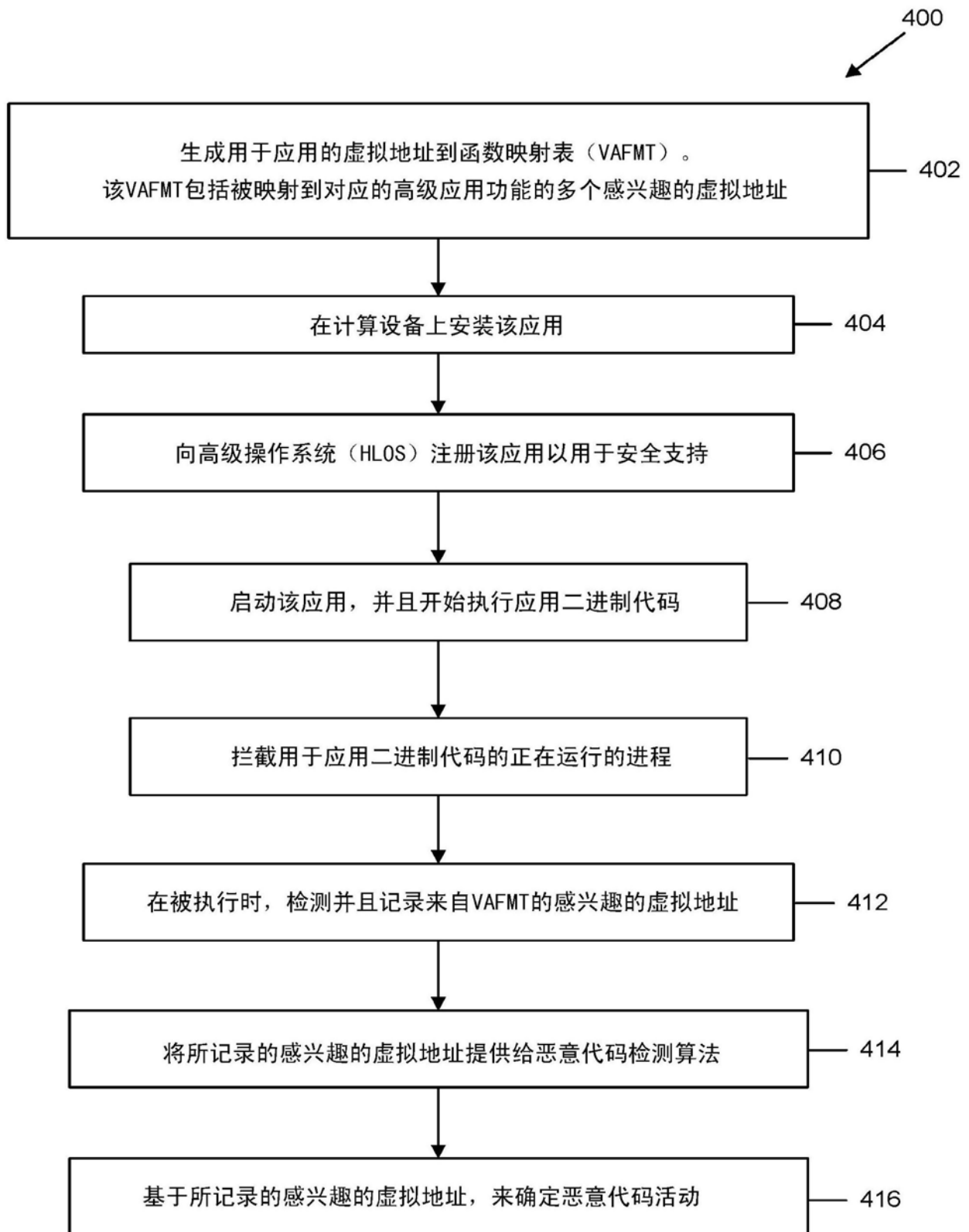


图4

500

虚拟地址到函数映射表			120
虚拟地址	302	功能兴趣点	304
0x3273fa94		EVAL_FUNCTION	
0x3473fac8		DOCUMENT_WRITE_FUNCTION_START	
0x3473fad4		DOCUMENT_WRITE_1	
0x3473fae8		DOCUMENT_WRITE_2	
0x29b93420		ONCLICK_FUNCTION	
0x59d782b4		DOCUMENT_COOKIE_FUNCTION	
0x59d78264		SETTIMEOUT_FUNCTION_START	
0x3673fa94		VM_CODE_SPACE_ALLOCATOR	
0x122242d4		KERNEL_ALLOCATOR_FUNCTION	
0x3673fad4		VM_CODE_SPACE_DEALLOCATOR	
0x122243e4		KERNEL_DEALLOCATOR_FUNCTION	

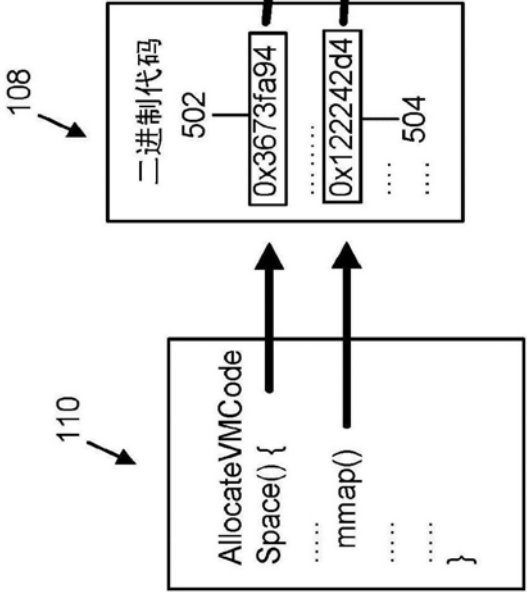


图5

虚拟地址到函数映射表			120
虚拟地址	302	功能兴趣点	304
0x3273fa94		EVAL_FUNCTION	
0x3473fac8		DOCUMENT_WRITE_FUNCTION_START	
0x3473fad4		DOCUMENT_WRITE_1	
0x3473fae8		DOCUMENT_WRITE_2	
0x29b93420		ONCLICK_FUNCTION	
0x59d782b4		DOCUMENT_COOKIE_FUNCTION	
0x59d78264		SETTIMEOUT_FUNCTION_START	
0x3673fa94		VM_CODE_SPACE_ALLOCATOR	
0x122242d4		KERNEL_ALLOCATOR_FUNCTION	
0x3673fad4		VM_CODE_SPACE_DEALLOCATOR	
0x122243e4		KERNEL_DEALLOCATOR_FUNCTION	

600 ↗

标识符到虚拟地址映射表			122
宏含义	604	虚拟地址	602
VM_CODE_SPACE_START		0x2fa2ca08	↓
VM_CODE_SPACE_END		0x3473fa80	
VM_LARGE_OBJ_SPACE_START			
VM_LARGE_OBJ_SPACE_END			
VM_NEWSPACE_START		0x3663fa80	
VM_NEWSPACE_END		0x3663fbb8	

图6

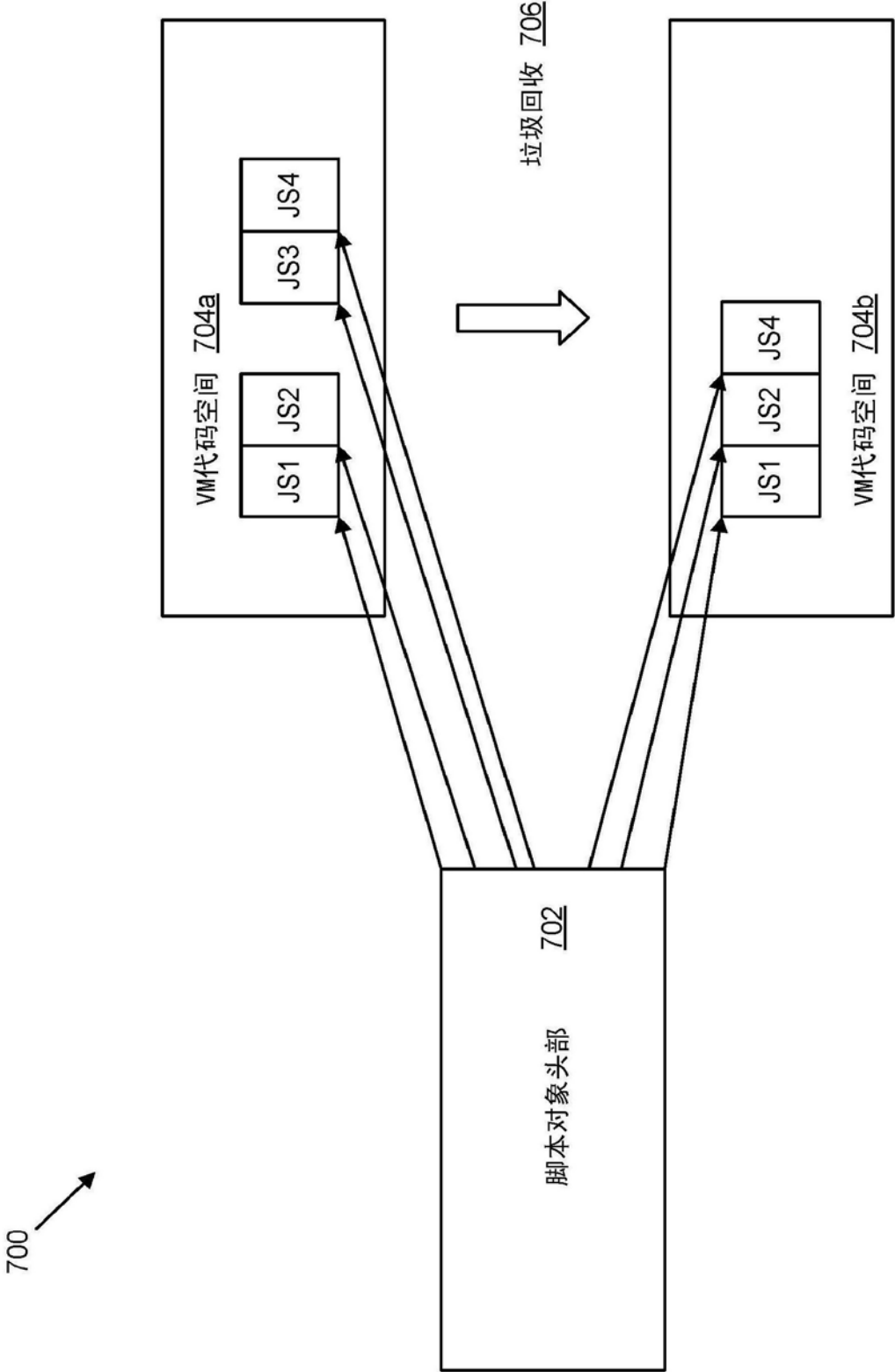


图7

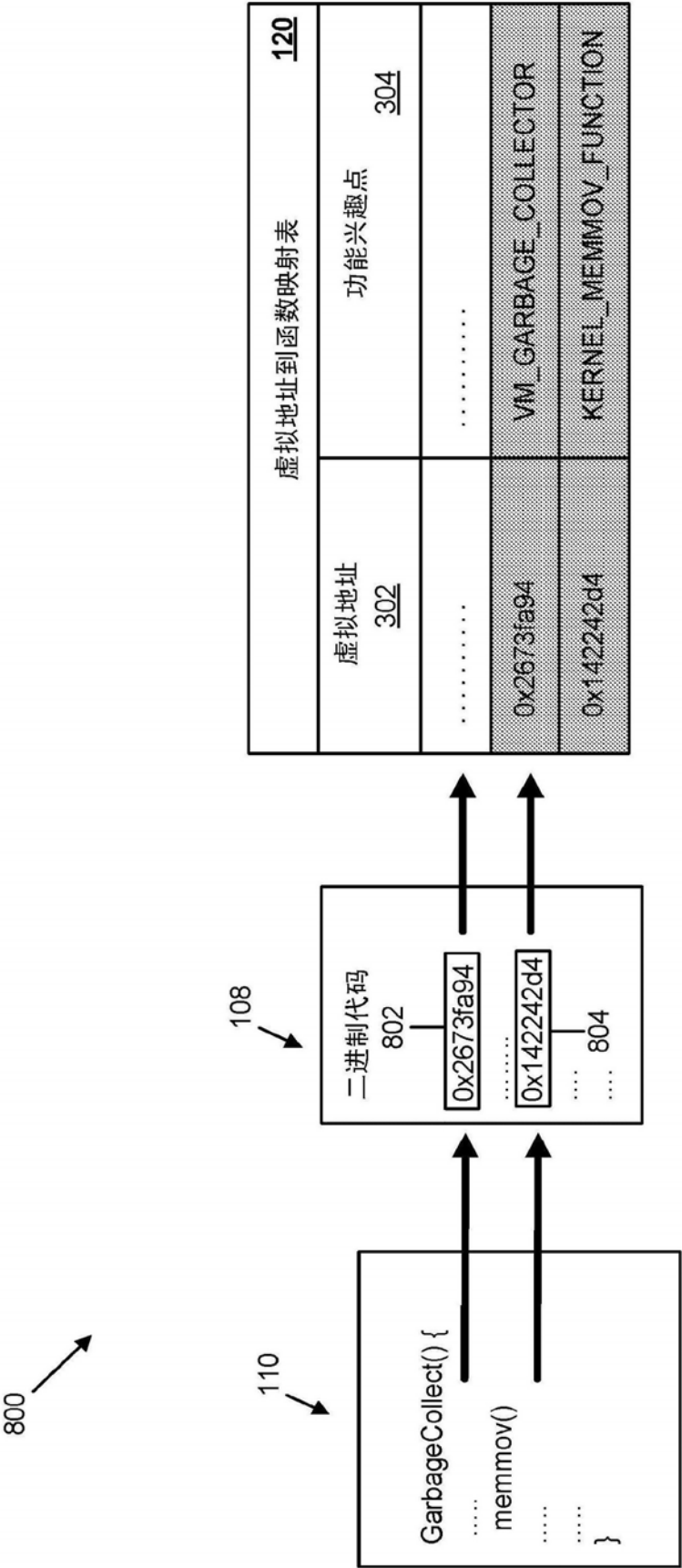


图8

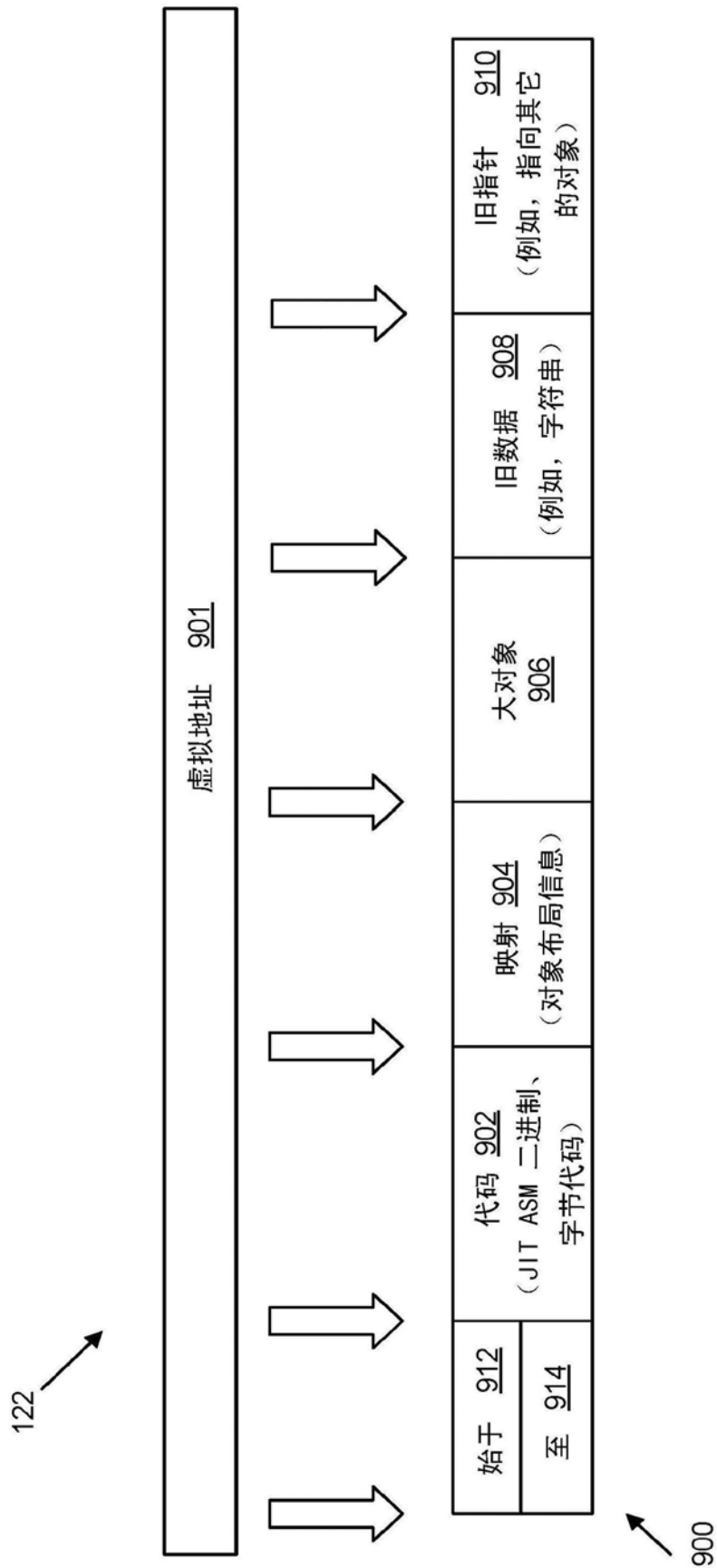


图9

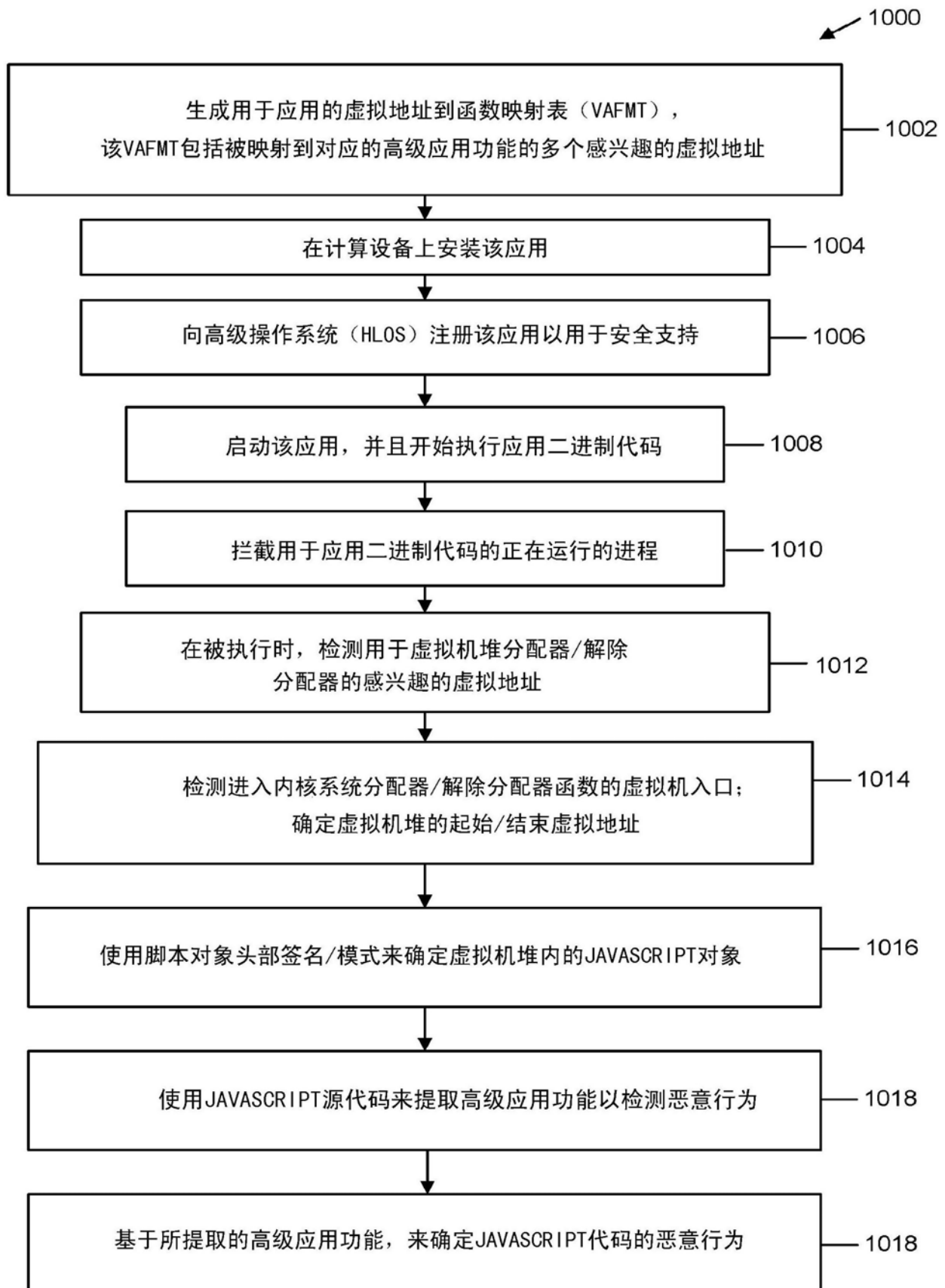


图10

120

1112

1104

具有偏移和长度的自定义虚拟地址映射表					
用于进行缓冲器或者数据结构分配的网页/浏览器应用函数的虚拟地址 (离线/静态地确定以及OTA更新) <u>1102</u>	用于从进行缓冲器或者数据结构分配的函数调用的“内核系统分配器”的虚拟地址 (在应用的动态加载期间确定的)	<u>1106</u>	缓冲器起始虚拟地址 (动态地确定)	缓冲器结束虚拟地址 (动态地确定) <u>1108</u>	所分配的缓冲器数据结构的成员字段的以字节为单位的长度 (离线/静态地确定以及OTA更新) (可选) <u>1114</u>
成员字段是否是指针吗 (离线/静态地确定以及OTA更新) (可选) <u>1114</u>	所分配的缓冲器数据结构的成员字段的以字节为单位的长度 (离线/静态地确定以及OTA更新) (可选) <u>1114</u>	所分配的缓冲器数据结构的成员字段的指针偏移 (离线/静态地确定以及OTA更新) (可选) <u>1110</u>	所分配的缓冲器数据结构的成员字段的指针偏移 (离线/静态地确定以及OTA更新) (可选) <u>1110</u>	是/否	是/否
0x3473fa94	0x4473fa94	0x2fa2ca08	0x3473fa80		
0x3473fac8	0x4473fa94	0x3663fa80	0x3663fb8		
.....					
.....					

图11

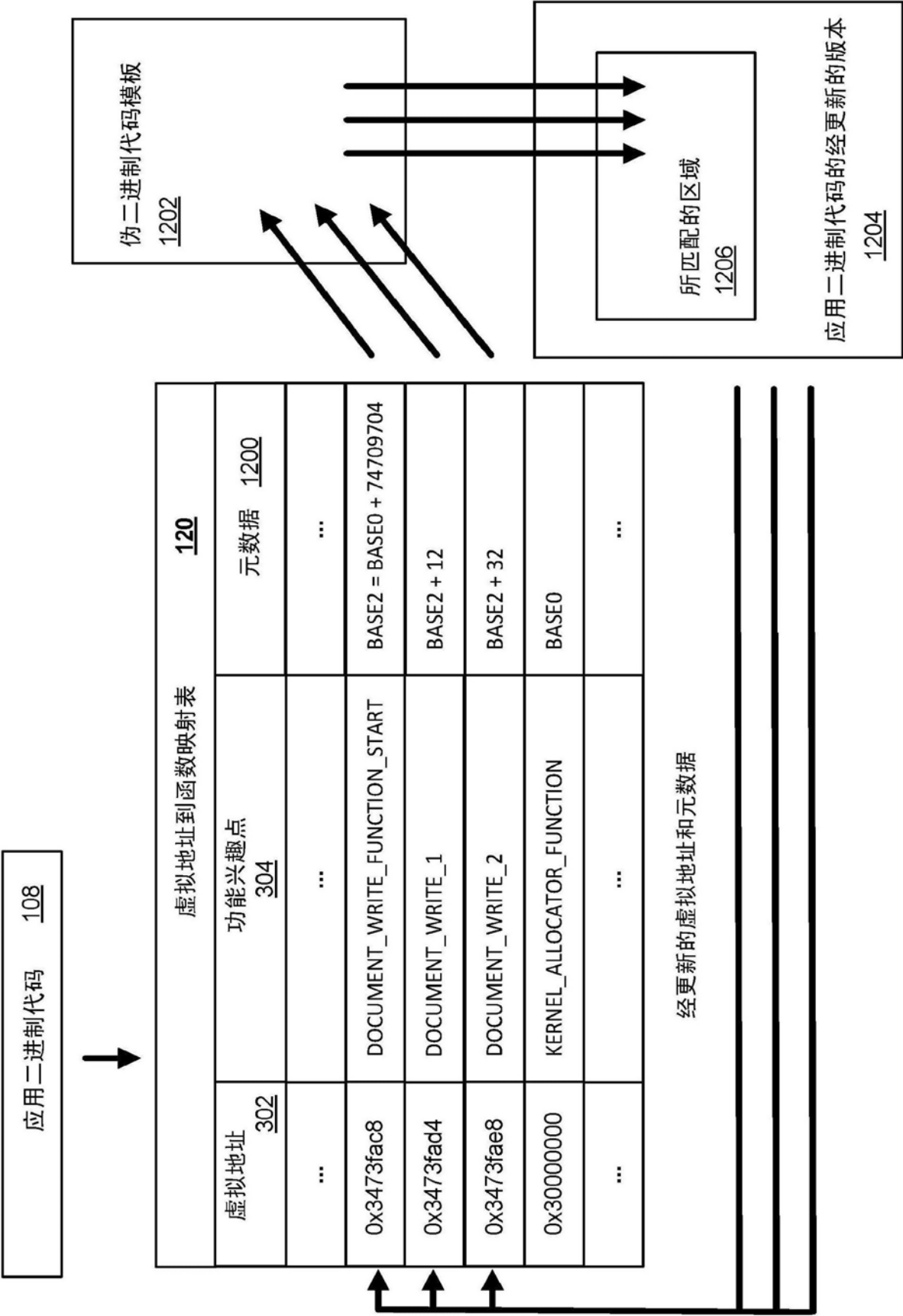


图12

虚拟地址到函数映射表			120
虚拟地址	表示功能含义的宏	304	使用字节偏移的无虚拟地址的表示
302			1200
...	...		...
0x3133b61c	DOCUMENT_WRITE_FUNCTION_START		BASE2 = BASE0 + 74709000
0x3133b62c	DOCUMENT_WRITE_1		BASE2 + 16
0x3133b640	DOCUMENT_WRITE_2		BASE2 + 36
0x2cbfe14	KERNEL_ALLOCATOR_FUNCTION		BASE0
...	...		...

图13

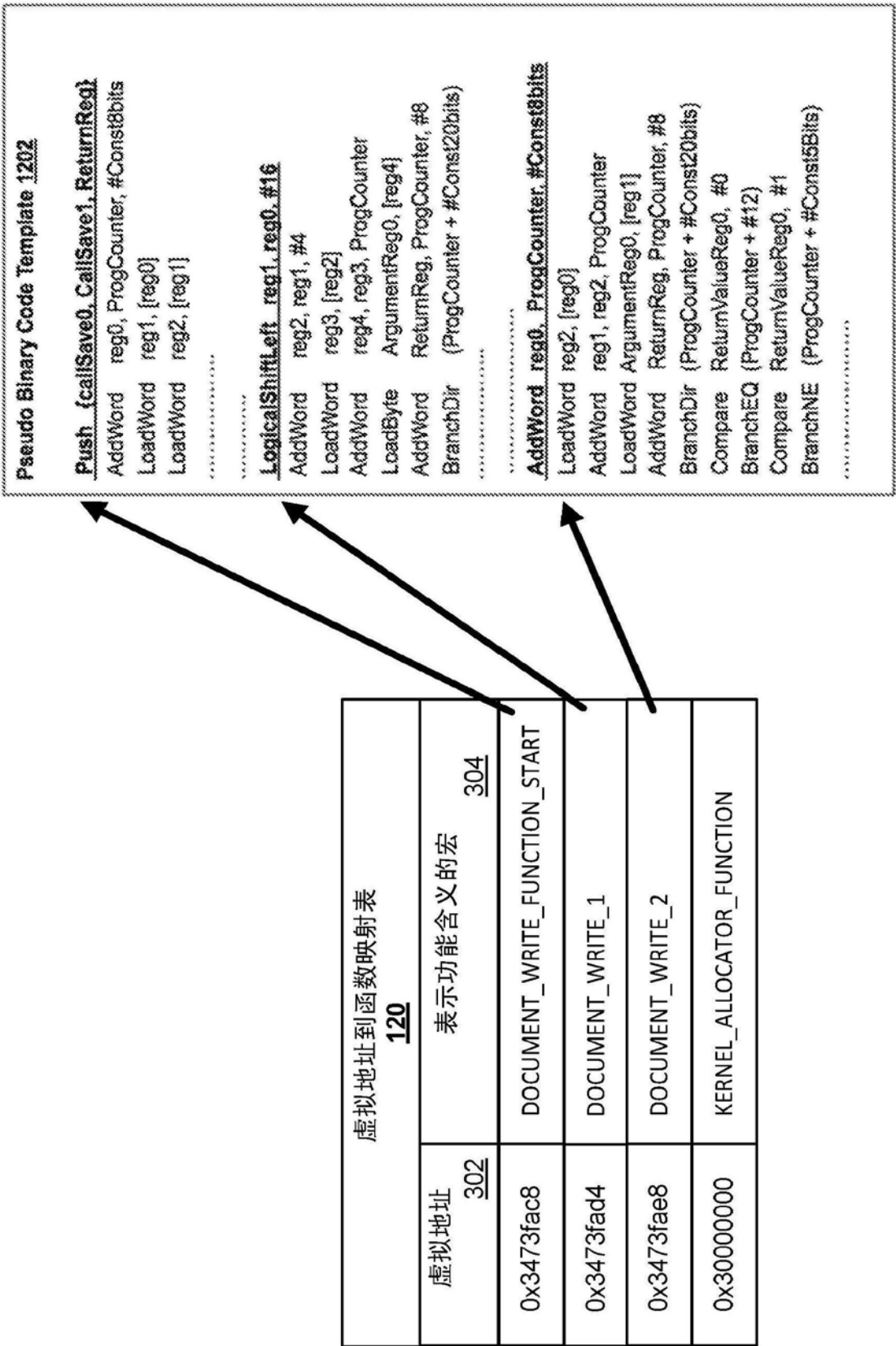


图14

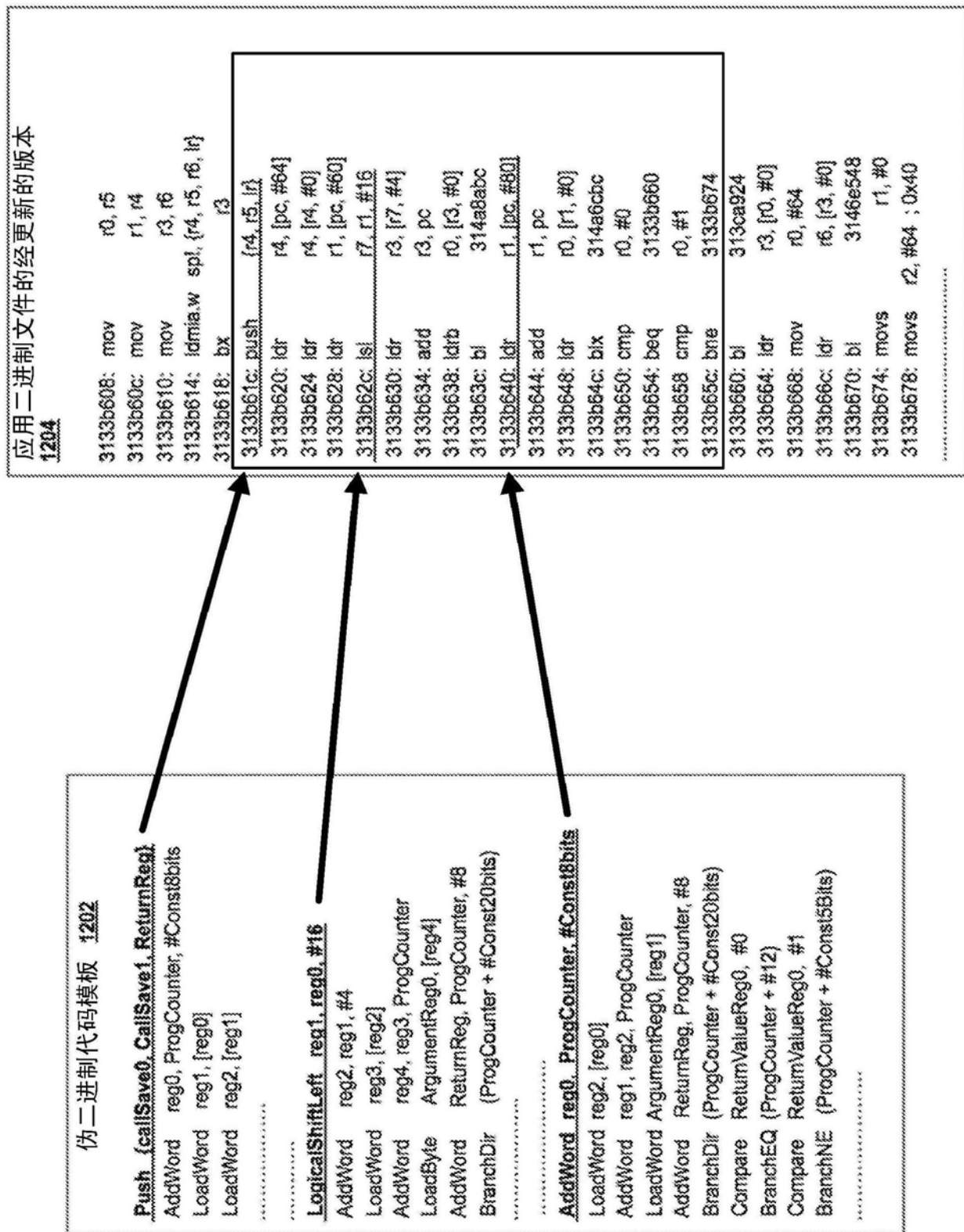


图15

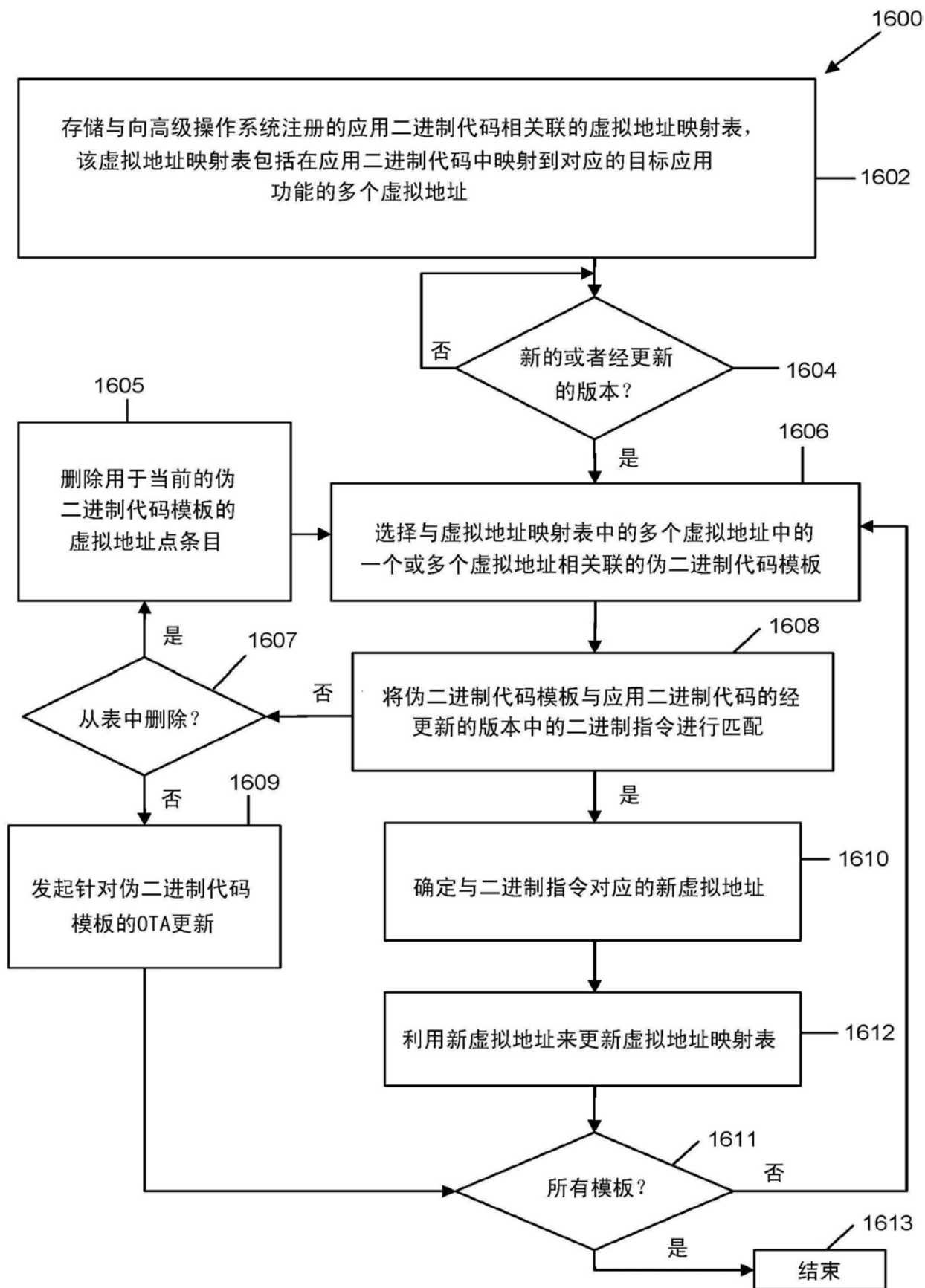


图16