



(12) 发明专利申请

(10) 申请公布号 CN 115989479 A

(43) 申请公布日 2023. 04. 18

(21) 申请号 202180053343.2

(22) 申请日 2021.05.03

(30) 优先权数据

17/008,270 2020.08.31 US

(85) PCT国际申请进入国家阶段日

2023.02.27

(86) PCT国际申请的申请数据

PCT/US2021/030388 2021.05.03

(87) PCT国际申请的公布数据

W02022/046192 EN 2022.03.03

(71) 申请人 微软技术许可有限责任公司

地址 美国华盛顿州

(72) 发明人 P·M·塞凯拉·德·胡斯托·特谢

拉

D·J·米霍克卡 J·R·贝里

R·C·哈德利 J·D·克利里

C·S·Y·丹格

(74) 专利代理机构 北京世辉律师事务所 16093

专利代理师 李峥宇

(51) Int.Cl.

G06F 8/41 (2006.01)

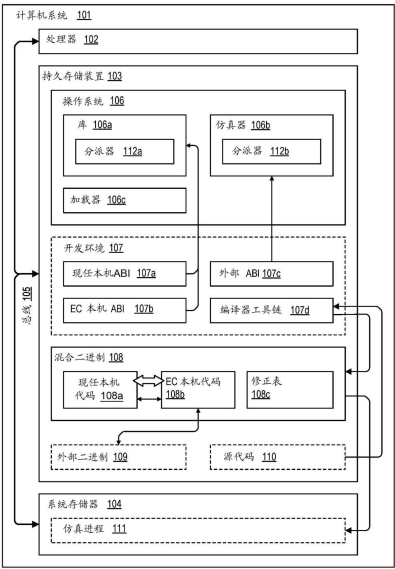
权利要求书2页 说明书24页 附图10页

(54) 发明名称

用于支持外部代码的仿真的本机仿真兼容的应用二进制接口

(57) 摘要

函数根据本机第一指令集架构 (ISA) 的第一应用二进制接口 (ABI) 和第二ABI被编译。第二ABI定义了不超过由外部第二ISA的第三ABI预期的的大小的上下文数据,并且使用被映射到第二ISA的寄存器的第一ISA的寄存器子集。当使用第一ABI和第二ABI两者来编译时,第二ABI对寄存器子集的使用导致某些函数可折叠。基于编译版本是否匹配,函数的第一编译版本和第二编译版本被标识为可折叠或不可折叠。当第一编译版本和第二编译版本不可折叠时,它们两者被发射到二进制文件中,而当它们可折叠时,仅第一编译版本或者第二编译版中的一个编译版本被发射到该二进制文件中。



1. 一种用于将源代码编译成二进制文件的方法,所述二进制文件以本机第一指令集体系结构(ISA)为目标,同时可与被编译成外部第二ISA的代码一起操作,所述方法被实现在计算机系统处,所述计算机系统包括处理器,所述方法包括:

标识(i)与所述第一ISA对应的第一应用二进制接口(ABI),(ii)与所述第一ISA对应的第二ABI,以及(iii)与所述第二ISA对应的第三ABI,其中所述第二ABI,

定义第一上下文数据格式,所述第一上下文数据格式具有第一大小,所述第一大小不超过由所述第三ABI使用的第二上下文数据格式的第二大小;以及

定义来自所述第一ISA的多个寄存器中并且由所述第一ABI使用的寄存器子集的使用,所述寄存器子集被映射到由所述第三ABI使用的所述第二ISA的寄存器的集合,当使用所述第一ABI和所述第二ABI中的每个来编译时,所述第二ABI对所述寄存器子集的使用导致函数的至少部分可折叠;以及

针对在所述源代码中定义的函数,至少执行以下操作:

生成以所述第一ABI为目标的所述函数的第一编译版本,生成所述第一编译版本包括在所述第一ISA中生成第一指令集;

生成以所述第二ABI为目标的所述函数的第二编译版本,生成所述第二编译版本包括在所述第一ISA中生成第二指令集;

至少基于确定所述第一指令集和所述第二指令集是否匹配,来确定所述函数的所述第一编译版本和所述函数的所述第二编译版本是否在所述二进制文件内可折叠;以及

至少基于确定所述函数的所述第一编译版本和所述函数的所述第二编译版本是否在所述二进制文件内可折叠,执行以下操作之一:

当所述函数的所述第一编译版本和所述函数的所述第二编译版本被确定为在所述二进制文件内不可折叠时,将所述函数的所述第一编译版本和所述函数的所述第二编译版本两者发射到所述二进制文件中;或者

当所述函数的所述第一编译版本和所述函数的所述第二编译版本被确定为在所述二进制文件内可折叠时,仅将所述函数的所述第一编译版本或者所述函数的所述第二编译版本中的一个编译版本发射到所述二进制文件中。

2. 根据权利要求1所述的方法,其中所述方法包括确定所述函数的所述第一编译版本和所述函数的所述第二编译版本在所述二进制文件内不可折叠,并且因此将所述函数的所述第一编译版本和所述函数的所述第二编译版本两者发射到所述二进制文件中。

3. 根据权利要求1所述的方法,其中所述方法包括确定所述函数的所述第一编译版本和所述函数的所述第二编译版本在所述二进制文件内可折叠,并且因此仅将所述函数的所述第一编译版本或者所述函数的所述第二编译版本中的一个编译版本发射到所述二进制文件中。

4. 根据权利要求1所述的方法,其中所述第二ABI使得所述二进制文件能够在以下两者上本机执行:(i)实现所述第一ABI但不实现所述第二ABI的第一计算机系统,以及(ii)实现所述第二ABI的第二计算机系统。

5. 根据权利要求1所述的方法,还包括:将入口形实转换程序发射到所述二进制文件中,所述入口形实转换程序包括在所述第一ISA中的代码,所述代码使所述第三ABI适应于所述第二ABI。

6. 根据权利要求1所述的方法,还包括:将一个或多个出口形实转换程序发射到所述二进制文件中,每个出口形实转换程序包括在所述第一ISA中的代码,所述代码使所述第二ABI适应于所述第三ABI。

7. 根据权利要求1所述的方法,还包括:

生成以所述第三ABI为目标的所述函数的第三编译版本,生成所述第三编译版本包括在所述第二ISA中生成第三指令集;以及

将所述函数的所述第三编译版本发射到所述二进制文件中。

8. 根据权利要求1所述的方法,其中由所述第二ABI对所述寄存器子集的使用使得被编译为以所述第二ABI为目标的函数的至少部分能够在没有入口形实转换程序的情况下从所述第三ABI被调用。

9. 根据权利要求1所述的方法,其中生成以所述第一ABI为目标的所述函数的所述第一编译版本包括使用所述第一ISA的源代码定义,并且生成以所述第二ABI为目标的所述函数的所述第二编译版本包括使用所述第二ISA的源代码定义。

用于支持外部代码的仿真的本机仿真兼容的应用二进制接口

背景技术

[0001] 在计算中,每个处理器或中央处理器(CPU)都实现一种指令集架构(ISA),该架构描述了机器代码指令必须遵守的语法规则,以便这些指令能够在该处理器上本机执行。当代ISA的示例有英特尔公司设计的IA-32ISA、超微半导体公司设计的x86-64 ISA、安谋国际科技设计的AArch64 ISA、国际商业机器公司设计的POWER ISA和加州大学伯克利分校设计的RISC-V ISA,尽管还有许多其他ISA。此外,在计算中,应用二进制接口(ABI)是两个二进制程序组件之间的接口,这些组件包括针对同一ISA的机器代码指令。ABI通常由操作系统(OS)定义,并且使第三方软件组件能够以相同ABI为目标的其他库/组件为目标,并且因此与之交互。例如,MICROSOFT WINDOWS提供了“WINDOWS-X64”ABI,该ABI使软件能够以包括实现x86-64 ISA的处理器在内的WINDOWS设备为目标,并提供了一种“ARM64”ABI,该ABI使软件能够将包括实现AArch64 ISA的处理器在内的WINDOWS设备作为目标。

[0002] 一般来说,软件组件相对于提供(例如,通过OS)该组件被编译到的ABI的设备是“本机”的,并且还包含具有与定义该组件的机器代码指令匹配的ISA的处理器。因此,任何与给定计算设备提供的ABI不匹配的软件组件都可以被认为是该设备的“外部”组件。例如,以针对x86-64的Linux ABI为目标的应用无法在具有以x86-64 ISA为目标但仅提供Windows-X64 ABI的处理器上本机执行。类似地,以Windows-X64 ABI为目标的应用无法在具有以针对x86-64 ISA为目标的处理器但仅为x86-64提供Linux ABI的计算设备上本机执行。

[0003] 此外,任何与设备的处理器的ISA不匹配的软件组件都可以被认为是该设备的“外部”组件。例如,以x86_64 ISA为目标的应用(而不管ABI目标是什么)不能在任何具有AArch64处理器的计算设备上本机执行。类似地,以AArch64 ISA为目标的应用(而不管ABI目标是什么)不能在任何具有x86-64处理器的计算设备上本机执行。

[0004] 某些环境(诸如WINDOWS)确实可以通过基于软件的ISA仿真来执行外部软件,但通常会造成严重的性能损失。因此,例如,可以通过AArch64仿真器在具有x86-64处理器的设备上仿真以ARM64 ABI为目标的应用,和/或可以通过x86-64仿真器在具有AArch64处理器的设备中仿真以Windows-X64 ABI为目标的应用。

[0005] 即使给定的软件应用多个ABI中可用,该应用的分发和使用也可能是一个问题。例如,普通计算机用户不知道其计算设备支持的ISA和/或ABI,因此无法就在其计算机上安装哪个版本的程序做出明智的决定,从而导致困惑和沮丧。此外,对于开发人员来说,将同一应用的多个版本分发和支持到不同的ABI/ISA是一种负担。试图解决这些问题的一个天真的解决方案是支持生成和操作系统支持所谓的“胖”二进制,其包括“本机”和“外部”码流。因此,例如,软件供应商可以为应用生成和分发单个胖二进制,该二进制包括当二进制被加载到基于AArch64的设备上时使用的AArch64码流,以及当二进制被加载到基于x86-64的设备上时使用的x86-64码流。虽然胖二进制在很大程度上向最终用户隐藏了不同处理器ISA的复杂性,但这是通过将二进制中包含的代码量增加一倍(即代码大小增加100%)来实现的,其中一半实际上未被给定设备使用。在一个经常具备有限的存储空间的可携带设备

很普遍的时代,胖二进制会很快消耗宝贵的存储资源,并导致软件故障和用户不满。

[0006] 考虑到当今可用设备的广泛多样的生态系统,用户通常希望能够在其计算设备上执行外部软件。因此,计算机制造商和/或操作系统供应商可能受到激励,使外部软件的执行成为一种相对无缝的体验。一些解决方案使外部软件能够通过仿真来执行,同时使外部软件可以与本机OS库交互。然而,这些解决方案受到了限制,因为这些应用仍然在仿真下执行,而不是在本机执行。

发明内容

[0007] 本文所描述的至少一些实施例有助于生成和/或消耗包括“仿真兼容”(EC)代码的本机二进制(例如,应用、模块、库等),该代码被配置为与仿真的外部代码紧密互操作,并且提供了与外部代码的高级别的互操作性和兼容。例如,本文的实施例使得应用开发者能够将应用定向到操作系统(OS)提供的本地EC应用二进制接口(ABI),以产生和分发在计算设备本地执行的应用。

[0008] 在实施例中,EC ABI表现出外部ABI的许多动作,从而使以EC ABI为目标的本地代码能够调用(经由仿真器)并与外部代码互操作。因此,借助于EC ABI,这些本地执行的应用能够高效且无缝地(例如,从终端用户的角度)与仿真的外部代码(例如,在该应用的执行上下文中加载的遗留库和插件)交互。

[0009] 在实施例中,EC ABI为本地代码提供上下文和调用约定(CC),其支持仿真外部代码内的异常展开和/或线程挂起。例如,在实施例中,EC ABI在保持与现任本机ABI期望的上下文数据及CC的兼容性的同时使用与外部ABI兼容的上下文数据和CC。

[0010] 在实施例中,即使EC ABI与外部ABI兼容,它也保持了与现任本地ABI的足够兼容性,使得针对EC ABI编译的代码的大部分与针对现任本地ABI编译的代码相同。这有助于在混合二进制中“折叠”该代码,该混合二进制既支持现任本地ABI(这使二进制能够被已知现任本地ABI但不知道EC ABI的遗留系统使用),也支持EC ABI(在已知EC ABI系统上实现与仿真外部代码的增强互操作性)。

[0011] 值得注意的是,在本文的实施例中,可存在编译到EC ABI的函数的三个入口点。首先,如果另一个EC ABI函数正在调用该函数,则可以通过其“真实”地址输入该函数。其次,如果外部ABI正在调用该函数,则可以通过“入口形实转换程序(entry thunk)”输入该函数。第三,如果该函数在运行时被在外部ABI下执行的代码修改,则可以通过“快进序列(fast-forward sequence)”输入该函数。无论函数的实际入口点是什么,当针对EC ABI编译的代码和针对外部ABI编译的代码交互时,实施例允许函数在跨ABI边界时具有一致的存储器地址参考,从而提高兼容性。

[0012] 一个或多个实施例涉及用于在可与编译成外部第二ISA的代码一起操作的同时将源代码编译成以本地第一指令集架构(ISA)为目标的二进制文件的方法、系统和计算机程序产品。在这些实施例中,计算机系统标识(i)与第一ISA对应的第一ABI,(ii)与第一ISA对应的第二ABI,以及(iii)与第二ISA对应的第三ABI。第二ABI(i)定义了第一上下文数据格式,该格式的第一大小不超过第三ABI使用的第二上下文数据格式的第二大小,并且(ii)定义了来自第一ISA中的多个寄存器中的寄存器的子集的使用,并且该寄存器的子集由第一ABI使用,寄存器的子集被映射到第三ABI使用的第二ISA的寄存器集,第二ABI对寄存器子

集的使用导致了当使用第一ABI和第二ABI中的每个编译时至少一部分函数可折叠。对于源代码中定义的函数,计算机系统生成以第一ABI为目标的函数的第一编译版本,包括在第一ISA中生成第一指令集,并生成以第二ABI为目标的函数的第二编译版本,包括在第一ISA中生成第二指令集。计算机系统至少基于确定第一指令集和第二指令集是否匹配来确定该函数的第一编译版本和该函数的第二编译版本在二进制文件内是否可折叠。至少基于确定该函数的第一编译版本和该函数的第二编译版本在二进制文件内是否可折叠,计算机系统或者(i)将该函数的第一编译版本和该函数的第二编译版本发出到二进制文件中(即,当该函数的第一编译版本和该函数的第二编译版本被确定为在二进制文件内不可折叠时),或者(ii)仅将该函数的第一编译版本或该函数的第二个编译版本发出到二进制文件中(即,当函数的第一编译版本和函数的第二编译版本被确定为在二进制文件内可折叠时)。

[0013] 一个或多个实施例还涉及用于生成混合二进制图像的方法、系统和计算机程序产品,该混合二进制图像在本机ABI和兼容ABI下可执行。在这些实施例中,至少基于标识与本机ABI对应的第一机器类型,计算机系统将第一机器类型发出到混合二进制图像的机器类型字段中。机器类型字段被构造为当混合二进制图像由在本机ABI下执行的本机进程加载时被利用。至少基于标识不可折叠的第一函数,计算机系统还将以下两者发出到混合二进制图像中:(i)在本机ABI下可执行的第一函数的第一编译版本,以及(ii)在兼容ABI下可执行的第一函数的第二编译版本。至少基于标识可折叠的第二函数,计算机系统还将第二函数的编译版本发出到混合二进制图像中,该第二函数的编译版本在本机ABI和兼容ABI两者下可执行。第二函数的编译版本被构造为当混合二进制图像由本机进程加载时调用第一函数的第一编译版本。计算机系统还将修正表发出到混合二进制图像中,该修正表被构造为当混合二进制图像由在兼容ABI下执行的兼容进程加载时被利用。修正表定义针对从混合二进制图像加载的存储器的多个变换,该多个变换包括第一变换,其调整机器类型字段以包括与兼容ABI对应的第二机器类型,以及第二变换,其配置第二函数的编译版本以调用第一函数的第二编译版本而不是第一函数的第一编译版本。

[0014] 一个或多个实施例还涉及用于通过在兼容ABI下执行的进程来消耗混合二进制图像的方法、系统和计算机程序产品,该混合二进制图像在本机ABI和兼容ABI两者下可执行。在这些实施例中,在混合二进制图像的加载期间,计算机系统确定存储在混合二进制图像的机器类型字段中的第一机器类型与和正在执行进程的兼容ABI对应的第二机器类型不匹配。基于确定第一机器类型与第二机器类型不匹配,计算机系统确定在二进制图像内定位修正表,该修正表定义针对从混合二进制图像加载的存储器的多个变换。计算机系统然后将多个变换中的至少一部分变换应用于从混合二进制图像加载的存储器,包括至少应用:第一变换,其调整机器类型字段以包括与兼容ABI对应的第二机器类型,以及第二变换,其修改调用站点以代替调用在兼容ABI下可执行的非折叠函数的第二编译版本,该调用站点调用在本机ABI下可执行的非折叠函数的第一编译版本。

[0015] 一个或多个实施例还涉及用于在处理进程内的调用时使用公共参考存储器地址的方法、系统和计算机程序产品,该进程支持以下两者的执行:(i)以与本机ISA对应的本机ABI为目标并且具有第一CC的本机代码,以及(ii)以与外部ISA对应的外部ABI为目标并且具有第二CC的外部代码。在这些实施例中,至少基于标识以针对被调用方函数的参考存储器地址为目标的调用,计算机系统确定被调用方函数与本机ABI对应还是与外部ABI对应。

被调用方函数基于以下之一被确定为与本机ABI对应：(i) 指示参考存储器地址被包含在存储本机代码的第一存储器范围内的查找结构，或(ii) 在参考存储器地址处标识的快进序列。至少基于指示参考存储器地址被包含在不存储本机代码的第二存储器范围内的查找结构，被调用方函数被确定为与外部ABI对应。至少基于确定，计算机系统基于执行以下之一来发起被调用方函数的执行：(A) 当调用方函数与外部ABI对应时，并且当被调用方函数被确定为与外部ABI对应时，在仿真器内使用参考存储器地址直接调用被调用方函数；(B) 当调用方函数与外部ABI对应时，并且当被调用方函数被确定为与本机ABI对应时，调用入口形实转换程序，入口形实转换程序(i) 使第二CC适应于第一CC，并且然后(ii) 使用参考存储器地址直接调用被调用方函数；(C) 当调用方函数与本机ABI对应时，并且当被调用方函数被确定为与外部ABI对应时，调用出口形实转换程序(exit thunk)，出口形实转换程序(i) 使本机ABI的第一CC适应于外部ABI的第二CC，并且然后(ii) 激活仿真器以使用参考存储器地址直接调用被调用方函数；或(D) 当调用方函数与本机ABI对应时，并且当被调用方函数被确定为与本机ABI对应时，使用参考存储器地址直接调用被调用方函数。

[0016] 提供该发明内容是为了以简化形式介绍概念的选择，这些概念将在下面的具体实施方式中进一步描述。本发明内容不旨在标识所要求保护的的主题的关键特征或基本特征，也不旨在用作确定所要求保护主题的范围的帮助。

附图说明

[0017] 为了描述可以获得本发明的上述和其他优点和特征的方式，将通过参考附图中所示的本发明的具体实施例来对上面简要描述的本发明进行更具体的描述。理解到这些附图仅描绘了本发明的典型实施例，因此不应被视为限制其范围，将通过使用附图以附加的具体性和细节来描述和解释本发明，其中：

[0018] 图1示出了有助于生成和/或使用包括仿真兼容(EC)代码的本机二进制的示例计算机系统，该EC代码被配置为与仿真的外部代码紧密互操作并且提供与外部代码的高级别互操作性和兼容性；

[0019] 图2示出了EC本机应用二进制接口(ABI)的示例，其在保持与现任本机ABI兼容的同时定义了镜像外部ABI的上下文和调用约定(CC)；

[0020] 图3A示出了双架构混合二进制的示例；

[0021] 图3B示出了导入和/或导出表的窗口视图的示例；

[0022] 图4A示出了编译为针对现任ABI的“现任”本机函数的调用动作，以及编译为针对EC ABI的“EC”本机函数的调用动作；

[0023] 图4B示出了“EC”本机函数的调用动作，该本机函数已被折叠，因为将源代码函数编译为现任ABI和EC ABI产生了相同的代码；

[0024] 图5示出了用于在可与编译为外部第二ISA的代码一起操作的同时以本机第一指令集架构(ISA)为目标将源代码编译为二进制文件的示例方法的流程图；

[0025] 图6示出了生成混合二进制图像的示例方法的流程图，该混合二进制图像可以在本机ABI和兼容ABI下执行；

[0026] 图7示出了用于通过在兼容ABI下执行的进程来消耗混合二进制图像的示例方法的流程图，该混合二进制图像可以在本机ABI和兼容ABI两者下执行；并且

[0027] 图8示出了在进程内处理调用时使用公共参考存储器地址的示例方法的流程图，该进程支持以下两项的执行：(i) 以与本机ISA对应的本机应用二进制接口 (ABI) 为目标并且具有第一CC的本机代码，以及 (ii) 以与外部ISA对应的外部ABI为目标并且具有第二CC的外部代码。

具体实施方式

[0028] 图1示出了示例计算机系统101，该系统有助于生成和/或消耗包括“仿真兼容”(EC) 代码的本机二进制 (例如，应用、模块、库等)，该代码被配置为与仿真的外部代码紧密互操作，并且提供了与外部代码的高级别互操作性和兼容性。计算机系统101另外有助于生成和/或消耗本机二进制，该本机二进制虽然包括EC代码，但仍然与不知道这些二进制中包含的EC特征的“遗留”计算设备兼容。

[0029] 一般而言，计算机系统101基于呈现外部ABI的许多动作的EC本机应用二进制接口 (ABI) 进行操作，从而使得以EC本机ABI为目标的本机代码能够调用 (经由仿真器) 并与外部代码 (例如遗留插件、库等) 进行互操作。在实施例1中，EC本机ABI为本机代码提供上下文和调用约定 (CC)，其支持模拟外部代码内的异常展开和/或线程挂起。例如，在实施例1中，EC本机ABI在保持与现任本机ABI期望的上下文数据及CC的兼容性的同时使用与外部ABI兼容的上下文数据和CC。在这样做的进程中，许多针对EC本机ABI编译的代码与针对现任本机ABI所编译的代码相同，有助于将代码“折叠”在混合二进制中，该混合二进制既支持现任的本机ABI (这使得二进制能够被已知现任本机ABI但不知道EC本机ABI的遗留系统使用)，也支持EC本机ABI (这使得能够在已知EC本机ABI的开放系统上增强与仿真外部代码的互操作性)。此外，当针对EC本机ABI编译的代码和针对外部ABI编译代码交互时，实施例1允许函数在跨ABI边界时具有一致的存储器地址参考，从而提高兼容性。

[0030] 在实施例1中，计算机系统101包括或利用专用或通用计算机硬件，例如，处理器102 (或多个处理器)、持久存储器103和系统存储器104，它们使用通信总线105通信耦合。在计算机系统101中，处理器102可以实现任何可用的处理器指令集架构 (ISA)，例如x86-64、AArch64、POWER、RISC-V等，在本文中称为处理器102的“本机”ISA。与处理器102的本机ISA不匹配的任何ISA在本文中称为“外部”ISA。通常，处理器102的ISA定义了处理器102的许多硬件方面，例如可由处理器102执行的机器代码指令的语法、由处理器102公开以供这些机器代码指令使用的一组寄存器、处理器102使用的存储器模型等。因此，例如，与处理器102实现x86-64 ISA相比，如果处理器102要实现AArch64 ISA，那么它将执行一组不同的机器代码指令 (包括例如一个或多个可用指令、指令格式等)，并公开一组不同的寄存器 (包括例如寄存器名称、寄存器大小、寄存器数目等中的一个或多个)。

[0031] 持久存储器103存储表示可执行软件组件的计算机可执行指令和/或数据结构；相应地，在 (多个) 处理器102处执行该软件期间，这些计算机可执行指令和/或数据结构的一个或多个部分被加载到系统存储器104中。例如，持久存储器103被示出为潜在地存储与操作系统106、开发环境107、混合二进制108、外部二进制109和源代码110对应的计算机可执行指令和/或数据结构。

[0032] 系统存储器104能够存储各种各样的数据，这些数据可以从持久存储器103加载、由处理器102存储和/或从诸如网络设备 (未示出) 之类的某个其他位置获取。通常，计算机

系统101通过将由存储在持久存储器103上的一个或多个二进制图像(例如,混合二进制108、外部二进制109等)定义的存储器页加载到系统存储器104中,并使用处理器102在这些存储器页上操作(加载到系统内存104中)来操作。这包括执行存储在这些存储器页中的机器代码指令,以对存储在这些存储器页中的数据进行操作。

[0033] 操作系统106(以下称为OS 106)包括库106a,诸如支持以使用处理器102的本机ISA的一个或多个ABI为目标的应用二进制的执行的库。如源于开发环境107的箭头所示,在一些实施例中,库106a包括对现任本机ABI 107a(以下称为现任ABI 107a)和EC本机ABI 107b(以下称为EC ABI 107b)的支持,这将在后面更详细地描述。OS 106还包括用于将二进制图像加载到系统存储器104中的加载器106c,该加载器106c知道现任ABI 107a和EC ABI 107b。因此,基于拥有库106a和加载器106c,OS 106支持本机二进制(例如,应用、模块、库等)的消耗,这些二进制包括以现任ABI 107a或EC ABI 107b中的一个或两个为目标的代码。

[0034] 在实施例中,OS 106还包括仿真器106b,如源于开发环境107的箭头所示,其通过仿真支持以外部ABI 107c为目标(例如,基于外部ISA)的二进制的执行。在实施例中,仿真器106b被配置为与库106a(包括提供EC ABI 107b的库)互操作。因此,基于拥有仿真器106b,OS 106支持包括以外部ABI 107c为目标的代码的外部二进制(例如,应用、模块、库等)的执行。

[0035] 如本文所用,术语“仿真”可以包括翻译和/或解释。例如,通过提前(AOT)翻译,外部ISA指令被翻译成本机ISA指令并被持久化到存储器(例如,持久存储器103)中;然后在运行时需要时调用这些翻译的指令。通常,AOT翻译发生在请求执行程序之前,例如安装应用时。在另一示例中,使用即时(JIT)翻译,当请求执行外部ISA指令时(例如,当用户运行程序时),外部ISA指令被翻译为本机ISA指令。使用JIT翻译,翻译后的本机代码一旦从外部代码翻译而来,就会立即执行。在实施例中,JIT翻译是分片进行的,因为“发现”了更多的外部代码以供执行。在实施例中,同一外部ISA代码块的JIT翻译仅进行一次,因此如果代码被执行一次以上,则翻译成本仅产生一次。在另一个示例中,使用解释,当需要执行时,读取外部ISA指令,并且由解释器执行等效函数,但不生成相应的本机ISA代码。由于外部代码不是在解释的情况下翻译的,如果同一外部ISA函数被执行两次,那么解释的成本将产生两次。

[0036] 开发环境107支持至少以EC ABI 107b为目标的二进制的创建,但通常也以现任ABI 107a为目标(即,作为双架构或“混合”二进制图像)。在实施例中,开发环境107甚至支持进一步以外部ABI 107c为目标的二进制的创建(并且因此可以在外部ABI 107c为本机的计算机系统上直接执行)。在实施例中,现任ABI 107a是以处理器102的本机ISA为目标的“遗留”ABI,而外部ABI 107c是以某些外部ISA为目标的ABI。另一方面,EC ABI 107b是处理器102的本机ABI,但其定义了镜像或至少与外部ABI 107c共享某些属性的上下文数据和CC。因此,在实施例中,现任ABI 107a可以被视为定义了完全“本机”的数据结构和动作,而EC ABI 107b虽然也是本机的,但可以被看作与外部ABI 107c共享至少一些数据结构属性和动作。在一个示例中,现任ABI 107a是以AArch64 ISA为目标的ARM64 ABI,外部ABI 107c是以x86-64 ISA为目标的Windows-X64 ABI并且EC ABI 107b以AArch64 ISA为目标,但包括至少部分镜像Windows-X64 ABI的上下文数据和CC。在实施例中,EC ABI 107b定义了镜像或映射到外部ABI 107c的足够的上下文数据和CC,以使仿真的外部代码和EC本机代码在非常低的水平上交互,但是也与现任ABI 107a具有足够的共同点(例如一组可用寄存器)以

导致将源函数的至少一部分编译到现任ABI 107a和EC ABI 107b中的每一个以产生相同的编译函数体。当编译函数体相同时,它们可以被认为是“可折叠的”,从而在生成的二进制中实际上只包含一个函数体,并且该单个函数体被以现任ABI 107a为目标的码流和以EC ABI 107b为目标的码流使用(例如,通过指针混叠)。

[0037] 为了进一步演示EC ABI 107b的实施例,图2示出了EC本机ABI的示例200,其在保持与现任本机ABI兼容的同时定义了镜像外部ABI的上下文和CC。起初,图2显示了图1中现任ABI 107a、EC ABI 107b和外部ABI 107c的表示。如图所示,现任ABI 107a定义了一组九个可用寄存器201a,包括寄存器N:A到N:I,在示例200中这些寄存器被假设为本机ISA中的所有可用寄存器。另一方面,外部ABI 107c定义了包括寄存器F:V到F:Z的一组五个可用寄存器201c,在示例200中,这些寄存器被假设 of 外部ISA中的所有可用寄存器。作为类比,AArch64 ISA(例如,“本机”ISA)定义了31个通用64位寄存器,而x86-64 ISA(例如“外部”ISA)则定义了16个通用64位寄存器。EC ABI 107b通过定义包括寄存器N:A、N:C、N:D、N:E和N:G的一组五个可用寄存器201b来桥接现任ABI 107a和外部ABI 107c。因此,即使本机ISA具有九个可用寄存器,在示例200中,EC ABI 107b被限制为仅使用这些寄存器中的五个,这是与外部ISA中可用寄存器的数目对应的数目。如本领域普通技术人员将理解的,即使以EC ABI 107b为本机代码可不使用由本机ISA定义的所有寄存器,该本机代码仍然可以在本机ISA上执行。因此,EC ABI 107b可以以与现任ABI 107a兼容的方式定义可用寄存器201b的使用,使得以EC ABI 107b为本机代码也可以在仅支持现任ABI 107a的系统下执行。

[0038] 在实施例中,该组可用寄存器201b被选择为使用针对现任ABI 107a编译的代码最常用的寄存器。因此,即使在示例200中,该组可用寄存器201b使用的寄存器少于该组可用寄存器集202a中的所有寄存器,但在以该组可用寄存器201a为目标时编译的代码可以经常仅实际使用从该组可用寄存器201b中选择的寄存器。在这些情况下,针对现任ABI 107a和EC ABI 107b中的每个被编译的代码是相同的,并且可以在生成的二进制中折叠。值得注意的是,尽管在示例200中,EC ABI 107b的该组可用寄存器201b包括的寄存器少于现任ABI 107a的该组可用寄存器201a的所有寄存器,但是备选示例可以定义该组可用寄存器201b以使用该组可用寄存器201a的所有寄存器(例如,如果外部ISA具有与本机ISA匹配的寄存器的数目)。

[0039] 此外,图2中的箭头表明,在EC ABI 107b的可用寄存器201b和外部ABI 107c的可用寄存器201c之间定义了映射。因此,例如,EC ABI 107b以镜像外部ABI 107c对寄存器F:V的使用的方式使用寄存器N:A,EC ABI 107b以镜像外部ABI 107c对寄存器F:W的使用的方式使用寄存器N:C,依此类推。在一个示例中,EC ABI 107b定义可用寄存器201b的使用,其镜像外部ABI 107c使用的CC。例如,为了实现外部ABI 107c的动作,EC ABI 107b可以限制可用于向函数传递参数的寄存器的数目(例如,与现任ABI 107a相比)。作为类比,X64 ABI仅使用四个寄存器来传递参数,因此EC ABI 107b可以仅允许四个寄存器用于传递参数,即使现任ARM64 ABI被定义为使用额外的寄存器用于参数传递目的。此外,EC ABI 107b可以镜像外部ABI 107c的堆栈使用(例如,用于保存返回地址、用于从函数返回值等),而不是为此目的使用寄存器(如现任ABI 107a的情况)。在实施例中,寄存器之间的映射包括寄存器之间波动率的映射。因此,例如,如果本机ISA中在现任ABI 107a下被视为非易失性的寄存

器被映射到被外部ABI 107c视为易失性寄存器,则EC ABI 107b可以认为该寄存器是易失性。

[0040] 图2还示出了现任ABI 107a定义上下文数据202a,EC ABI 107b定义上下文数据202b,而外部ABI 107c定义上下文数据202c。在实施例中,上下文数据是由ABI定义的格式,用于存储处理器上下文的快照,例如寄存器和标志。此上下文数据可用于各种目的,例如用于应用观察其自身状态(例如,作为复制保护方案的一部分,作为病毒或恶意软件检测的一部分等),用于异常展开例程以在异常后展开堆栈,或促进线程挂起/恢复。如上下文数据202a、上下文数据202b和上下文数据202c中的每个上下文数据的可视大小所示,在实施例中,EC ABI 107b定义了不超过外部ABI 107c使用的上下文数据202c的大小的上下文数据格式202b(例如,可用寄存器201b的选择和排列),即使现任ABI 107a可以定义超过这个大小的上下文数据202a。通过将上下文数据202b定义为不超过上下文数据202c的大小,实现了EC ABI 107b和外部ABI 107c之间的互操作性。例如,如果在外部ABI 107c下执行的应用分配用于存储上下文数据202c的格式的存储器缓冲器,并且该存储器缓冲器由EC ABI 107b使用上下文数据202b的格式来填充(例如,作为处理异常的一部分),则由EC ABI 107b写入的上下文数据202b不超过所分配的存储器缓冲器。

[0041] 在一些实施例中,上下文数据202b被定义为具有上下文数据202a和上下文数据202c的混合的格式。例如,上下文数据202a的格式用前对角线可视化地表示,而上下文数据202c的格式用后对角线可视化地表示。然后,上下文数据202b的格式用正向对角线和反向对角线可视化地表示,表示格式的至少部分混合。例如,即使上下文数据202b定义了本机ISA的寄存器的排列,它也可以以外部ISA所期望的方式(例如,基于可用寄存器201b和可用寄存器201c之间所示的映射)排列这些寄存器。

[0042] 值得注意的是,当定义EC ABI 107b(包括可用寄存器集201b、上下文数据202b中的一个或多个,或可用寄存器201b和可用寄存器201c之间的映射)时,在将EC ABI 107b定义为与现任ABI 107a更相似与将EC ABI 107b定义为与外部ABI 107c更相似之间可存在权衡。例如,EC ABI 107b与现任ABI 107a的相似度越高,编译到EC ABI 107b的代码就越有可能与编译到本机ABI 107a的代码“折叠”;然而,这也可能使得当在EC ABI 107b和外部ABI 107c之间翻译时更需要采用ABI翻译(经由稍后讨论的入口和/或出口形实转换程序)。相反,EC ABI 107b与外部ABI 107c的相似度越高,则EC ABI 107b中的本机代码和外部ABI 107c中的外部代码就越有可能在不使用ABI翻译的情况下进行交互(形实转换程序);然而,这也可能使编译到EC ABI 107b的代码不能与编译到本机ABI 107a的代码“折叠”。

[0043] 回到图1,在一些实施例中,开发环境107支持创建以两个或多个ABI为目标的“双架构”混合二进制。例如,如各种箭头所示,图1显示了编译器工具链107d消耗源代码110以生成混合二进制108,该混合二进制108包括以现任ABI 107a(即,现任本机代码108a)和EC ABI 107b(即,EC本机代码108b)为目标的代码。因此,当仅包含本机代码时,混合二进制108可以被视为以现任本机动作(即,现任本机代码108a)为目标并且以外部动作(即EC本机代码108b)为目标。尽管未示出,但在一些实施例中,混合二进制还包括以一个或多个附加ABI(例如外部ABI 107c)为目标的代码。

[0044] 在图1中,现任本机代码108a和EC本机代码108b之间有一个宽箭头,指示现任本机代码108a和EC本机代码108b至少部分“折叠”在一起(即,现任本机代码108a和EC本机代码

108b都使用指令集)。如所讨论的,即使EC ABI 107b以至少一些外部动作为目标,它与现任ABI 107a具有足够的共同点,以导致将源函数的至少一部分编译到现任ABI 107a和EC ABI 107b中的每一个,从而产生相同的编译函数体。在实施例中,当将函数编译到现任ABI 107a和EC ABI 107b中的每一个导致相同的编译代码时,编译器工具链107d将该函数“折叠”到混合二进制108中,仅发出该函数的单个编译版本,供现任本机代码108a和EC本机代码108b使用。例如,即使编译器工具链107d为函数生成编译代码的现任本机“版本”和编译代码的EC本机“版本”,当这些“版本”匹配(即,具有相同的本机代码)时,编译器工具链107d也仅将这些版本中的一个版本发出到混合二进制108中。在测试中,发明人已经观察到,由于代码折叠,以这种方式支持双ABI(即,现任ABI 107a和EC ABI 107b)仅导致二进制大小比以单个ABI为目标的二进制大小增加10-30%。这与传统的胖二进制形成了鲜明对比,胖二进制缺乏折叠代码的能力,二进制的大小将增加接近100%。

[0045] 在实施例中,当编译源代码110以以EC ABI 107b为目标时,编译器工具链107d遵循诸如预处理器指令的源代码定义,就好像编译器工具链107d以外部ABI 107c的ISA为目标一样(即使编译器工具链107d正在生成本机代码)。这是因为EC ABI 107b表现出外部ABI 107c的动作。例如,以下函数根据目标ABI遵循不同的逻辑路径:

```
int function1 (int x) {  
    #if defined (_ARM64_)  
        return x + 10;  
[0046]    #else  
        return x - 2;  
    #endif  
}
```

[0047] 特别地,预处理器指令定义,当以ARM64 ISA为目标时,值10应加到“x”上,而当不以ARM64 ISA为目标时,值2应从“x”中减去。在实施例中,当使用ARM64 ISA编译该函数以以现任ABI 107a为目标时,编译器工具链107d生成将值10添加到“x”(即,ARM64逻辑路径)的指令;相反,当编译该函数以以EC ABI 107b为目标时,编译器工具链107d生成从“x”(即,非ARM64逻辑路径)减去值2的指令。

[0048] 在实施例中,混合二进制108由编译器工具链107d配置为由遗留加载器(即,知道现任ABI 107a但不知道EC ABI 107b)本机解析和利用,但由开放加载器(即,知道EC ABI 107b)使用。在实施例中,混合二进制108使用遗留加载器所期望的布局/格式,因此“默认”在现任ABI 107a下执行。然而,混合二进制108还包括附加信息(例如使得混合二进制108也能够被EC ABI 107b消耗的修正表108c)。

[0049] 图3A示出了“双架构”混合二进制的示例300a。更具体地,图3A示出了图1中混合二进制108的更详细的表示,示出了两个相互关联的码流——码流301a和码流301b。在实施例中,码流301a对应于现任本机代码108a,码流301b对应于EC本机代码108b。因此,混合二进制108以两个不同的本机ABI(即,现任ABI 107a和EC ABI 107b)以及它们的不同动作(即,

完全本机动作和外部动作)为目标。如码流301a和码流301b之间的箭头所示,在一些函数之间存在函数的折叠。特别地,码流301a和码流301b中的每个都包括被描绘为具有实线的框的“非折叠”函数,指示这些函数的不同编译版本存在于码流301a与码流301b的每一个中。然而,码流301a和码流301b中的每一个还包括“折叠”函数,其被描绘为具有虚线并由箭头连接的框,指示每个函数仅存在一个版本供码流301a与码流301b两者使用(例如,通过混叠)。图3A显示了混合二进制108可以包括附加码流301c,例如外部码流。然而,更典型地,在实施例中,混合二进制108与包括外部码流302的外部二进制109(例如,对应于遗留插件/库)交互。

[0050] 如上所述,在实施例中,混合二进制108被配置为由现任ABI 107a本机执行。例如,混合二进制108中的机器标识符303字段标识现任ABI 107a所期望的机器类型,入口点304指定码流301a的入口点,并且任何导入和/或导出表(输入/导出表305)提供公开与现任ABI 107相关的函数(及其位置)的“本机”视图。此外,调用非折叠函数的任何“折叠”函数被配置为调用码流301a中的函数的“现任”版本,而不是码流301b中的“EC”版本。这样,为了执行码流301a,遗留加载器只需要像加载任何其他兼容二进制一样加载混合二进制108。

[0051] 为了便于由开放的加载器(例如,加载器106c)加载,图1和图3A示出了混合二进制108包括修正表108c。在实施例中,修正表108c指定当混合二进制108被加载到仿真进程111中时,加载器106c将要应用于从混合二进制108加载的存储器的一个或多个存储器变换(例如,可以在处理器102处本机执行EC本机代码108b的兼容进程,以及经由仿真器106b执行诸如外部二进制109的外部代码)。

[0052] 在实施例中,加载器106c在识别出混合二进制108中的机器标识符303对于EC ABI 107b不合适时定位修正表108c。然后,加载器106c将在修正表108c中指定的一个或多个变换应用于系统存储器104的包含从混合二进制108加载的存储器页的部分,以便执行码流301b而不是码流301a。在实施例中,在修正表108c中指定的每个变换标识存储器位置(例如,通过相对地址、通过绝对地址等),连同要在该存储器位置应用的变换(例如,以在存储器位置处替换一个或多个字节,以在存储器位置处应用算术运算等)。在实施例中,修正表108c包括一个或多个修正以调整机器标识符303以匹配加载了混合二进制文件108的进程(例如,仿真进程111)的ISA。在实施例中,这是与外部ABI 107c对应的外部ISA。在实施例中,修正表108c包括一个或多个修正以调整入口点304以指定代码流301b的入口点。在实施例中,修正表108c包括一个或多个修正以使折叠函数调用码流301b中的非折叠函数的EC版本而不是码流301a中的函数的现任版本。在实施例中,修正表108c包括一个或多个修正,这些修正使导入/导出表305提供“兼容性”视图,以公开与EC ABI 107b相关的函数(及其位置)。

[0053] 在一些实施例中,使折叠函数调用码流301b中的非折叠函数的EC版本而不是码流301a中的函数的现任版本包括修补折叠函数本身中的调用。然而,在其他实施例中,折叠函数被配置为经由分派表306间接调用非折叠函数。在这些实施例中,修正表108c包括到该分派表306的一个或多个修正,其用到码流301b中的函数的EC版本的地址或偏移替换到码流301a中的函数的当前版本的地址或者偏移。在实施例中,分派表306的使用可以通过限制为了执行码流301b而不是码流301a而需要应用存储器修正的存储器页的数目来提供效率。

[0054] 在一些实施例中,使导入/导出表305提供与EC ABI 107b相关的公开函数(及其位

置)的“兼容性”视图包括修补对导入/导出表305的参考以公开导入/导出列表305的不同部分(窗口)。为了说明这个概念,图3B示出了导入/导出表的窗口视图的示例300b。特别地,图3B示出了导入/导出表305,该表将多个函数与其对应的存储器位置(例如,地址或偏移)进行映射。在示例300b中,导入/导出表305包括第一阴影部分/区域,其包括仅可应用于码流301a(例如,作为非折叠函数)的函数(即,A、B和C),非阴影部分/区域,其包括可应用于码流301a和码流301b(例如,作为折叠函数)的函数(即,D和E),以及第二阴影部分/区域,其包括仅可应用于码流301b(例如,作为非折叠函数)的函数(即,A'、B'和F)。括号描绘了导入/导出表305的本机视图307a和导入/导出表305的兼容性视图307b。在实施例中,本机视图被指定为对导入/导出表305的第一入口和大小/计数(例如,为5)的基本参考,其包括函数A、B和C(仅适用于码流301a)以及函数D和E(适用于两个码流)。在实施例中,兼容性视图被指定为对导入/导出表305的第四入口和大小/计数(例如,为5)的基本参考,其包括函数D和E(适用于两个码流)以及函数A'、B'和F(仅适用于码流301b)。因此,在实施例中,修补对导入/导出表305的基本参考以公开导入/导出表305的不同窗口包括修补基本参考或大小/计数中的一个或两个。

[0055] 如上所述,EC ABI 107b促进了与外部ABI 107c的互操作性,例如在EC ABI 107 b下本机执行的应用的上下文中执行遗留插件/库(通过仿真)。这包括使EC本机代码108b中的函数能够调用外部代码中的函数(例如,外部二进制109中的码流302),以及使外部代码中的函数能够调用EC本机代码108b中的函数。由于外部代码在仿真器106b下执行,所以对于这些跨ABI函数调用中的每一个跨ABI函数调用,都存在到仿真器106b的变换或从仿真器106b出来的变换。在实施例中,混合二进制108可以包括用于从外部ABI 107c变换到EC ABI 107b的入口形实转换程序(即,当外部二进制109中的代码调用EC本机代码108b中的函数时),以及用于从EC ABI 107b变换到外部ABI 107c的出口形实转换程序(即,当EC本机代码108b中的代码调用外部二进制109中的函数时)。在一些实施例中,EC本机代码108b中可以由外部代码调用的每个函数包括一个入口形实转换程序(其大小可以为零)和用于对不同外部函数的每个调用的出口形实转换程序。在实施例中,形实转换程序例如通过使EC ABI 107b的CC适应于外部ABI 107c的CC(反之亦然)来适应EC ABI 107b和外部ABI 107c之间的差异。

[0056] 在实施例中,每个入口形实转换程序确保由外部函数传递给EC本机函数的参数位于适当的位置以供EC本机函数使用。这可以包括例如将值从一个寄存器移动到另一个寄存器、将值从堆栈位置移动到寄存器、将值从寄存器移动到堆栈位置等。在一些实施例中,EC ABI 107b可以这样定义,即由外部函数传递给EC本机函数的参数已经位于供EC本机函数使用的适当位置。在这些情况下,入口形实转换程序的大小可以为零(因此不执行任何操作或被省略)。在一些实现中,如果传送给EC本机函数的参数的数目低于阈值,则可能出现这种情况。例如,发明人已经观察到,对于AArch64本机ISA和x86_64外部ISA,可以以以下的方式定义EC ABI 107b(包括AArch64与x86_64寄存器之间的映射):如果从外部函数传递到EC本机函数的参数少于或等于四个,并且所有这些参数都是基本整数,则使具有大小为零的入口形实转换程序成为可能。

[0057] 在实施例中,出口形实转换程序用于由EC本机函数对外部函数的每次调用,并且因此单个EC本机函数可以具有零个或多个出口形实转换程序(取决于EC本机函数调用多少

不同的外部函数)。在实施例中,每个出口形实转换程序执行以下一项或多项:(i) 将函数返回地址保存到适当的位置(例如,堆栈或寄存器)以返回到EC本机函数,(ii) 确保从EC本机函数传送到被调用的外部函数的参数位于适当的位置以供外部函数使用,或(iii) 在仿真器106b内发起外部函数的执行。

[0058] 图4A和图4B说明了本机函数之间以及本机函数和外部函数之间的调用的示例400a和示例400b。特别地,图4A说明了编译为以现任ABI 107a为目标的现任本机函数403的调用动作,以及编译为以EC ABI 107b为目标的EC本机函数404的调用动作。在实施例中,现任本机函数403和EC本机函数404是同一源代码函数的不同编译版本,但由于所得到的编译代码之间的不匹配而没有折叠在一起。在实施例中,这种不匹配可能出现,因为现任本机函数403使用在现任ABI 107a下可用的一个或多个寄存器(但不在EC ABI 107b下),而EC本机函数403仅使用映射到外部ABI 107c的缩减的寄存器集(如结合图3A所讨论的)。备选地,这种不匹配可能出现,因为当以现任ABI 107a和EC ABI 107b为目标时,条件代码编译强制代码逻辑分歧。例如,如所讨论的,在实施例中,当编译同一函数以以现任ABI 107a和EC ABI 107b中的每一个为目标时,编译器工具链107d遵循不同的预处理器指令(并且因此遵循不同的逻辑路径)。特别地,编译器工具链107d在以现任ABI 107a为目标时采用“本机ISA”逻辑路径,但在以EC ABI 107b为目标时采用“外部ISA”逻辑路径。

[0059] 在图4A中,路径#1显示使用本机CC(图中的“本机CC”)调用现任本机函数403。此外,路径#1显示现任本机函数403可以使用本机CC进行调用。另一方面,路径#2显示通过入口形实转换程序401使用外部CC(图中的外部CC)间接调用EC本机函数404,该入口形实转换程序401适应外部CC和本机CC之间的任何差异,然后调用EC本机函数404。此外,路径#2显示EC本机函数404可以通过出口形实转换程序402间接调用外部函数,该出口形实转换程序402适应本机CC和外部CC之间的任何差异,然后调用外部函数。路径#3显示使用外部CC调用(例如,通过出口形实转换程序402或另一外部函数)外部函数405(例如,在码流301c和/或码流302内)形实转换程序外部。

[0060] 图4B说明了EC本机函数406的调用动作,该函数已被折叠,因为将源代码函数编译到现任ABI 107a和EC ABI 107b时产生了相同的代码。在图4B中,路径#1显示可以使用本机CC调用EC本机函数406,而路径#2显示也可以通过入口形实转换程序401使用外部CC间接调用EC本机函数406。此外,图4B显示“EC本机函数404可以使用本机CC调用其他本机函数或通过出口形实转换程序402调用外部函数。路径#3显示使用外部CC调用外部函数405(例如,由出口形实转换程序402或另一外部函数)。

[0061] 在实施例中,计算机系统101使函数调用站点能够一致地调用/参考被调用方函数的真实存储器地址,即使在桥接本机代码和外部代码(即,在ABI 107b和外部ABI 107c之间)时也是如此。值得注意的是,由于使用入口和出口形实转换程序来桥接ABI变换,当桥接本机代码和外部代码时,在调用站点使用真实的存储器地址是一个挑战。这与桥接本机代码和外部代码的现任解决方案形成对比,其中两个本机函数内的调用站点实际上可使用不同的地址来调用同一个外部函数。这是因为,在这些先前的解决方案中,调用站点实际上调用的是形实转换程序的地址,而不是外部函数的真实地址。如果本机代码中的程序逻辑依赖于地址比较(例如,比较由第一本机函数获得的指向外部函数的第一指针与由第二本机函数获得的指向外部函数的第二指针),如果指针从本机代码传送到外部代码(其中它对外

部代码无效/不可用),或者如果指针从外部代码传送到本地代码(对于本地代码无效/不可用),则此动作可以引入兼容性问题。使用本文描述的解决方案,当获得(或“获取”)第一外部函数的存储器地址(例如,在外部二进制109内)时,实施例确保EC本机代码108b中的第一本机函数和EC本机代码108中的第二本机函数都为第一外部函数获得相同的存储器地址,该存储器地址是第一外部函数开始的真实存储器地址。此外,实施例还确保第二外部函数(例如,在外部二进制109内)也获得第一外部函数的相同真实存储器地址。实施例还确保本机函数的地址是相同的,而不管该地址是由外部函数还是由另一个本机函数获取。

[0062] 在实施例中,一致的存储器地址参考由混合二进制108内的EC查找结构307(例如,由编译器工具链107d发出到混合二进制108中)与分派器112(即,库106a中的分派器112a和仿真器106b中的分派器112b)一起启用。在实施例中,EC查找结构307是可用于确定由混合二进制108定义的存储器映像的存储器地址的哪个范围包含EC本机代码108b的任何类型的结构。在实施例中,EC查找结构307是位图,其使用一位值来指示存储器的对应范围(例如,存储器页)是否损害EC本机代码108b。然而,EC查找结构307可以包括备选数据结构类型,例如哈希表或二叉树。

[0063] 在实施例中,当从调用方函数向被调用方函数进行调用时,分派器112使用EC查找结构307来确定被调用方函数的目标存储器地址是否在EC本机代码内。然后,利用调用方函数是本机代码还是外部代码的固有知识,分派器112适当地分派调用。在实施例中,分派器112在至少四种场景中操作:本机调用方和本机被调用方、本机调用方和外部被调用方、外部调用方与外部被调用方、以及外部调用方与本机被调用方。

[0064] 在第一种情况下,调用方函数是在EC ABI 107b下执行的本机函数,因此调用由库106a中的分派器112a处理。分派器112a使用EC查找结构307来确定被调用方的参考存储器地址在与EC本机代码108b对应的存储器区域内,并且因此被调用方函数也是在EC ABI 107b下执行的本机函数。在这种情况下,调用方正在调用被调用方函数的真实存储器地址,并且不需要形实转换程序,因此分派器112a使用参考存储器地址直接调用被调用方函数。

[0065] 在第二种情况下,调用方函数是在EC ABI 107b下执行的本机函数,因此调用由库106a中的分派器112a处理。分派器112a使用EC查找结构307来确定被调用方的参考存储器地址不在对应于EC本机代码108b的存储器区域内,并且因此被调用方函数是在外部ABI 107c下执行的外部函数。参考图4A,这是路径#2的情况,其中EC本机函数404通过出口形实转换程序402调用外部函数。这样,分派器112a不能直接调用(即,被调用方函数的)参考存储器地址,因为这将绕过出口形实转换程序402。相反,分派器112a定位到出口形实转换程序402的新参考地址,并使用新参考存储器地址调用出口形实转换程序402。在实施例中,出口形实转换程序402继而将EC ABI 107b的CC适应为外部ABI 107c的CC,然后调用仿真器106b。仿真器106b进而使用原始参考存储器地址直接调用被调用方函数。

[0066] 在实施例中,出口形实转换程序的新参考地址包含在被调用方函数本身内。在这些实施例中,调用站点与两个参考存储器地址相关联:被调用方函数的原始参考存储器地址和到出口形实转换程序的新参考地址。当在现任ABI 107a下加载混合二进制108时,现任ABI 107使用的分派器直接使用被调用方函数的原始参考存储器地址,而忽略到出口形实转换程序的新参考地址。值得注意的是,现任ABI 107a可以忽略到出口形实转换程序的新参考地址,因为它不与外部代码交互。另一方面,当混合二进制108被加载到EC ABI 107b下

时,分派器112a还利用到出口形实转换程序的新参考地址来与外部代码交互。值得注意的是,编译器工具链107d通过在编译代码中包括原始参考存储器地址和新的参考地址来促进可折叠性,而不管函数是以现任ABI 107a还是EC ABI 107b为目标。

[0067] 在第三种情况下,调用方函数是在仿真器106b内的外部ABI 107c下执行的外部函数,因此调用由仿真器106b中的分派器112b处理。分派器112b使用EC查找结构307来确定被调用方的参考存储器地址不在与EC本机代码108b对应的存储器区域内,并且因此被调用方函数也是在外部ABI 107c下执行的外部函数。在这种情况下,调用方正在调用被调用方函数的真实存储器地址,并且不需要形实转换程序,因此分派器112b使用仿真器106b内的参考存储器地址直接调用被调用方函数。

[0068] 在第四种情形中,调用方函数是在仿真器106b内的外部ABI107c下执行的外部函数,因此调用由仿真器106b中的分派器112b处理。分派器112b使用EC查找结构307来确定被调用方的参考存储器地址在与EC本机代码108b对应的存储器区域内,并且因此被调用方函数是在EC ABI 107b下执行的本机函数。参考图4A,这是路径#2的情况,其中外部函数通过入口形实转换程序401调用EC本机函数404。这样,分派器112b不能直接调用(即,被调用方函数的)参考存储器地址,因为这将绕过入口形实转换程序401。相反,分派器112b定位到入口形实转换程序401的新参考地址,并使用新的参考存储器地址调用入口形实转换程序401。在实施例中,入口形实转换程序401进而将外部ABI 107c的CC适应为EC ABI 107b的CC,然后使用原始参考存储器地址调用被调用方函数。

[0069] 值得注意的是,在第四种情况下,调用方函数可以是没有意识到或与EC ABI 107b兼容的遗留外部代码。因此,与第二种情况不同,调用方函数不能被修改(例如,通过编译器工具链)以包含入口形实转换程序的地址。在实施例中,分派器112b从紧接在被调用方函数的原始参考存储器地址之前的存储器块获得入口形实转换程序的新参考地址。在实施例中,用于获得入口形实转换程序的新参考地址的数据在生成混合二进制108期间由编译器工具链107d插入到该存储器块中。包含在该存储器块中的特定数据可以变化,但在一些实施例中,它是存储器偏移(例如,从由混合二进制108定义的存储器图像的开头)。在其他实施例中,它可以是对入口形实转换程序的直接地址参考。在其他实施例中,分派器112b以某种备选方式(例如从数据树或地址的有序阵列)获得新的参考地址。

[0070] 在实施例中,EC ABI 107b与编译器工具链107d和分派器112一起支持“兼容模式”,该模式使得调用本机函数的外部代码能够成功地标识并潜在地修补在本机函数开头插入的外部代码。这使得外部代码能够反汇编被调用的函数的开头(例如,作为防病毒保护、复制保护等的一部分)和/或修补与调用该函数相关的被调用的函数的开头(例如作为分析重定向的一部分)。由于可以没有意识到正在被仿真的外部代码可以期望在被调用的函数的开头处找到可识别的外部代码,在一些实施例中,编译器工具链107d向混合二进制108发出用于本机函数的“快进序列”(形实转换程序的形式),外部代码可以在该序列上进行反汇编和/或修补。在实施例中,“快进序列”包括以参考或跳转到本机函数的真实地址结尾的占位符外部代码。在实施例中,该快进序列被存储在EC查找结构307中标记为不包括EC本机代码的存储器页中。因此,在实施例中,当外部调用方调用具有快进序列的本机函数时,该调用最初被视为外部到外部调用(即,如在上述第三场景中),使得仿真器106b执行快进序列并在快进序列结束时(即,如上所述第四场景中)进行外部到本机调用。

[0071] 值得注意的是,这种布置可导致双重形实转换程序性能问题,其中仿真器106b仅被调用以仿真快进形实转换程序中的一些指令(例如,如第三场景中那样),然后仅发起对另一形实转换程序的调用,即对本机函数的入口形实转换程序(例如,第四场景中那样)。一些实施例通过配置分派器112b(在仿真器106b中)以在确定被调用方的参考存储器地址不在对应于EC本机代码108b的存储器区域内(即,使用EC查找结构307)之后执行附加检查,来避免这种双重形实转换程序性能问题。这个附加检查是在被调用的函数的开头“读取数据”,以确定是否存在快进序列的签名。如果是,则分派器112b从快进序列中获得本机函数的真实地址,并直接发起对本机函数的调用(即,如上述第四种情形中那样)。通过进行该附加检查,已经避免了仿真器106b对快进形实转换程序的处理。

[0072] 鉴于前面的介绍,下面的讨论现在涉及一些方法和方法动作。尽管方法动作可以以特定顺序被讨论,或者可以在流程图中说明为以特定顺序发生,但除非特别说明,否则不需要特定顺序,或者因为一个动作依赖于在执行该动作之前完成的另一个动作。

[0073] 用于支持外部代码的仿真的本机仿真兼容的应用二进制接口

[0074] 图5示出了一种示例方法500的流程图,该方法用于将源代码编译为以本机第一ISA为目标的二进制文件,同时可以与编译为外部第二ISA的代码进行操作。最初,方法500包括标识针对第一ISA的第一ABI和第二ABI以及针对第二ISA的第三ABI的动作501,第二ABI定义不超过第三ABI的上下文数据大小的上下文数据,并且定义映射到第三ABI的寄存器子集的使用。在实施例中,动作501包括标识(i)与第一ISA对应的第一ABI,(ii)与第一ISA对应的第二ABI,以及(iii)与第二ISA对应的第三ABI。在示例中,编译器工具链107d标识现任ABI 107a(即,第一ABI)、EC ABI 107b(即,第二ABI)和外部ABI(即,第三ABI)中的每一个。

[0075] 在这些实施例中,第二ABI定义第一上下文数据格式,其具有不超过第三ABI所使用的第二上下文数据格式的第二大小的第一大小。例如,参考图3A,EC ABI 107b定义了上下文数据202b的格式,其大小不超过上下文数据202c的大小(由外部ABI 107c定义)。

[0076] 在这些实施例中,第二ABI还定义了第一ISA的多个寄存器中的寄存器子集的使用,这些寄存器由第一ABI使用。在一些实施例中,寄存器子集包括少于第一ISA的多个寄存器中的所有寄存器(例如,如在示例200中),尽管在其他实施例中寄存器子集包括第一ISA的所有多个寄存器。在实施例中,该寄存器子集被映射到由第三ABI使用的第二ISA的一组寄存器。在实施例中,第二ABI仅允许使用寄存器子集,排除第一ABI允许的一个或多个其他寄存器。再次参考图3A,EC ABI 107b定义了可用寄存器201b,它们是现任ABI 107a使用的可用寄存器201a的子集(即,它们省略了现任ABI 107a使用的一个或多个可用寄存器201a)。此外,可用寄存器201b由EC ABI 107b映射到外部ABI 107c使用的该组可用寄存器201c。

[0077] 在实施例中,当使用第一ABI和第二ABI中的每一个被编译时,第二ABI使用寄存器子集导致函数的至少一部分是可折叠的。例如,在实施例中,可用寄存器201b的选择足够紧密地反映可用寄存器201a,使得在至少一些情况下,针对现任ABI 107a和EC ABI 107b中的每一个编译的函数产生相同的编译代码。在实施例中,第二ABI对寄存器子集的使用还使得编译成以第二ABI为目标的函数的至少一部分能够在没有入口形实转换程序的情况下从第三ABI调用。例如,如果第一ABI是AArch64 ABI,而第三ABI是Windows-X64 ABI的话,那么可

以以省略接收四个或更少输入参数的本机函数的入口形实转换程序的方式来定义第二ABI。

[0078] 方法500还包括将函数发出到二进制中的动作502。如图所示,动作502包括在以第一ABI为目标以创建第一编译版本的同时编译函数的动作503,以及在以第二ABI为目标以创建第二编译版本的同时编译函数的动作504。如图所示,在动作503和动作504之间没有示出特定的顺序,并且应当理解,这些动作可以并行或串行(以任一顺序)执行。在实施例中,对于源代码中定义的函数,动作503包括生成以第二ABI为目标的函数的第二编译版本,包括在第一ISA中生成第二指令集,而动作504包括生成以第一ABI为目标的函数的第一编译版本,包括在第一ISA内生成第一指令集。在示例中,编译器工具链107d在以第一ABI为目标时编译源代码函数两次,在以第二ABI为目标时编译一次,从而产生该函数的两个编译版本。如所讨论的,在实施例中,编译器工具链107d在以现任ABI 107a为目标时采用“本机ISA”逻辑路径,而在以EC ABI 107b为目标时采用“外部ISA”逻辑路径。因此,在一些实施例中,在动作502中生成以第一ABI为目标的函数的第一编译版本包括使用第一ISA的源代码定义,并且在动作503中生成以第二ABI为目标的函数的第二编译版本包括使用第二ISA的源码定义。

[0079] 动作502还包括确定第一编译版本和第二编译版本是否匹配并且因此是可折叠的动作505。在实施例中,动作505包括至少基于确定第一指令集和第二指令集是否匹配,确定该函数的第一编译版本和该函数的第二编译版本是否可在二进制文件内折叠。在示例中,编译器工具链107d在动作503和动作504中产生的函数的两个编译版本之间执行比较。如果它们匹配,则该函数的两个编译版本是可折叠的;否则,它们不可折叠。

[0080] 取决于动作505的结果,动作502包括将两个编译的版本发出到二进制的动作506(当编译的函数不可折叠时),或(当编译的函数可折叠时)将仅一个编译的版本发出到二进制的动作507。在实施例中,动作506包括,至少基于确定该函数的第一编译版本和该函数的第二编译版本是否可在二进制文件内折叠,以及当确定该函数的第编译版本和第二编译版在二进制文件中不可折叠时,将该函数的第一编译版本和该函数的第二编译版本两者发出到二进制文件中。在示例中,当编译的函数不可折叠时,编译器工具链107d将该函数的现任版本(例如,诸如现任本机函数403)和该函数的EC版本(例如,EC本机函数404)两者发出到混合二进制108中。在实施例中,动作507包括,至少基于确定该函数的第一编译版本和该函数的第二编译版本是否可在二进制文件内折叠,以及当该函数的第一编辑版本和该函数的第二编辑版本被确定为可在二进制文件内折叠时,仅将该函数的第一编译版本或该函数的第二编译版本中的一个发出到二进制文件中。在示例中,当编译的函数是可折叠的时,编译器工具链107d将该函数的单个EC版本(例如,诸如EC本机函数406)发出到混合二进制108中。

[0081] 在实施例中,第二ABI使得二进制文件能够在以下两者上本机执行:(i)实现第一ABI但不实现第二ABI的第一计算机系统和(ii)实现第二ABI的第二计算机系统。例如,使用方法500生成的混合二进制108可以在仅实现现任ABI 107a的“遗留”计算机系统上执行,或者在实现EC ABI 107b的现代计算机系统上可执行。

[0082] 在一些实施例中,方法500还包括向二进制文件中发出口形实转换程序,该入口形实转换程序包括第一ISA中的代码,该代码使第三ABI适应第二ABI。在示例中,当执行动

作506时,编译器工具链107d发出用于所发出的函数的EC版本(例如,EC本机函数404)的入口形实转换程序,使得外部代码能够调用该函数。在另一示例中,当执行动作507时,编译器工具链107d为单个发出的函数发出口形实转换程序,其可以被认为是EC函数(例如,EC本机函数406),使得外部代码能够调用该函数。

[0083] 在一些实施例中,方法500还包括将一个或多个出口形实转换程序发出到二进制文件中,每个出口形实转换程序包括第一ISA中的代码,该代码使第二ABI适应第三ABI。在示例中,当执行动作506时,编译器工具链107d针对所发出的函数的EC版本(例如,EC本机函数404)发出一个或多个出口形实转换程序,从而使该函数能够调用外部代码。在另一示例中,当执行动作507时,编译器工具链107d针对单个发出的函数发出一个或多个出口形实转换程序,从而使该函数能够调用外部代码。在一些实施例中,方法500为该函数的每个不同调用生成不同的出口形实转换程序,但是在其他实施例中方法500可以为每个被调用方生成单个出口形实转换程序。

[0084] 如上所述,混合二进制可以包括附加的外部码流,例如包括外部代码的码流301c。因此,方法500可以进一步包括生成以第三ABI为目标的函数的第三编译版本,包括在第二ISA中生成第三指令集,并将该函数的第三编译版本发出到二进制文件中。

[0085] 方法500的一些变体可以省略“现任”码流,使得发出的二进制以EC ABI 107b而不是现任ABI 107a为目标。如将认识到的,所得到的二进制提供了与以外部ABI 107c为目标的代码的兼容性,但将与实现现任ABI 107a而不是EC ABI 107b的程序缺乏向后兼容性。在实施例中,省略“现任”码流对于仅打算在不使用现任ABI 107a的应用中使用的二进制可以是有用的。作为一个示例,二进制实现函数(例如,对过时视频格式的编解码器支持)已被弃用于本机应用,但为了与仿真应用兼容而保留。

[0086] 支持码流折叠的混合二进制

[0087] 虽然图5关注使用EC ABI 107b生成代码的方法,该方法能够实现代码折叠,但是混合二进制108可以包括附加特性,使其能够在使现代系统(例如,具有EC ABI 107b)能够利用额外的特性与外部代码交互的同时,实现与遗留系统的向后兼容(例如,仅具有现任ABI 107a)。为了进一步描述这些双架构混合二进制108的创建,图6示出了用于生成混合二进制图像的示例方法600的流程图,该混合二进制图像可以在本机ABI和兼容ABI下执行。然后,为了进一步描述这些双架构混合二进制108的消耗,图7示出了用于通过在兼容ABI下执行的进程消耗混合二进制图像的示例方法700的流程图,该混合二进制图像可以在本机ABI和兼容ABI两者下执行。

[0088] 首先参考二进制图像创建,并参考图6,方法600包括多个动作(即,动作601-604),如流程图所示,这些动作可以相对于彼此以任何顺序执行。如图所示,方法600包括标识本机ABI的机器类型并将其发出到由本机ABI读取的机器类型字段中的动作601。在一些实施例中,动作601包括,至少基于标识与本机ABI对应的第一机器类型,将第一机器类型发出到混合二进制图像的机器类型字段中,该机器类型字段被构造为当在本机ABI下执行的本机进程加载混合二进制图像时使用。在示例中,编译器工具链107d将机器标识符303发出到混合二进制108中,该机器标识符303在加载以现任ABI 107a为目标的二进制时与现任ABI 107a期望的标识符相匹配。通过发出现任ABI 107a所期望的标识符,即使混合二进制108也与EC ABI 107b兼容,现任ABI 107a也可以在加载混合二进制108时看到期望的内容。

[0089] 方法600还包括标识和发出不可折叠函数的动作602。如图所示,动作602包括以下两者:(i)发出在本机ABI下执行的不可折叠函数的第一编译版本的动作602a和(ii)发出在兼容ABI下执行的不可折叠函数的第二编译版本的动作602b。如图所示,在动作602a和动作602b之间没有示出特定的顺序,并且将理解,这些动作可以并行或串行(以任一顺序)执行。在一些实施例中,动作602a包括,至少基于标识不可折叠的第一函数,向混合二进制图像发出在本机ABI下可执行的第一函数的第一编译版本发出,而动作602b包括,至少基于标识不可折叠的第一函数,向混合二进制图像发出在兼容性ABI下可执行的第一函数的第二编译版本。在示例中,至少基于将源代码函数编译为不同的编译版本(例如,在方法500的动作503和动作504中),并且还基于确定那些编译版本是不可折叠的(例如,在方法500的动作505中),编译器工具链107d在动作602a和动作602b中将这两个编译版本发出到混合二进制108中(例如,作为方法500的动作506的一部分)。

[0090] 方法600还包括标识和发出在本机ABI和兼容ABI下执行的可折叠函数的动作603,包括利用对不可折叠函数的第一编译版本的调用来发出可折叠函数的编译版本。在一些实施例中,动作603包括,至少基于标识可折叠的第二函数,将第二函数的编译版本发出到混合二进制图像中,该编译版本可在本机ABI和兼容ABI两者下执行。在示例中,至少基于将源代码函数编译成不同的编译版本(例如,在方法500的动作503和动作504中),并且还基于确定那些编译版本是可折叠的(例如,方法500的动作505中),编译器工具链107d在动作603中仅将那些编译版本中的其中一个发出到混合二进制108中(例如,作为方法500的动作507的一部分)。

[0091] 在实施例中,第二函数的编译版本被构造为当本机进程加载混合二进制图像时调用第一函数的第一编译版本。在示例中,编译器工具链107d配置混合二进制108,使得在动作603中发出的代码“默认”调用在动作602a中发出的代码,从而当混合二进制108加载到现任ABI 107a下时,执行动作602a(而不是动作602b)发出的代码。

[0092] 方法600还包括发出被兼容ABI利用的修正表的动作604。在一些实施例中,动作604包括将修正表发出到混合二进制图像中,修正表被构造为当混合二进制图像被在兼容ABI下执行的兼容进程(例如,仿真进程111)加载时使用,该修正表定义从混合二进制图像加载到存储器的多个变换。在示例中,编译器工具链107d向混合二进制108发出修正表108c,该修正表包括当混合二进制108被加载到EC ABI 107b下时应用的变换,该变换导致混合二进制108的EC特征被利用。

[0093] 在实施例中,修正表108c中的多个变换包括调整机器类型字段以包括与兼容ABI对应的第二机器类型的变换。在示例中,编译器工具链107d向修正表108c中发出存储器变换,该存储器变换(在系统存储器104中)用与和外部ABI 107c对应的外部ISA匹配的机器标识符替换从机器标识符303加载到系统存储器104的机器标识符。

[0094] 在实施例中,修正表108c中的多个变换包括将(可折叠的)第二函数的编译版本配置为调用(不可折叠的)第一函数的第二编译版本而不是(不可折叠的)第一函数的第一编译版本的变换。在示例中,编译器工具链107d将参考(不可折叠的)第一函数的第一编译版本的第一存储器地址替换为参考(不可折叠的)第一函数的第二编译版本的第二存储器地址的存储器变换发出到修正表108c中。虽然第一存储器地址可以直接在(可折叠的)第二函数的编译版本内被第二存储器地址替换,在实施例中,(可折叠的)第二函数的编译版本被

构造为经由参考(不可折叠的)第一函数的第一编译版本的分发表306来调用(不可折叠的)第一函数的第一编译版本。在这些实施例中,该变换本身不是修改(可折叠的)第二函数的编译版本,而是修改分发表306以参考(不可折叠的)第一函数的第二编译版本,而不是(不可折叠的)第一函数的第一编译版本。此外,在这些实施例中,方法600还包括将分发表发出到混合二进制图像中。

[0095] 在实施例中,方法600还包括将参考(可折叠的)第一函数的第一编译版本的入口点304发出到混合二进制图像中。在实施例中,该入口点304被构造为当本机进程加载混合二进制图像时(例如,使用现任ABI 107a)被利用。在实施例中,修正表108c中的多个变换包括调整入口点以参考(可折叠的)第一函数的第二编译版本的变换。因此,当通过兼容进程(例如,使用EC ABI 107b)加载混合二进制图像时,利用该经调整的入口点。

[0096] 如结合图3A和图3B所讨论的,混合二进制108可以包括一个或多个导入表或导出表(导入/导出表305),其中混合二进制108被配置为显示与现任ABI 107a相关的“本机”视图公开函数(及其位置)。例如,如图3B所示,可以使用公开本机视图307a的基本参考大小/计数来参考导入/导出表305。因此,在实施例中,方法600包括将一个或多个表发出到混合二进制图像中,该表包括(i)使用该(未折叠的)第一函数的第一编译版本来参考至少(未折叠的)第一函数的第一区域,(ii)使用该第二函数的编译版本来参考至少该(已折叠的)第二函数,以及(iii)使用(未折叠的)第一函数的第二编译版本参考至少该(未折叠的)第一函数的第三区域。

[0097] 在实施例中,为了提供这些表的“本机”视图,方法600包括在排除第三区域的同时将对一个或多个表的参考发出到混合二进制图像中,该参考提供包括第一区域和第二区域的一个或多个表的本机视图。该参考被构造为当本机进程加载混合二进制图像(例如,对应于现任ABI 107a)时使用。在这些实施例中,修复表108c中的多个变换然后包括在排除第一区域的同时调整对一个或多个表的参考的变换,以提供包括第二区域和第三区域的一个或多个表(供EC ABI 107b使用)的兼容性视图。因此,在实施例中,对一个或多个表的参考通过指定偏移和大小来提供一个或多个表的本机视图,并且修正表108c中的变换通过修改偏移或大小中的一个或多个来调整参考以提供这些表的兼容性视图。

[0098] 在实施例中,发出的表包括导入表;因此,在方法600的实施例中,一个或多个表包括一个或多个导入表,并且第一入口、第二入口和第三入口包括第一函数导入、第二函数导入和第三函数导入。在附加或备选实施例中,发出的表包括导出表;因此,在方法600的实施例中,或者一个或多个表包括一个或多个导出表,并且第一入口、第二入口和第三入口包括第一函数导出、第三函数导出和第二函数导出。

[0099] 现在参考二进制图像消耗和图7,方法700包括在使用兼容ABI的进程中发起混合二进制的加载的动作701。在示例中,加载器106c充分利用实现EC ABI 107b的库106a之一来发起混合二进制108的加载。

[0100] 方法700还包括确定存储在混合二进制中的机器类型与针对兼容ABI的机器类型不匹配的动作702。在一些实施例中,动作702包括在加载混合二进制图像期间,确定存储在混合二进制图像的机器类型字段中的第一机器类型与和正在执行该进程的兼容ABI对应的第二机器类型不匹配。在示例中,作为加载混合二进制108的一部分,加载器106c将包含机器标识符303的存储器页复制到系统存储器104中。然后,加载器106c确定机器标识符303的

值与EC ABI 107b的期望值(例如,与外部ABI 107c对应的外部ISA)不匹配。

[0101] 方法700还包括基于不匹配来定位修正表的动作703。在一些实施例中,动作703包括,基于确定第一机器类型与第二机器类型不匹配,在二进制图像内定位修正表,该修正表定义从混合二进制图像加载到存储器的多个变换。在示例中,加载器106c例如通过参考混合二进制108内的预定义地址或偏移,或者通过参考由从混合二进制108加载的存储器页填充的系统存储器104的一部分内的预定义地址或偏移来标识混合二进制108中的修正表108c。

[0102] 方法700还包括将修正表内的一个或多个变换应用于从混合二进制加载的存储器的动作704。在一些实施例中,动作704包括将多个变换的至少一部分应用于从混合二进制图像加载的存储器。在示例中,加载器106c将从修正表108c获得的一个或多个变换应用于由从混合二进制108加载的存储器页填充的系统存储器104的一个或多个部分。

[0103] 虽然加载器106c可以一次应用修正表108c中的所有变换,但是在实施例中,加载器106c在从混合二进制108加载这些页时(例如,作为页错误处理例程的一部分)逐页应用它们。因此,在图7中,动作704被示出为具有指向动作704的箭头,指示当从混合二进制108逐渐加载存储器页时,可以重复应用该动作。因此,在实施例中,结合处理存储器页错误,将对从混合二进制图像加载的存储器的变换应用于从混合二进制图像加载的存储器页。

[0104] 在实施例中,从修正表108c获得的多个变换包括调整机器类型字段以包括与兼容ABI对应的第二机器类型的变换。在示例中,加载器106c将变换应用于与系统存储器104内的机器标识符303对应的存储器位置,其调整存储器位置以存储与和外部ABI 107c对应的外部ISA相匹配的值。

[0105] 在实施例中,从修正表108c获得的多个变换包括修改调用在本机ABI下可执行的未折叠函数的第一编译版本的调用站点的变换,以代替调用在兼容ABI下可以执行的未折叠函数的第二编译版本。在示例中,加载器106c应用与调用站点对应的存储器位置,以将参考未折叠函数的第一编译版本的第一存储器地址替换为参考未折叠函数的第二编译版本的第二存储器地址。虽然调用站点本身可以被变换,但是在实施例中,调用站点被构造为经由参考未折叠函数的第一编译版本的分发表306来调用未折叠函数的第一编译版本。在这些实施例中,该变换本身不是修改调用站点,而是修改分发表306以参考未折叠函数的第二编译版本,而不是未折叠函数的第一编译版本。

[0106] 在实施例中,混合二进制图像包括参考未折叠函数的第一编译版本的入口点。在实施例中,从修正表108c获得的(并且在动作704中应用的)多个变换包括调整入口点以参考未折叠函数的第二编译版本的变换。

[0107] 如结合图6所讨论的,在实施例中,方法600发出一个或多个表(例如与导入表和/或导出表对应)以及对提供本机视图的这些表的参考。因此,在方法700的上下文中,在一些实施例中,混合二进制图像包括一个或多个表,其包括(i)使用未折叠函数的第一编译版本参考至少该非折叠函数的第一区域,(ii)参考至少该折叠函数的第二编译版本的第二区域,以及(iii)使用未折叠函数的第二编译版本参考至少未折叠函数的第三区域。在这些实施例中,混合二进制图像还包括在排除第三区域的同时对一个或多个表的参考,该参考提供了包括第一区域和第二区域的一个或多个表的本机视图。为了在EC ABI 107b下加载二进制图像时提供兼容性视图,方法700可以包括在排除第一区域的同时应用调整对一个或

多个表的参考的变换,以提供包括第二区域和第三区域的一个或多个表的兼容性视图。

[0108] 具有一致性参考地址的双架构函数指针

[0109] 图8示出了一个示例方法800的流程图,该方法用于在处理进程内的调用时使用公共参考存储器地址,该进程支持执行以下两者:(i) 以与本机ISA对应并且具有第一CC的本机ABI为目标的本机代码,以及(ii) 以与外部ISA对应并且具有第二CC的外部ABI为目标的

外部代码。

[0110] 如图所示,方法800包括使用参考地址标识对被调用方函数的调用的动作801。在一个示例中,分派器112a标识从EC本机代码108b中的本机函数到参考存储器地址的调用。在另一示例中,分派器112b标识从外部二进制109中的外部函数(并且正由仿真器106b仿真)到参考存储器地址的调用。

[0111] 方法800还包括使用查找结构和参考地址来确定被调用方函数是对应于本机ABI还是对应于外部ABI的动作802。在实施例中,动作802包括,至少基于标识以被调用方函数的参考存储器地址为目标的调用,确定被调用方函数是对应于本机ABI还是对应于外部ABI。在实施例中,被调用方函数基于指示参考存储器地址包含在存储本机代码的第一存储器范围内的查找结构确定为对应于本机ABI,并且被调用方函数至少基于指示参考存储器地址包含在不存储本机代码的第二存储器范围内的查找结构确定为对应于外部ABI。在示例中,分派器112(当调用方是本机的时,它可以是分派器112a,或者当调用方是外部的时,可以是分派器112b)查询EC查找结构307以确定被调用方函数的参考存储器地址是否在与EC本机代码108b对应的存储器区域内(在这种情况下,被调用方被确定为对应于EC ABI 107c),或者被调用方函数的参考存储器地址是否不在与EC本机代码108b对应的存储器区域内(在这种情况下,被调用方被确定为对应于外部ABI 107c)。

[0112] 如所指出的,在实施例中,EC查找结构307是位图,其使用一个位值来指示对应的存储器范围(例如,存储器页)是否危及EC本机代码108b。然而,EC查找结构307可以包括备选数据结构类型,例如哈希表或二叉树。因此,在方法800中,查找结构包括位图、哈希表或二叉树中的至少一个。

[0113] 如所讨论的,可能存在另一种情况,其中被调用方函数被确定为对应于本机ABI,即,当分派器112b确定被调用方函数包含快进序列的签名时。因此,在动作802的一些实施例中,基于以下之一来确定被调用方函数对应于本机ABI:(i) 指示参考存储器地址包含在存储本机代码的第一存储器范围内的查找结构,或(ii) 在参考存储器地址处标识的快进序列。此外,在动作802的一些实施例中,当查找结构指示参考存储器地址包含在不存储本机代码的第二存储器范围内时,动作802包括确定快进序列是否在参考存储器地址处可标识。在这些实施例中,分派器112b从快进序列中获得新的参考存储器地址(即,本机函数的真实地址),并将该新的参考存储器地址用于本机调用。因此,在实施例中,当在参考存储器地址处标识快进序列时,方法800包括用从快进序列获得的新参考存储器地址更新参考存储器地址。

[0114] 方法800还包括发起被调用方函数的执行的动作803。在实施例中,动作803包括,至少基于该确定,发起被调用方函数的执行。如图所示,动作803包括执行以下之一:动作803a,当调用方是外部的并且被调用方是外部的时,在仿真器中直接调用参考地址;动作803b,当调用方是外部的并且被调用方是本机的时,定位并调用入口形实转换程序;动作

803c,当调用方是本机的并且被调用方是外部的时,调用出口形实转换程序;或动作803d,当调用方是本机的并且被调用方是本机的时,直接调用参考地址。

[0115] 在实施例中,动作803a包括,当调用方函数对应于外部ABI时,并且当被调用方函数被确定为对应于外部ABI时,在仿真器内使用参考存储器地址直接调用被调用方函数。在示例中,基于来自外部函数的调用,分派器112b确定被调用方也是外部函数。因此,调用方正在调用被调用方函数的真实存储器地址,并且不需要形实转换程序,因此分派器112b使用仿真器106b内的参考存储器地址直接调用被调用方函数。

[0116] 在实施例中,动作803b包括,当调用方函数对应于外部ABI时,以及当被调用方函数被确定为对应于本机ABI时调用入口形实转换程序,该入口形实转换程序(i)使第二CC适应于第一CC,然后(ii)使用参考存储器地址直接调用被调用方函数。在示例中,基于来自外部函数的调用,分派器112b确定被调用方是本机函数。因此,分派器112b不能直接调用(即,被调用方函数的)参考存储器地址,因为这将绕过入口形实转换程序。相反,分派器112b将新的参考地址定位到入口形实转换程序,并且使用新的参考存储器地址调用入口形实转换程序。入口形实转换程序进而将外部ABI 107c的CC适应为EC ABI 107b的CC,并使用原始参考存储器地址调用被调用方函数。

[0117] 如所讨论的,在实施例中,分派器112b基于包含在紧接在被调用方函数的原始参考存储器地址之前的存储器块中的数据(例如偏移或直接地址参考)来定位到入口形实转换程序的新参考地址。因此,在实施例中,方法800还包括至少基于以下来标识入口形实转换程序的位置:(i)读取紧接在参考存储器地址之前的存储器块,以及(ii)从存储器块确定到入口形实转换程序的位置的偏移或指针。

[0118] 在实施例中,动作803c包括,当调用方函数对应于本机ABI时,以及当被调用方函数被确定为对应于外部ABI时调用出口形实转换程序,该出口形实转换程序(i)将本机ABI的第一CC适应为外部ABI的第二CC,然后(ii)调用仿真器以使用参考存储器地址直接调用被调用方函数。在示例中,基于来自本机函数的调用,分派器112a确定被调用方是外部函数。因此,分派器112a不能直接调用(即,被调用方函数的)参考存储器地址,因为这将绕过出口形实转换程序。相反,分派器112a定位到出口形实转换程序的新参考地址,并使用新的参考存储器地址调用出口形实转换程序。出口形实转换程序进而将EC ABI 107b的CC适应为外部ABI 107c的CC,并调用仿真器106b。仿真器106b进而使用原始参考存储器地址直接调用被调用方函数。如所讨论的,在实施例中,出口形实转换程序的新参考地址包含在被调用方函数本身内。因此,在方法800的一些实施例中,出口形实转换程序的位置包含在调用方函数内。

[0119] 在实施例中,动作803d包括,当调用方函数对应于本机ABI时,以及当被调用方函数被确定为对应于本机ABI时,使用参考存储器地址直接调用被调用方函数。在示例中,基于来自本机函数的调用,分派器112a确定被调用方也是本机函数。因此,调用方正在调用被调用方函数的真实存储器地址,并且不需要形实转换程序,因此分派器112a使用参考存储器地址直接调用被调用方函数。

[0120] 尽管主题已经以特定于结构特征和/或方法动作的语言描述,但是应当理解,所附权利要求中定义的主题不一定限于上述描述的特征或动作,或者上述动作的顺序。相反,所描述的特征和动作被公开为实现权利要求的示例形式。

[0121] 如下面更详细地讨论的,本发明的实施例可以包括或利用专用或通用计算机系统(例如,计算机系统101),其包括计算机硬件,例如,一个或多个处理器(例如,处理器102)和系统存储器(例如,系统存储器104)。本发明范围内的实施例还包括用于承载或存储计算机可执行指令和/或数据结构的物理和其他计算机可读介质。这种计算机可读介质可以是可由通用或专用计算机系统访问的任何可用介质。存储计算机可执行指令和/或数据结构的计算机可读介质是计算机存储介质(例如,持久存储器103、系统存储器104)。携带计算机可执行指令和/或数据结构的计算机可读介质是传输介质。因此,作为示例而非限制,本发明的实施例可以包括至少两种明显不同的计算机可读介质:计算机存储介质和传输介质。

[0122] 计算机存储介质是存储计算机可执行指令和/或数据结构的物理存储介质。物理存储介质包括计算机硬件,例如RAM、ROM、EEPROM、固态驱动器(“SSD”)、闪存、相变存储器(“PCM”)、光盘存储、磁盘存储或其他磁存储设备,或可用于以计算机可执行指令或数据结构的形式的形式存储程序代码的任何其他硬件存储设备,其可由通用或专用计算机系统访问和执行以实现本发明的公开功能。

[0123] 传输介质可以包括网络和/或数据链路,其可以用于以计算机可执行指令或数据结构的形式的形式携带程序代码,并且可以由通用或专用计算机系统访问。“网络”被定义为一个或多个数据链路,其能够在计算机系统和/或模块和/或其他电子设备之间传输电子数据。当信息通过网络或另一通信连接(硬连线、无线、或硬连线或无线的组合)传输或提供给计算机系统时,计算机系统可以将该连接视为传输介质。上述内容的组合也应包括在计算机可读介质的范围内。

[0124] 此外,一旦到达各种计算机系统组件,计算机可执行指令或数据结构形式的程序代码可以自动地从传输介质转移到计算机存储介质(反之亦然)。例如,通过网络或数据链路接收的计算机可执行指令或数据结构可以缓冲在网络接口模块(例如,“NIC”)内的RAM中,然后最终转移到计算机系统RAM和/或计算机系统处的易失性较低计算机存储介质。因此,应当理解,计算机存储介质可以包括在计算机系统组件中,这些组件也(或甚至主要)利用传输介质。

[0125] 计算机可执行指令包括例如指令和数据,当在一个或多个处理器处执行时,这些指令和数据使通用计算机系统、专用计算机系统或专用处理设备执行特定功能或功能组。计算机可执行指令可以是例如二进制、诸如汇编语言的中间格式指令,甚至是源代码。

[0126] 本领域技术人员将理解,本发明可以在具有许多类型的计算机系统配置的网络计算环境中实践,包括个人计算机、台式计算机、膝上型计算机、消息处理器、手持设备、多处理器系统、基于微处理器或可编程的消费电子产品、网络PC、小型计算机、大型计算机、移动电话、PDA、平板计算机、寻呼机、路由器、交换机等。本发明还可以在分布式系统环境中实践,其中本地和远程计算机系统都执行任务,本地和远程计算机系统通过网络链接(通过硬连线数据链路、无线数据链路或通过硬连线和无线数据链路的组合)。这样,在分布式系统环境中,计算机系统可以包括多个组成计算机系统。在分布式系统环境中,程序模块可以位于本地和远程存储器存储设备中。

[0127] 本领域技术人员还将理解,本发明可以在云计算环境中实践。云计算环境可以是分布式的,尽管这不是必需的。当分布式时,云计算环境可以在组织内国际分布和/或具有跨多个组织拥有的组件。在本说明书和以下权利要求中,“云计算”被定义为用于实现对可

配置计算资源(例如,网络、服务器、存储、应用和服务)的共享池的按需网络访问的模型。“云计算”的定义并不局限于在适当部署时能够从这种模型中获得的任何其他众多优势。

[0128] 云计算模型可以由各种特征组成,如按需自助服务、广泛的网络访问、资源池、快速弹性、可测量的服务,等等。云计算模型也可以各种服务模型的形式出现,例如,软件即服务(“SaaS”)、平台即服务(PaaS)和基础设施即服务(IaaS)。还可以使用不同的部署模型来部署云计算模型,例如私有云、社区云、公共云、混合云,等等。

[0129] 一些实施例(例如云计算环境)可以包括一个或多个主机的系统,每个主机能够运行一个或多个虚拟机。在操作期间,虚拟机模拟可操作的计算系统,同时支持操作系统和一个或多个其他应用。在一些实施例中,每个主机包括管理程序,该管理程序使用从虚拟机的视图中抽象出来的物理资源来仿真虚拟机的虚拟资源。管理程序还提供虚拟机之间的适当隔离。因此,从任何给定虚拟机的角度来看,管理程序提供虚拟机与物理资源交互的错觉,即使虚拟机仅与物理资源的外观(例如,虚拟资源)交互。物理资源的示例包括处理容量、存储器、磁盘空间、网络带宽、媒体驱动器等。

[0130] 在不脱离本发明的精神或基本特征的情况下,本发明可以以其他特定形式实施。所描述的实施例在所有方面仅被认为是说明性的而非限制性的。因此,本发明的范围由所附权利要求而不是由前述描述来指示。在权利要求的含义和等效范围内的所有变更都应包含在其范围内。当在所附权利要求中引入元素时,条款“一”、“一个”、“该”和“所述”旨在表示存在一个或多个元素。术语“包括”、“包含”和“具有”旨在具有包容性,意味着可存在除所列元素之外的其他元素。术语“集”和“子集”被缩进以排除空集,因此“集”被定义为非空集,“子集”被定义为非空子集。

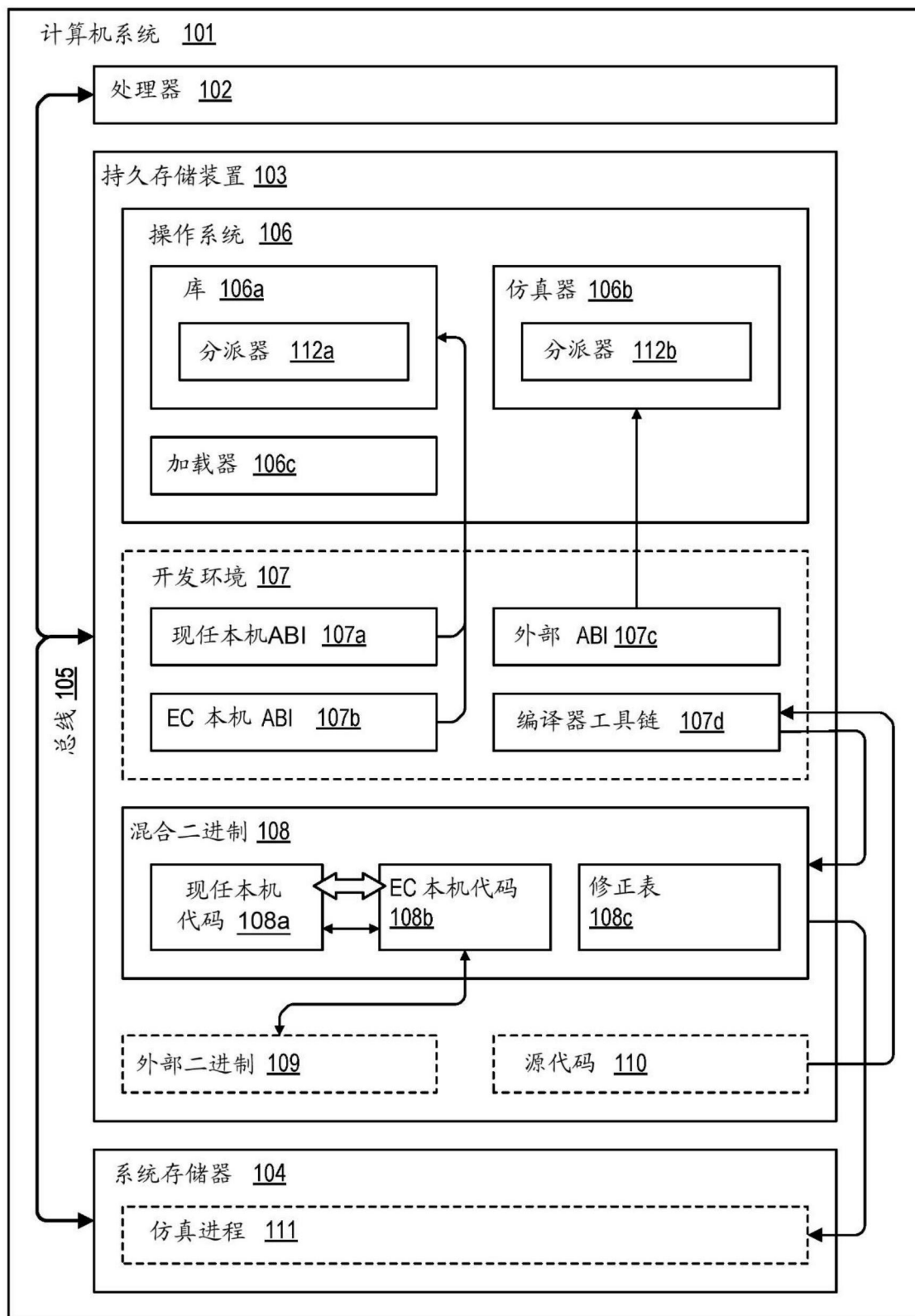


图1

200

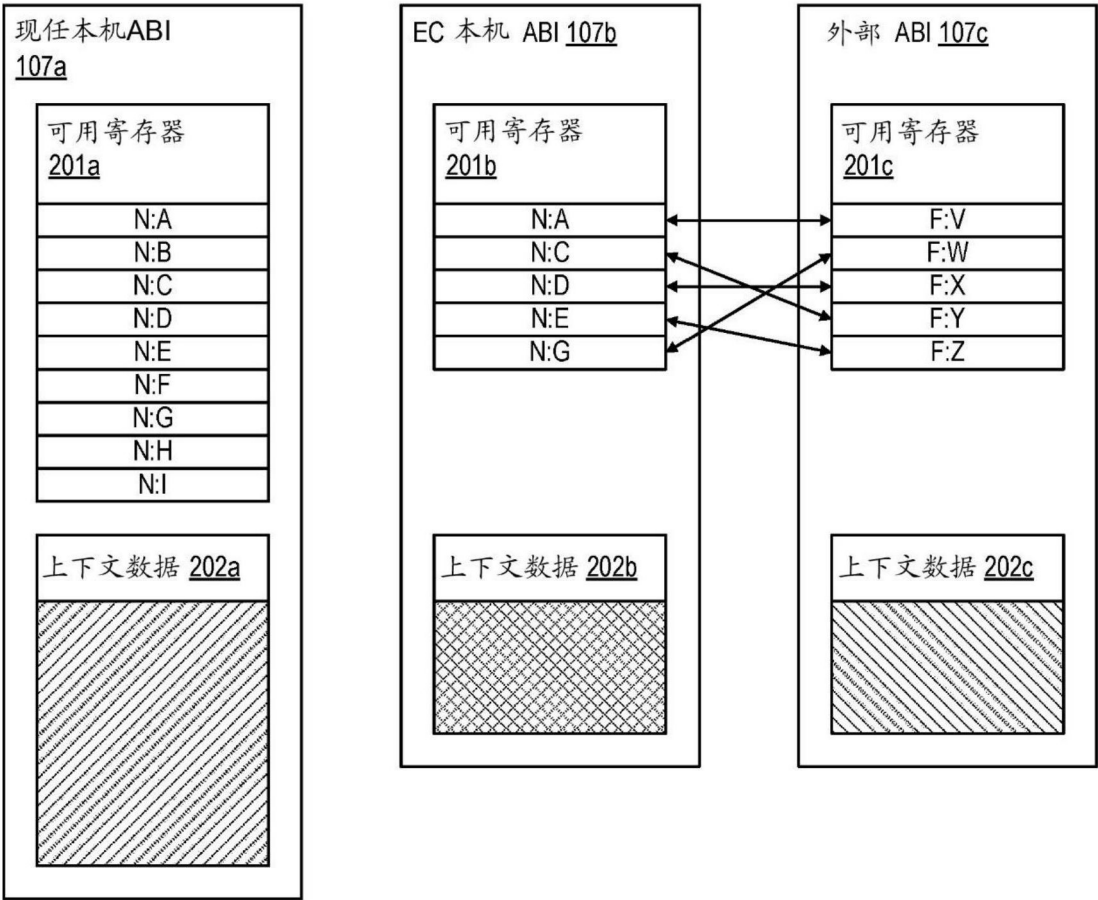


图2

300a

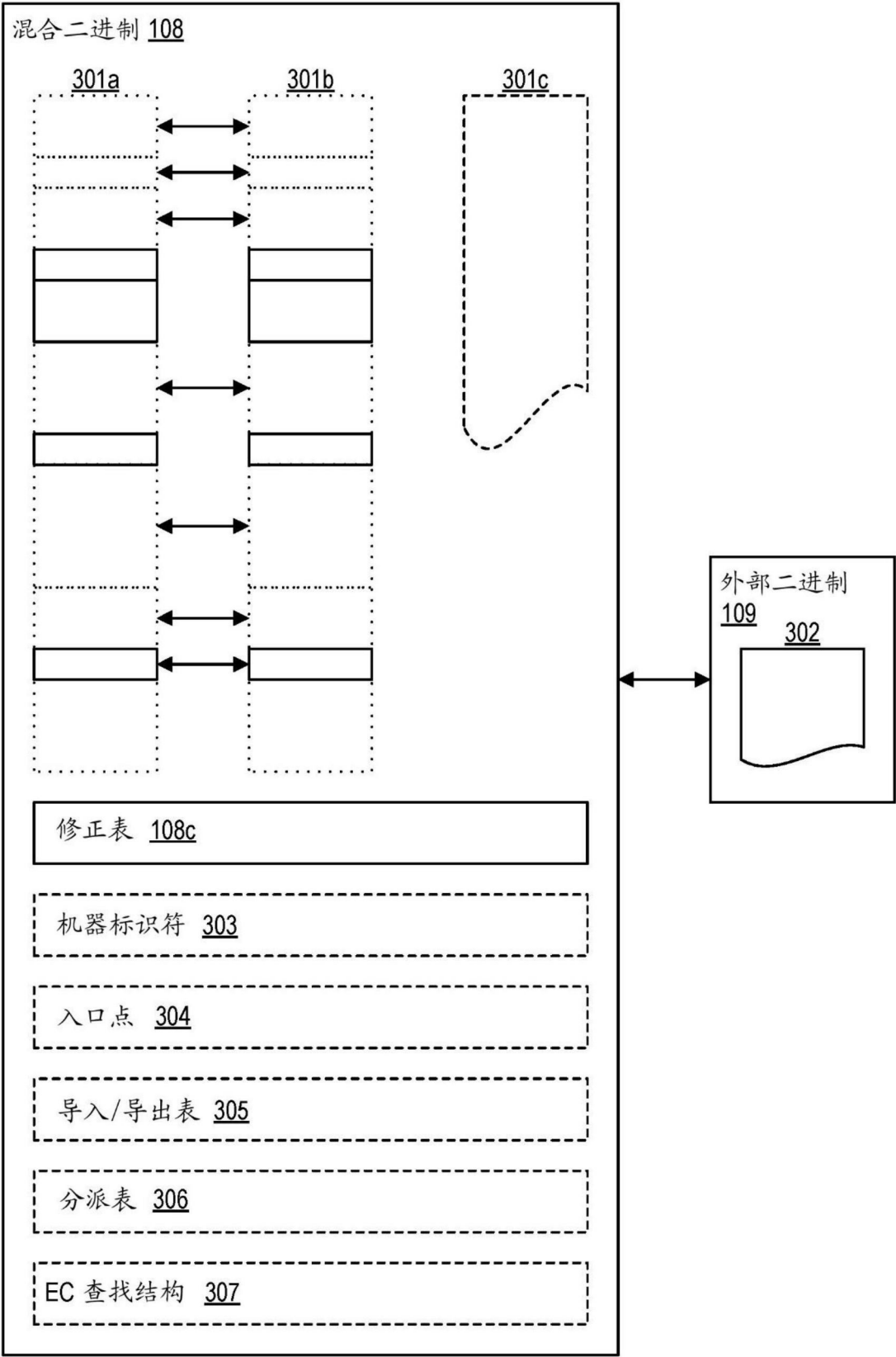


图3A

300b

导入/导出表 305

		函数	位置
307a	{	A	地址/偏移 1
		B	地址/偏移 2
		C	地址/偏移 3
		D	地址/偏移 4
		E	地址/偏移 5
		A'	地址/偏移 6
307b	{	B'	地址/偏移 7
		F	地址/偏移 8

图3B

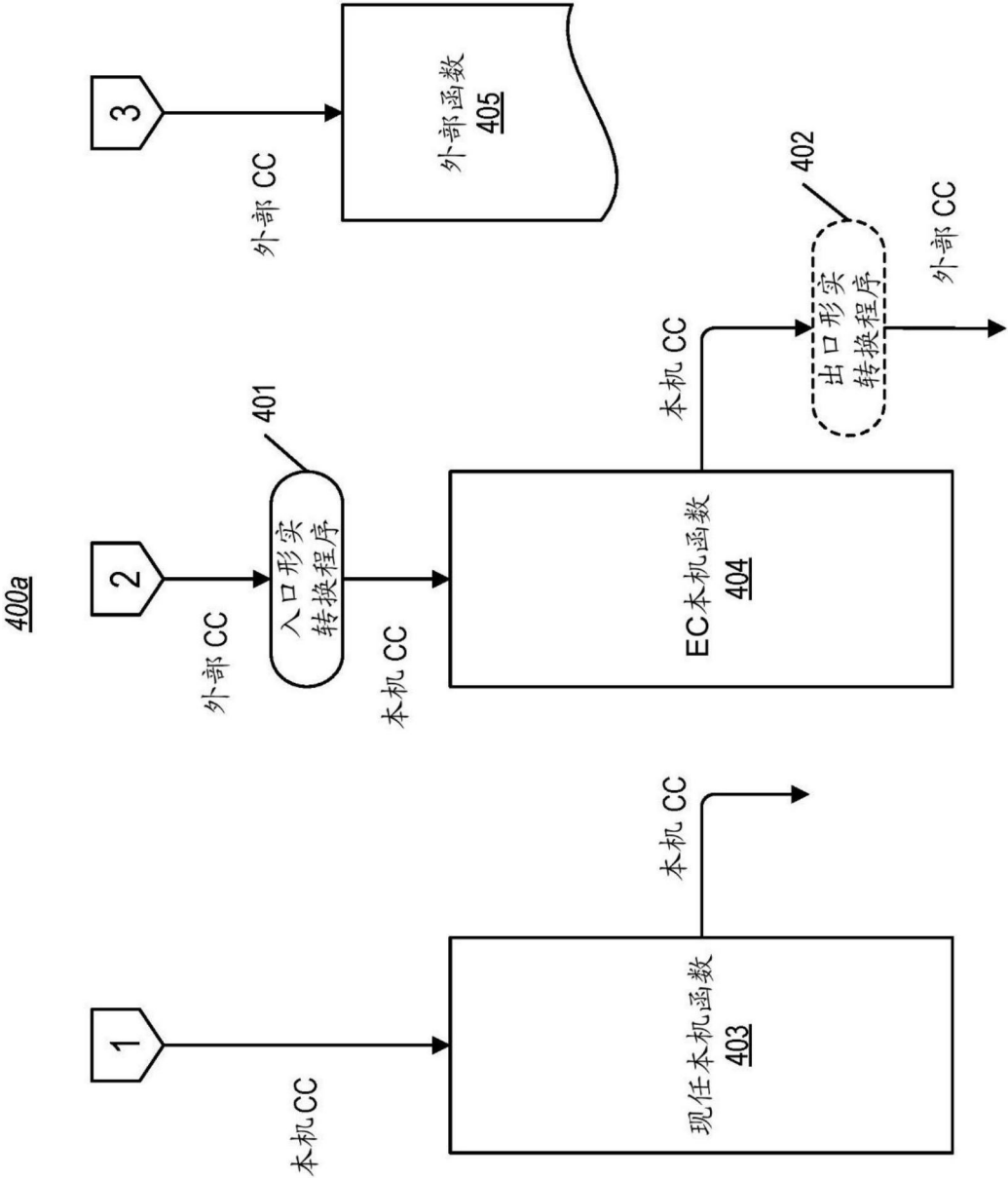


图4A

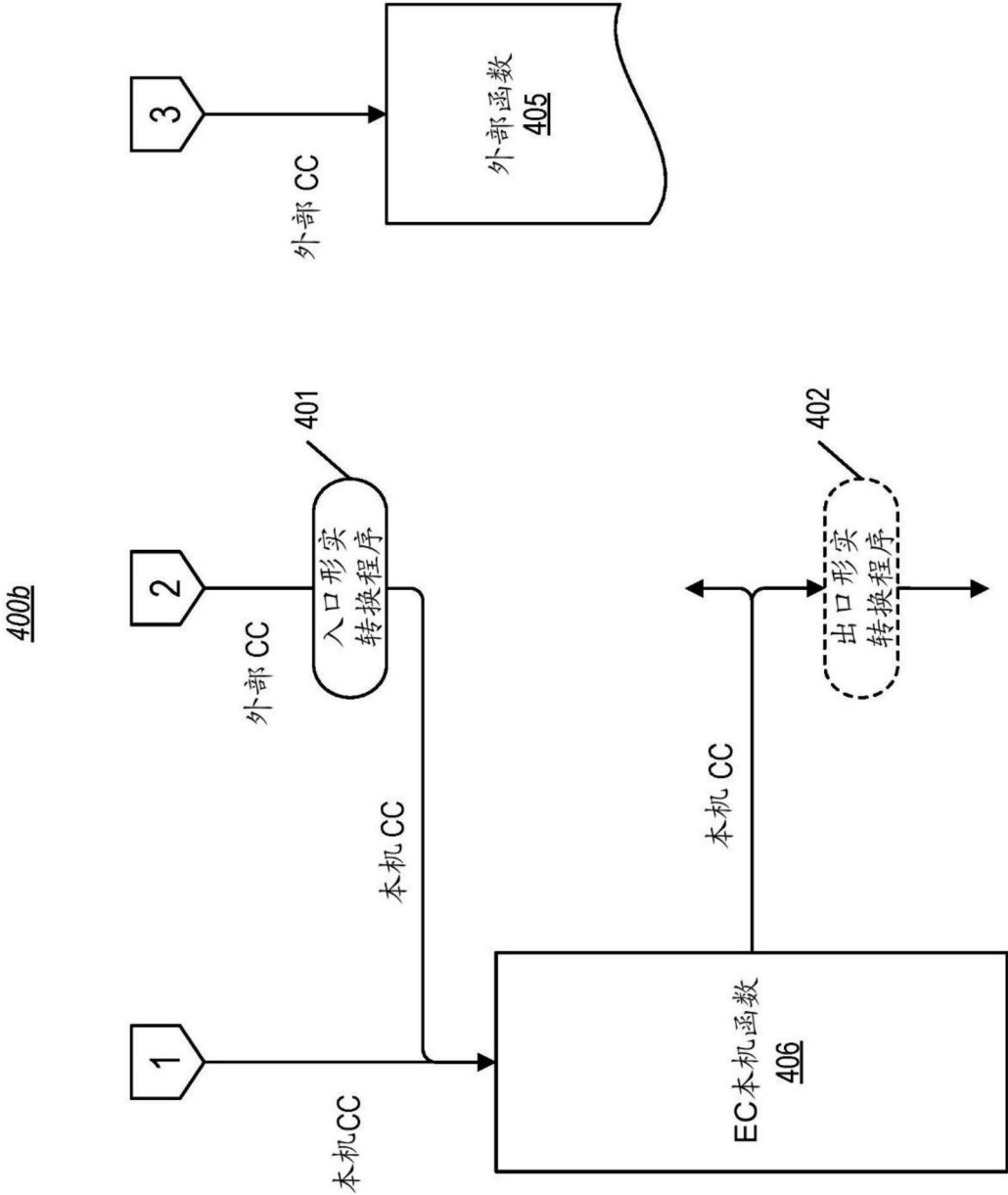


图4B

500

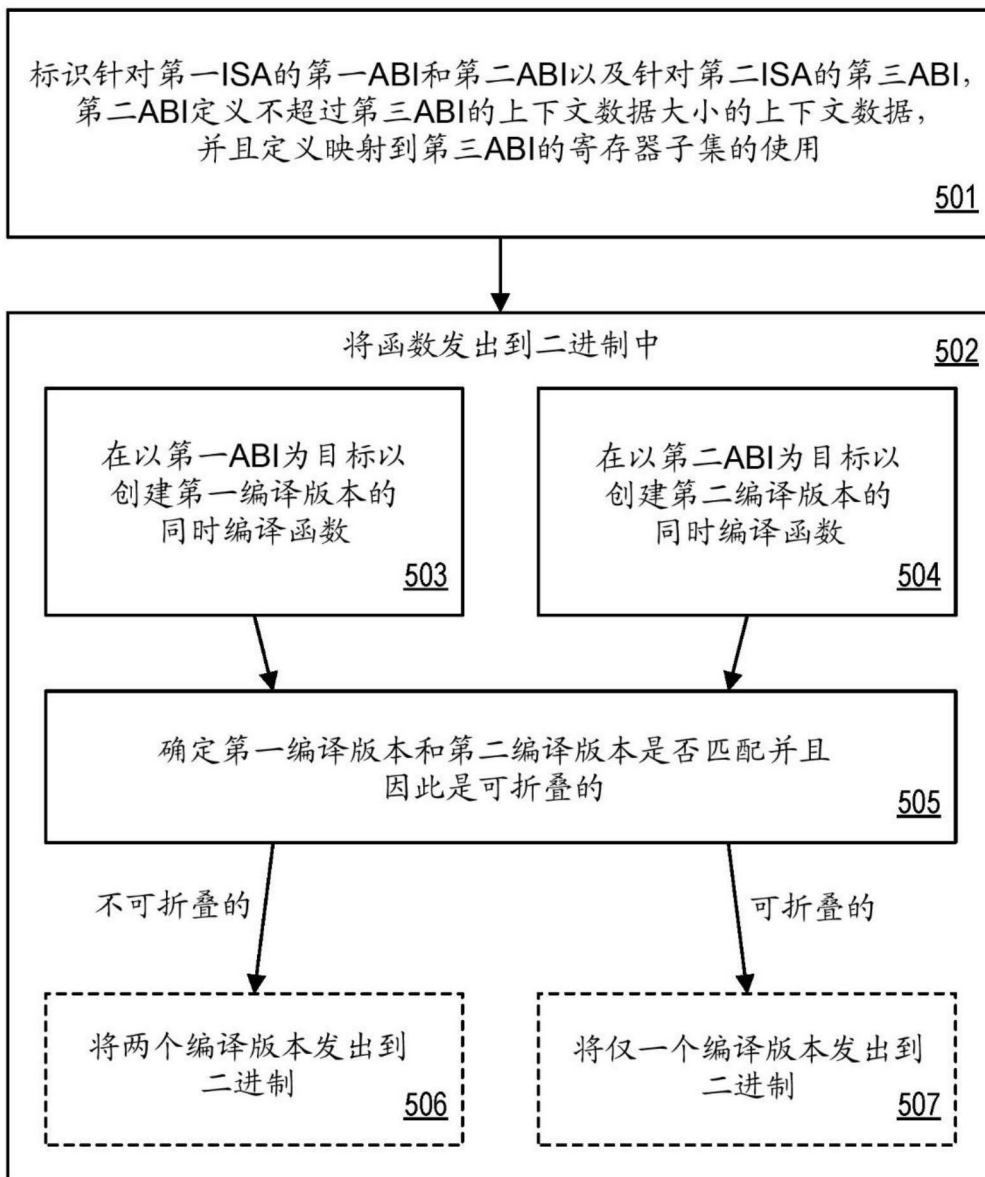


图5

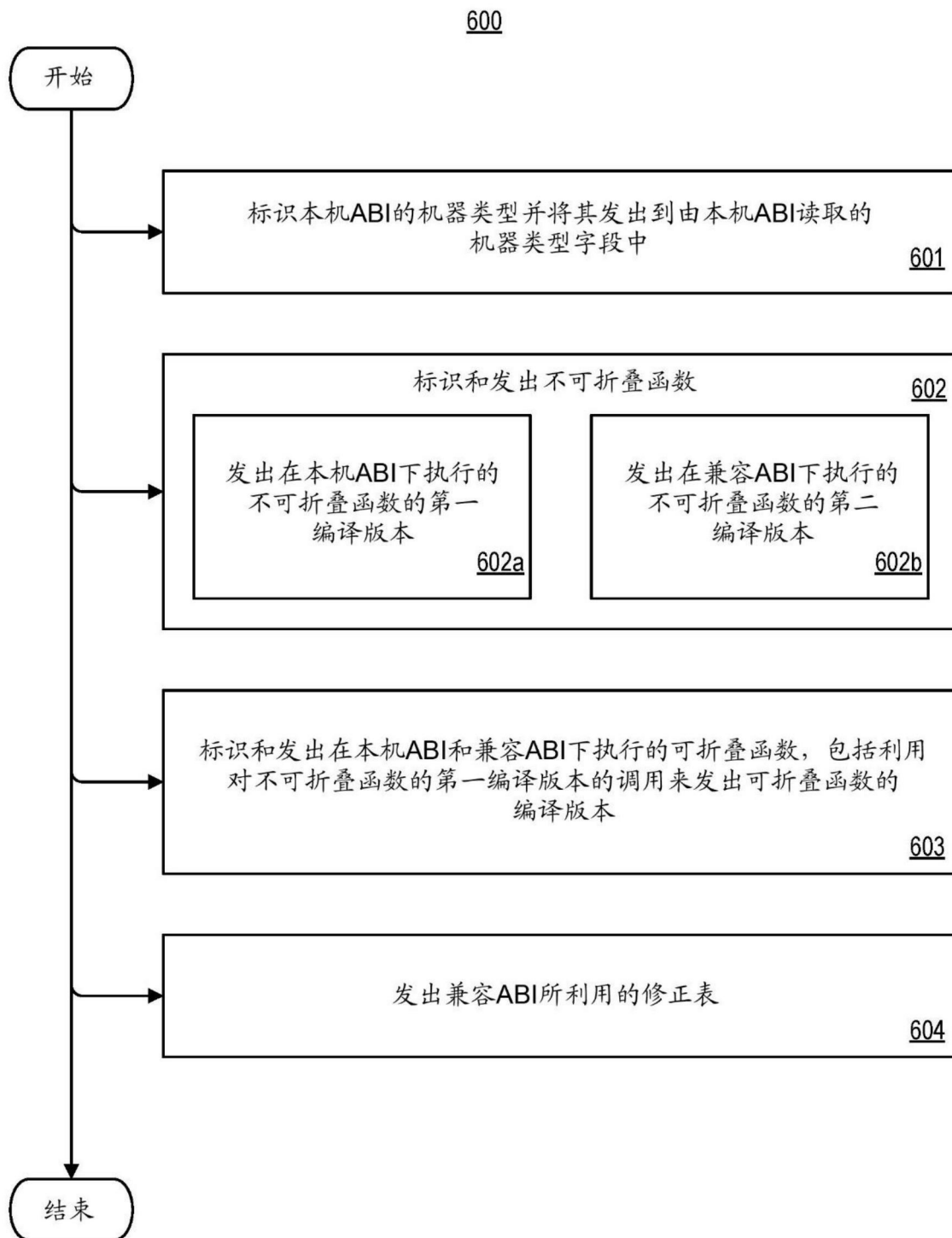


图6

700

图7

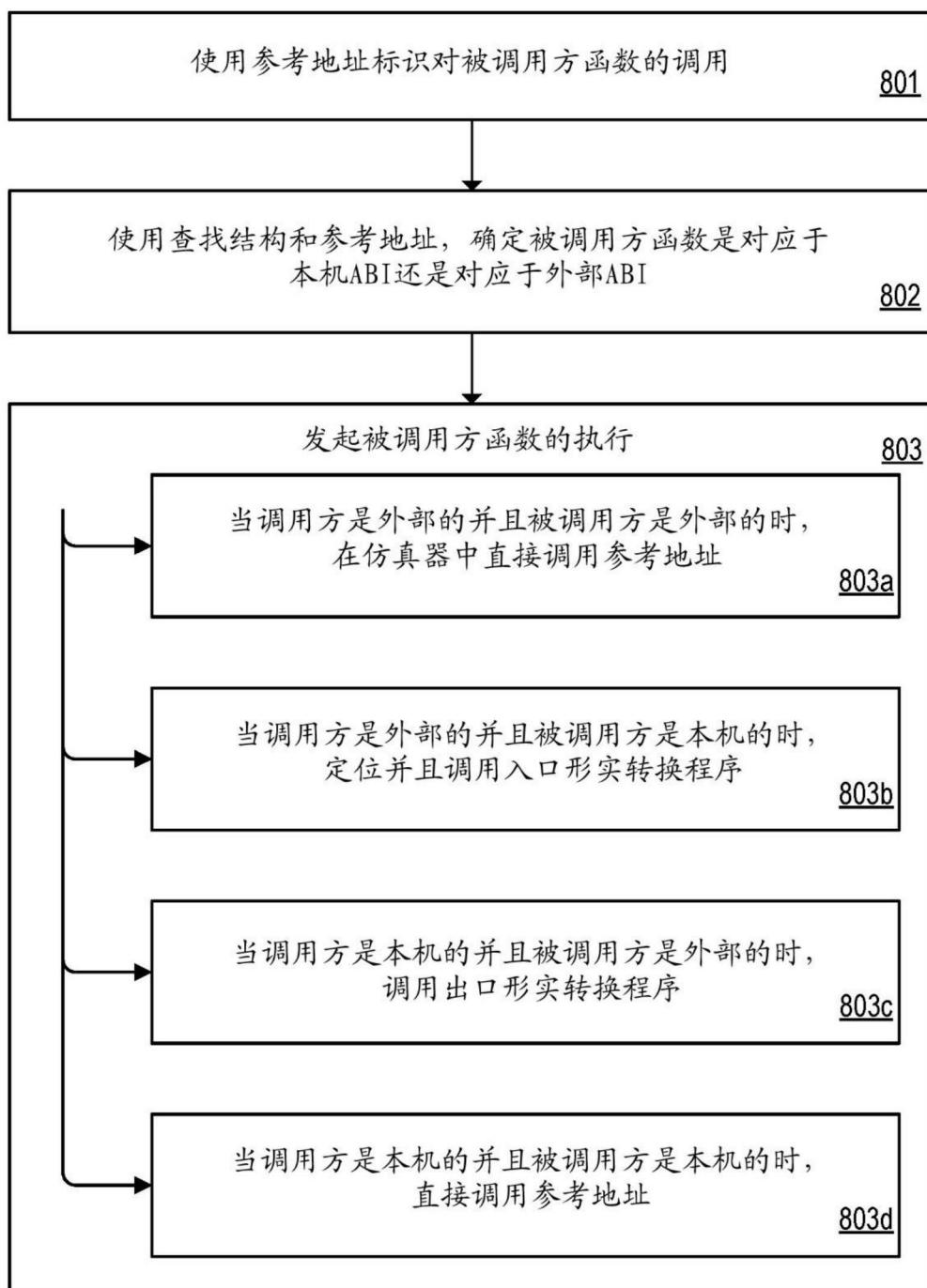
800

图8