(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2021 年 12 月 16 日 (16.12.2021)

WIPO | PCT

(10) 国际公布号
WO 2021/249193 A1

(51) 国际专利分类号:
G06F 9/455 (2006.01)

(21) 国际申请号: PCT/CN2021/096075

(22) 国际申请日: 2021 年 5 月 26 日 (26.05.2021)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
202010539884.3 2020年6月12日 (12.06.2020) CN

(71) 申请人: 华 为 技 术 有 限 公 司 (**HUAWEI TECHNOLOGIES CO.,LTD.**) [CN/CN]; 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。

(72) 发明人: 蒋毅飞 (**JIANG, Yifei**); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。 赵思齐 (**ZHAO, Siqi**); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。 万波 (**WAN, Bo**); 中国广东省深圳市龙岗区坂田华为总部办公楼, Guangdong 518129 (CN)。

(81) 指定国(除另有指明，要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL,

WO 2021/249193 A1

(54) **Title:** METHOD AND APPARATUS FOR PROCESSING EXCEPTION OR INTERRUPT UNDER HETEROGENEOUS INSTRUCTION SET ARCHITECTURE

(54) 发明名称: 一种异构指令集架构下异常或中断的处理方法、装置

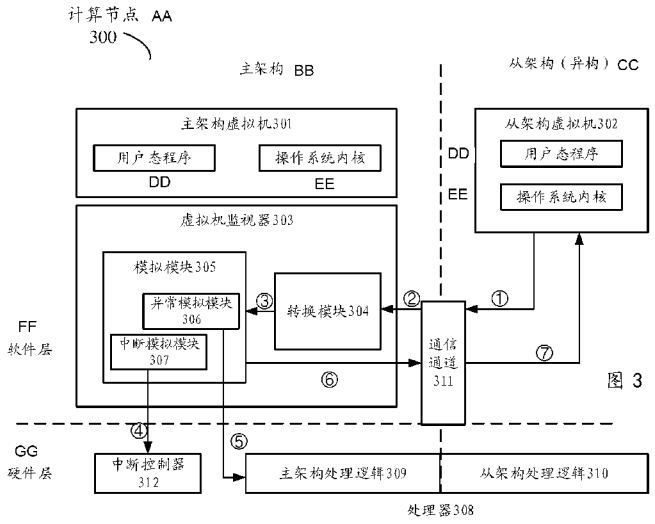

图 3

301	Master architecture virtual machine	311	Communication channel
302	Slave architecture virtual machine	312	Interrupt controller
303	Virtual machine monitor	AA	Computing node
304	Conversion module	BB	Master architecture
305	Simulation module	CC	Slave architecture (heterogeneous)
306	Exception simulation module	DD	User mode program
307	Interrupt simulation module	EE	Operating system kernel
308	Processor	FF	Software layer
309	Master architecture processing logic	GG	Hardware layer
310	Slave architecture processing logic		

(57) **Abstract:** Provided is a method for processing an exception or interrupt under a heterogeneous instruction set architecture. A physical host to which the method is applied can support two instruction set architectures. When a slave architecture virtual machine (302) triggers an exception or interrupt, a virtual machine monitor (303) can convert a code of the exception or interrupt under a slave instruction set architecture into a code of the exception or interrupt under a master instruction set architecture. The virtual machine monitor (303) can identify the code of the exception or interrupt under the master instruction set architecture, and therefore, the virtual

[见续页]

ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW。

(84) **指定国**(除另有指明，要求每一种可提供的地区保护)：ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布:

— 包括国际检索报告(条约第21条(3))。

machine monitor (303) identifies, by means of the code obtained after the conversion, the type of the exception or interrupt triggered by the slave architecture virtual machine (302), thereby processing the exception or interrupt. Therefore, the virtual machine monitor (303) can identify and process the exception or interrupt triggered by the slave architecture virtual machine (302), such that the stability of a system can be ensured, a heterogeneous instruction set virtual machine can operate normally, and a diversified software ecosystem is constructed.

(57) 摘要：提供了一种异构指令集架构下异常或中断的处理方法，该方法应用的物理主机能够支持两种指令集架构。从架构虚拟机（302）触发异常或中断时，虚拟机监视器（303）可以将从指令集架构下该异常或中断的编码转换成主指令集架构下的该异常或中断的编码，虚拟机监视器（303）可以识别主指令集架构下的异常或中断的编码，因此虚拟机监视器（303）通过转换后的编码识别出从架构虚拟机（302）触发的异常或中断的类型，从而对其处理。由此虚拟机监视器（303）可以识别、处理从架构虚拟机（302）触发的异常或中断，能够保证系统的稳定性、使得异构指令集虚拟机能正常运行，构建多元化的软件生态。

一种异构指令集架构下异常或中断的处理方法、装置

技术领域

本申请涉及计算机领域，具体涉及一种异常或中断的处理方法。

背景技术

指令集是存储于处理器中，用来引导处理器进行加减运算和控制计算机操作系统的一系列指令的集合。指令集或指令集架构也是处理器能支持的指令的集合。目前常见的指令集架构（instruction set architecture, ISA）有 X86 架构、高级精简指令集机器（advanced risc machine，ARM）架构和 RISC-V 架构等。不同的处理器支持的指令集架构可能不同，处理器要支持新的指令集架构，就要修改硬件电路，软件要支持新的指令集，就要修改程序，重新编译。当需要在新的指令集架构执行业务时，支持原始指令集架构的应用程序和操作系统不能直接在支持新的指令集架构的处理器上运行，而需要被重新开发以支持新的指令集架构。例如 X86 指令集架构下某个软件的二进制文件，不能直接在支持 ARM 指令集架构的硬件或软件环境中运行，需要基于 ARM 指令集架构重新开发、编译以获得该软件的新的二进制文件。

在虚拟化场景下，若需要在该服务器上新启动一个支持 ARM 架构的虚拟机，但此时服务器上搭载的虚拟化平台以及虚拟化平台上运行的虚拟机均是支持 X86 架构，则需要把支持 X86 架构的虚拟化平台替换成支持 ARM 架构的虚拟化平台，再在支持 ARM 架构的虚拟化平台上启动支持 ARM 架构的虚拟机，并且服务器的硬件设备，包括处理器，均需要替换成支持 ARM 架构的硬件设备。这种整套硬件和软件的替换的代价太大，也不利于构建多元化的软件生态。

如果要在支持一种指令集架构的虚拟化平台上运行支持不同指令集架构的多个虚拟机，构建多元化的软件生态，需要让支持异构指令集架构的虚拟机的指令能被执行或处理，尤其是异常或中断。由于支持异构指令集架构的虚拟机与虚拟化平台各自支持的指令集架构不同，而虚拟化平台只能识别出虚拟化平台所支持的指令集架构下触发的异常或中断，所以异构指令集架构下触发的中断或异常不能被正常识别、处理，会导致系统崩溃。

发明内容

本申请提供的异常或中断处理方法，可以应用在支持异构指令集虚拟机的物理主机上，能够处理异构指令集的虚拟机触发的异常或中断，提高系统的稳定性。

第一方面，本申请实施例提供了一种异常的处理方法。该方法应用在支持异构指令集虚拟机的物理主机上，异构指令集架构指的是至少有两种不同的指令集架构---主指令集架构和从指令集架构，如本申请实施例中提到的 RISC-V 架构和 ARM 架构。也就是说该物理主机上可以运行有支持不同指令集架构的虚拟机，主架构虚拟机（例如 ARM 虚拟机）和从架构虚拟机（例如 RISC-V 虚拟机）。该物理主机的处理器包括支持主指令集架构的主处理核和支持从指令集架构的从处理核。其中，主处理核和从处理核是处理器中支持不同指令集架构的逻辑核，逻辑核是物理核上逻辑划分出的处理单元；在本申请

各个实施例中处理核还称为处理逻辑。该方法包括：

从架构虚拟机触发异常或中断时，获得该异常或中断的状态信息；其中，该异常或中断的状态信息为异常或中断的第一编码，第一编码用于表示在从指令集架构下该异常或中断的类型，在从指令集架构的规范中使用第一编码来表示该异常或中断的类型。从异常映射关系或中断映射关系中获取从架构虚拟机触发的异常或中断的第二编码，根据异常或中断的第二编码识别出从架构虚拟机触发的异常或中断的类型，从而能处理该异常或中断。

在不同指令集架构下，对于同一类异常或同一类中断的编码是不同的。在不同指令集架构下，使用不同的编码来代表同一类异常或同一类中断的类型。编码的不同，包括格式的不同以及编码值的不同；或者说编码的不同在于编码所包括的字段的数量以及每个字段的值是不同的。在从指令集架构的规范中，从架构虚拟机触发的这类异常或中断的类型用第一编码表示；而在主指令集架构的规范中，采用第二编码来表示该类异常或中断的类型。第一编码和第二编码是不同的。异常映射关系和中断映射关系可以存储在主架构虚拟机监视器可访问的存储空间（例如内存）。异常映射关系就记录了多类异常中每类异常的第一编码和第二编码的对应关系，这种对应关系表示主指令集架构下异常的类型与从指令集架构下异常的类型的对应关系，例如对于对应关系中的任一条，可表示同一类型或相似类型的异常分别在主指令集架构的规范中的第二编码以及在从指令集架构的规范中的第一编码。中断映射关系就记录了多类中断中每类中断的第一编码和第二编码的对应关系，这种对应关系表示主指令集架构下中断的类型与从指令集架构下中断的类型的对应关系，例如对于对应关系中的任一条，可表示同一类型或相似类型的中断分别在主指令集架构的规范中的第二编码以及在从指令集架构的规范中的第一编码。其中，从架构虚拟机触发的中断可包括从处理核接收到的物理中断，以及从架构虚拟机的虚拟处理器接收到的虚拟中断。

在第一方面以及下述第一方面的任一种实现方式中所描述的方法可以由物理主机上运行的支持主指令集架构的虚拟机监视器来执行，由于虚拟机监视器是支持主指令集架构的，无法直接根据异常的第一编码识别出从架构虚拟机触发的异常的类型。所以虚拟机监视器从异常映射关系中获取从架构虚拟机触发的异常的第二编码，根据异常的第二编码识别出从架构虚拟机触发的异常的类型，从而能处理该异常。其中虚拟机监视器可以被称为 virtual machine monitor，VMM，也可以被称为 hypervisor。

由于处理器中包括了支持主指令集架构的主处理核和支持从指令集架构的从处理核，因此从架构虚拟机的指令可以直接被从处理核执行，无需进行指令翻译，能够避免指令翻译带来的指令膨胀问题。并且对于从架构虚拟机触发的异常或中断，虚拟机监视器可以根据异常映射表或中断映射表，将无法识别的第一编码转换成可以识别的第二编码，再根据第二编码获知异常或中断的类型从而进行处理。这种方法可以保证异构指令集架构的虚拟机的正常运行，从架构虚拟机的异常或中断可以被准确识别并处理；不会因为虚拟机监视器无法识别异常或中断的类型而导致无法处理该异常或中断从而造成系统崩溃。因此，本方法可以提高系统稳定性、使得异构指令集虚拟机能正常运行，构建多元化的软件生态。

进一步的，虚拟机监视器是根据与从架构虚拟机触发的异常或中断对应的主指令集架构下的该异常或中断的编码来识别该异常或中断的类型，且虚拟机监视器本身就能够

处理支持主指令集架构的主架构虚拟机触发的异常或中断，并且针对不同类型的异常或中断有对应的处理逻辑。所以物理主机处理从架构虚拟机触发的异常或中断的过程可以复用对主架构虚拟机触发异常或中断的软件处理逻辑。相比于设置支持两种指令集架构的异常或中断处理逻辑，本申请实施例提供的方法仅使用支持主指令集架构的虚拟机监视器就能处理两种指令集架构的虚拟机触发的异常或中断，实现难度较小，且不会增加软件设计的复杂度和成本。

在一种实现方式中，物理主机中包括支持硬件辅助虚拟化的硬件设备，且该硬件设备支持主指令集架构。硬件辅助虚拟化指的是一种平台虚拟化方法，它可以使用来自硬件功能的帮助来实现有效的虚拟化。在硬件辅助虚拟化（hardware-assisted virtualization）中，硬件提供结构支持帮助创建虚拟机监视器并允许客户机操作系统独立运行。例如 X86 架构下 MMU 中的 EPT、二级地址转换（RISC 架构下称为 Two-Stage Address Translation，ARM 架构下称为 stage 2 translations）用于加速处理虚拟地址到物理地址的映射过程；例如硬件的中断控制器用于获取其他硬件设备的中断请求，发送给处理器。

该方法中处理异常或中断，包括：通过支持硬件辅助虚拟化的硬件设备处理异常或中断，从而提高对异常的处理速度。由于复用了主指令集架构下的支持硬件辅助虚拟化的硬件设备，所以不会增加硬件涉及的复杂度和成本。

在一种实现方式中，物理主机还包括从架构寄存器；前述获得从架构虚拟机触发的异常或中断的状态信息，包括：

从共享内存或共享寄存器中获取该异常或中断的状态信息。其中，共享内存或共享寄存器是从处理核和主处理核之间的通信通道，被从处理核和主处理核共享，是从处理核和主处理核都可访问的存储空间，用于存储异常的状态信息。异常或中断的状态信息为从处理核从从架构寄存器中复制到共享内存中的。由于从架构寄存器是符合从指令集架构规范的寄存器，而支持主指令集架构的虚拟机监视器无法直接从从架构寄存器中读取异常的状态信息，因此本方法设计了共享内存来传递异常或中断的状态信息。

在另一种实现方式中，物理主机还包括共享寄存器；前述获得从架构虚拟机触发的异常或中断的状态信息，包括：从共享寄存器中获取异常或中断的状态信息，其中，共享寄存器被从处理核和主处理核共享，用于存储从架构虚拟机触发的异常的状态信息的寄存器。与仅支持单一指令集架构规范的现有寄存器不同，该共享寄存器可以符合主从两种指令集架构规范，可以被主处理核和从处理核访问。对于从架构这一侧，共享寄存器可用于存储从架构虚拟机触发的异常或中断的状态信息，从主架构这一侧来说，主架构虚拟机监视器可以从共享寄存器中获取该异常或中断的状态信息。

在一种实现方式中，异常的状态信息还包括触发该异常的指令或触发该异常的地址中的至少一个；还可以包括其他的状态信息，对于不同类型的异常，其异常状态信息可以不同，异常状态信息表示处理该异常所需的信息，用于表示触发异常时系统的状态，例如系统调用指令触发的异常，该异常的状态信息是系统调用的参数，存储在从架构寄存器中。其中，触发异常的指令为导致从架构虚拟机触发异常的指令，或者说是从架构虚拟机触发异常时正在执行的指令；触发异常的地址为导致从架构虚拟机触发异常的地址，或者说是从架构虚拟机触发异常时，从架构虚拟机要访问的内存地址。异常映射关系或中断映射关系还包括从架构寄存器和主架构寄存器的对应关系，主架构寄存器为符合主指令集架构规范的寄存器，从架构寄存器为符合从指令集架构规范的、用于存储该

异常的状态信息的寄存器。

在根据所述异常的第二编码，识别出异常或中断的类型，并处理异常或中断之前，该方法还包括：根据异常映射关系或中断映射关系查找到与从架构寄存器对应的主架构寄存器；将异常的第二编码，以及将触发异常的指令和触发异常的地址中的至少一个写入主架构寄存器；或将中断的第二编码写入主架构寄存器；

对应的，根据异常或中断的第二编码，识别出异常或中断的类型并处理异常或中断，包括：从主架构寄存器中读取异常的第二编码，以及触发异常的指令和触发异常的地址中的至少一个，根据异常的第二编码，以及触发异常的指令和所述触发异常的地址中的至少一个处理该异常；或者从主架构寄存器中读取中断的第二编码，根据中断的第二编码，识别出中断的类型并处理中断。由于虚拟机监视器处理主架构虚拟机触发的异常或中断时，是从主架构寄存器中读取异常或中断的状态信息，从而对该异常或中断进行处理。为了较少的改动原有的虚拟机监视器的软件逻辑，本方法中虚拟机监视器查找到从架构虚拟机触发的异常或中断的第二编码后，将第二编码以及触发异常的指令和触发异常的地址写入主架构寄存器里，虚拟机监视器就可以把从架构虚拟机触发的异常或中断当做主架构虚拟机触发的异常一样进行处理。

当然，在另一种实现方式中，虚拟机监视器可直接根据得到的异常或中断的第二编码以及从共享内存或共享寄存器中获取的触发异常的指令和所述触发异常的地址，识别并处理异常或者中断，不需要先将上述信息存储在主架构寄存器中，再从主架构寄存器中读取以进行异常或中断的识别或处理。

第二方面，本申请提供了一种计算节点，该计算节点包括存储器和处理器，处理器包括支持主指令集架构的主处理核和支持从指令集架构的从处理核，存储器存储计算机指令，主处理核运行计算机指令以执行如第一方面以及任一种实现方式所描述的方法。

第三方面，本申请提供一种计算机可读存储介质，计算机可读存储介质存储计算机指令，计算机指令被处理器调用以执行如第一方面以及任一种实现方式所描述的方法。

第四方面，本申请提供一种物理主机，该物理主机包括：转换模块和模拟模块。其中转换模块的具体实现可参考具体实施例中的转换模块 304 或 RISC-V 转换模块 404；用于处理异常的模拟模块的具体实现可参考具体实施例中的 RISC-V 异常模拟模块 406 或异常模拟模块 306；用于处理中断的的模拟模块的具体实现可参考具体实施例中的 RISC-V 中断模拟模块 407 或中断模拟模块 307。

转换模块用于：

从架构虚拟机触发异常或中断时，获得异常或中断的状态信息。其中，从架构虚拟机为物理主机上运行的支持从指令集架构的虚拟机；异常的状态信息包括异常的第一编码；异常的第一编码表示在从指令集架构下异常的类型；中断的状态信息包括中断的第一编码；中断的第一编码表示在从指令集架构下中断的类型；

从异常映射关系或中断映射关系中获取异常或中断的第二编码。其中，异常的第二编码表示在主指令集架构下所述异常的类型；中断的第二编码表示在所述主指令集架构下所述中断的类型；异常映射关系包括多类异常中每类异常的第一编码与第二编码的对应关系，中断映射关系包括多类中断中每类中断的第一编码与第二编码的对应关系。

模拟模块用于：根据异常或中断的第二编码，识别出异常或中断的类型；处理异常或中断。

在一种实现方式中，物理主机还包括从架构寄存器，转换模块还用于：

从共享内存中获取异常或中断的状态信息，其中共享内存被从处理核和主处理核共享，异常或中断的状态信息为从从架构寄存器中复制到共享内存中的；从架构寄存器为符合从指令集架构规范的寄存器，用于存储从架构虚拟机触发的异常或中断的状态信息。

在一种实现方式中，物理主机还包括共享寄存器，转换模块还用于：

从共享寄存器中获取异常或中断的状态信息，其中共享寄存器被从处理核和主处理核共享，且共享寄存器存储有异常或中断的状态信息。

在一种实现方式中，物理主机还包括从架构寄存器和主架构寄存器，中断映射关系还包括从架构寄存器与主架构寄存器的对应关系，主架构寄存器为符合主指令集架构规范的寄存器；从架构寄存器为符合所述从指令集架构规范的、用于存储所述中断的状态信息的寄存器；

转换模块还用于：根据中断映射关系查找与从架构寄存器对应的主架构寄存器；将中断的第二编码写入主架构寄存器；

模拟模块用于：从主架构寄存器中读取中断的第二编码；根据中断的第二编码处理中断。

在一种实现方式中，物理主机还包括从架构寄存器和主架构寄存器，异常的状态信息还包括触发异常的指令或触发异常的地址中的至少一个，异常映射关系还包括从架构寄存器与主架构寄存器的对应关系，主架构寄存器为符合主指令集架构规范的寄存器，从架构寄存器为符合从指令集架构规范的、用于存储异常的状态信息的寄存器；

转换模块还用于：根据异常映射关系查找与从架构寄存器对应的主架构寄存器；将异常的第二编码写入主架构寄存器，以及将触发异常的指令和触发异常的地址中的至少一个写入主架构寄存器；

模拟模块用于：从主架构寄存器中读取异常的第二编码，以及触发异常的指令和触发异常的地址中的至少一个；根据异常的第二编码，以及触发异常的指令和触发异常的地址中的至少一个处理异常。

在一种实现方式中，物理主机中包括支持硬件辅助虚拟化的硬件设备，且硬件设备支持所述主指令集架构；

模拟模块用于：通过支持硬件辅助虚拟化的硬件设备处理异常。

第五方面，本申请提供一种计算机程序产品，计算机程序产品包括计算机指令，计算机指令被处理器调用以执行如第一方面或第二方面以及任一种实现方式所描述的方法。

第六方面，本申请提供一种芯片，芯片上包括支持主指令集架构的主处理核和支持从指令集架构的从处理核，主处理核被配置为执行如第一方面以及任一种实现方式所描述的方法。

附图说明

图 1 为本申请实施例提供的计算节点 100 的结构示意图；

图 2 为本申请实施例提供的计算节点 203 的结构示意图；

图 3 为本申请实施例提供的计算节点 300 的结构示意图；

图 4 为本申请实施例提供的计算节点 400 的结构示意图；

图 5 为本申请实施例提供的处理 RISC-V 虚拟机触发的异常的流程图；

图 6 为本申请实施例提供的处理 RISC-V 虚拟机触发的中断的流程图；

图 7 为本申请实施例提供的 RISC-V 架构规范中 scause 寄存器的结构示意图；

图 8 为本申请实施例提供的 ARM 架构规范中 esr_el2 寄存器的结构示意图；

图 9 为本申请实施例提供的计算节点 700 的结构示意图；

图 10 为本申请实施例提供的多核处理器 80 的结构示意图。

具体实施方式

为了方便理解本申请各个实施例，首先以图 1 所示的计算节点 100 为例介绍本申请所涉及的虚拟化领域内一些基本概念。

虚拟化是将计算节点的硬件层中的硬件资源（例如处理器、存储器中的存储空间以及网络资源）虚拟化后共享给多个虚拟计算机使用。虚拟计算机为所有类型的虚拟化设备中通过软件虚拟出来的运行环境的统称，该概念包括虚拟机、容器。

如图 1 所示，计算节点 100 包括硬件层 112、宿主机层 109 和虚拟化层，虚拟化层包含虚拟机 101 和 102。虚拟机的个数还可以更多或更少，在此只以两个为例。硬件层 112 包括处理器 114、存储器 113、通信接口 115 和中断控制器 116。

虚拟机（virtual machine ,VM）是通过虚拟化软件在物理计算节点上模拟出的一台或者多台虚拟计算机。这些虚拟机运行在完全隔离的环境中，就像真正的计算机那样进行工作。虚拟机（图 1 中 101 和 102）上可以安装客户操作系统（guest operating system, guest OS）（图 1 中 105 和 106），客户操作系统上运行有一个或多个应用程序（图 1 中 103 和 104）。虚拟机还可访问网络资源。对于在虚拟机中运行的应用程序而言，就像是在真正的计算机中工作。

虚拟处理器（如图 1 中 107 和 108）：在虚拟化技术下，代表以共享或者分片方式提供给虚拟计算机使用的物理处理单元，例如虚拟 CPU(virtual central processing unit , vCPU)。一台虚拟计算机可以有一个或多个虚拟处理器为其服务，当存在多个虚拟处理器时，通常有一个虚拟处理器为主虚拟处理器，其他为从虚拟处理器。虚拟机包含的虚拟存储器等其他虚拟硬件资源在图 1 中未示出。应理解，虚拟机相当于一台独立的计算机，所以虚拟机执行动作也可以认为是虚拟处理器执行该动作。虚拟处理器是通过虚拟化软件虚拟出的，它的运行实际是宿主机的处理器或物理核读取并运行软件程序实现的，例如一个物理核读取软件程序并在该物理核的硬件辅助虚拟化的特定模式（例如 x86 的 non-Root 模式）下运行该软件程序以实现一个虚拟处理器。一台虚拟机的多个虚拟处理器可以位于不同的物理核上。

虚拟处理器陷入和虚拟处理器陷出：虚拟化系统包括两种模式：宿主模式（host mode）与客户模式（guest mode）。当一个虚拟处理器进入客户模式，叫做陷入（虚拟）；当该虚拟处理器离开客户模式，叫陷出（虚拟）。虚拟处理器陷出后物理处理器将暂时不执行该虚拟处理器的代码，所以此时可以理解为虚拟处理器没有运行。针对一个物理处理器而言，其上运行的虚拟处理器陷入，则可以认为该物理处理器处于客户模式，运行虚拟处理器的代码，当其上运行的虚拟处理器陷出到宿主模式，则可以认为该物理处理器处于宿主模式，运行宿主机相关的代码，比如虚拟机监视器。

宿主机（host）层 109 作为管理层，用以完成硬件资源的管理、分配，为虚拟机呈现虚拟硬件平台，实现虚拟机的调度和隔离等。在一些实现方式下，宿主机层 109 可以包括宿主机操作系统 111 和虚拟监控装置，例如虚拟机监视器 110（virtual machine monitor,

VMM）。其中虚拟监控器 110 可部署在宿主机操作系统 111 之内，也可以部署在宿主机操作系统 111 之外。在其他虚拟化架构中虚拟监控装置还可以称为 hypervisor 或其他类型的虚拟监控装置。在另一些实现方式下，例如在虚拟化架构 Xen 中，宿主机层 109 还可以包括 1 个特权虚拟机。其中，虚拟硬件平台对其上运行的各个虚拟计算机提供各种硬件资源，如虚拟处理器、虚拟内存、虚拟磁盘、虚拟网卡等。虚拟计算机则运行在宿主机层为其准备的虚拟硬件平台上。宿主机层 109 还可以称为虚拟化平台，有时宿主机层还可以简称为宿主机。

硬件层 112：虚拟化环境运行的硬件平台。其中，硬件层可包括多种硬件，如图 1 所示，硬件层可包括处理器 114 和存储器 113，还可以包括通信接口 115，例如网卡（network interface card，NIC）；还可以包括中断控制器 116、输入/输出（input/output , I/O）设备等。其中处理器 114 中可包括多个物理核，例如核 1 和核 0。

处理器 114，有时称为物理处理器。物理核代表处理器中最小处理单元，如图 1 所示，本实施例中处理器可具有两个物理核：核 0 和核 1，以及多个寄存器。在其他一些实施例中，处理器包含的核的数量可以更多或更少，各个处理器包含的核的个数也可以不同。具有多个物理核的处理器被称为多核处理器。按照内核架构是否相同，可以分为同构多核与异构多核。虚拟处理器和物理核可以是绑定的关系，即一个虚拟处理器固定在某个物理核上运行，不能被调度到其他物理核上运行，则该虚拟处理器为绑核；一个虚拟处理器可以根据需要被调度到不同的物理核上运行，则该虚拟处理器为非绑核。

中断控制器 116：设置在触发中断请求的硬件和处理器之间，主要用于收集各个硬件产生的中断请求，并按照一定的优先级或其他规则发送给处理器。例如高级可编程中断控制器(advanced programmable interrupt controller, APIC)。

中断（interruption）是指暂停当前程序的指令转而执行中断服务程序。

中断服务程序（interrupt service routine, ISR）是用于处理中断请求的程序。当处理器接收到中断请求时，暂时停止当前程序的执行转而执行该中断请求对应的中断服务程序。

中断请求（interrupt request）：中断请求指的是硬件产生的一种事件，硬件将该事件发送到处理器，当处理器接收到该事件时，暂时停止当前程序的执行转而执行该事件对应的程序。硬件产生中断请求可能是硬件自己触发的，也可能是软件触发硬件产生的。中断请求有时也简称为中断。计算机中的某些硬件（例如网卡、声卡、鼠标、硬盘等）能在没有处理器介入的情况下完成一定的工作，但是这些硬件还是需要定期中断处理器，让处理器为其做一些特定的工作。中断号为一个中断请求的标识，本申请中用英文 IRQ ID 表示。

硬件层 112 还可以包括内存管理单元（memory management unit，MMU）。MMU 是一种负责处理内存访问请求的计算机硬件。它的功能包括虚拟地址到物理地址的转换（即虚拟内存管理）、内存保护、中央处理器高速缓存的控制。MMU 通常借助转译后备缓冲器（translation lookaside buffer， TLB）的相联高速缓存（associative cache）来将虚拟页号转换为物理页号。

存储器 113 提供的存储空间（地址空间）被划分给虚拟机和宿主机使用。宿主机物理地址 (host physical address，HPA)指的是本地主机（宿主机）可以使用的物理地址空间；宿主机虚拟地址 (host virtual address，HVA)是本地主机（宿主机）可使用的虚拟地址空

间。客户机物理地址 (guest physical address，GPA)是虚拟机的客户操作系统可使用的物理地址空间；客户机虚拟地址 (guest virtual address，GVA)是虚拟机的客户操作系统可使用的虚拟地址空间。

虚拟机请求访问 GVA 时，MMU 需要将 GVA 转换至 GPA，再将 GPA 转换为 HPA。影子页表用于完成 GVA 到 HPA 的直接转换。而扩展页表（extended page table，EPT）用于完成 GPA 到 HPA 的转换，而 GVA 到 GPA 的转换使用客户机页表进行地址转换。EPT 由 VMM 维护，EPT 的转换过程由硬件完成，因此比影子页表转换效率更高。用于加速处理虚拟地址到物理地址的映射的过程在不同架构下有不同的名称，在 X86 架构下称为 EPT、RISC 架构下称为 Two-Stage Address Translation，ARM 架构下称为 stage 2 translations。

图 2 示出本申请适用的一种系统架构 200，该系统架构 200 可适用于公有云、私有云或终端云的场景下。该系统架构 200 包括云管理平台 201、一个或多个计算节点 203 和云网络 202，云管理平台 200 可通过云网络 202 与一个或多个计算节点 203 通信连接，从而配置、管理这些计算节点。计算节点之间也可通过云网络进行通信连接。，计算节点 203 可以为物理设备，例如服务器或终端设备。终端设备可以是具有无线连接功能的手持式设备、或连接到无线调制解调器的其他处理设备。例如，可以为移动电话、计算机、平板电脑、个人数码助理（personal digital assistant，PDA）、移动互联网设备（mobile Internet device，MID）、可穿戴设备和电子书阅读器（e-book reader）等；也可以是便携式、袖珍式、手持式、计算机内置的或者车载的移动设备。

图 2 中的计算节点 203 可以包括图 1 中的计算节点 100 的部分或全部组件。计算节点 203 上可运行有支持不同指令集架构的多个虚拟机，即主架构虚拟机和从架构虚拟机。不同的指令集架构包括主指令集架构（简称为主架构）和从指令集架构（简称为从架构）。主架构和从架构可以为任意两种的指令集架构，例如 X86 架构、ARM 架构和 RISC-V 架构中的任意两种。RISC-V 架构是一个基于精简指令集（reduced instruction set computing，RISC）的开源指令集架构。

其中，计算机指令就是指挥机器工作的指示和命令，程序就是一系列按一定顺序排列的指令，执行程序的过程就是计算机的工作过程。指令集（instruction set），就是 CPU 中用来计算和控制计算机系统的一套指令的集合，每款 CPU 在设计时就规定了一系列与其硬件电路相配合的指令系统。指令的强弱也是 CPU 的重要指标，指令集是提高微处理器效率的最有效的工具之一。常见的指令集架构 ISA 有复杂指令集运算（complex instruction set computing，CISC）和精简指令集运算（reduced instruction set computing，RISC），其中，CISC 的典型代表是 X86，RISC 的典型代表是高级精简指令集机器（advanced risc machine，ARM）架构和无内部互锁流水级的微处理器（microprocessor without interlocked pipelined stages，MIPS）架构。

虚拟化平台软件位于虚拟机和硬件层中间、用于管理控制其上运行的虚拟机，虚拟化平台可以包括图 1 中的宿主机层 109 的部分或全部。该虚拟化平台软件上可运行有支持不同指令集架构的虚拟机，如图 2 中的主架构虚拟机和从架构虚拟机。计算节点 203 上可能之前只运行有主架构虚拟机，当用户需要一台支持新指令集的虚拟机时，用户可

以通过云管理平台 201 向计算节点 203 下发请求以创建从架构虚拟机，从架构虚拟机支持新的指令集架构。虚拟化平台软件接收到该请求后，可创建、启动从架构虚拟机，从架构虚拟机中运行有支持从架构指令集的操作系统和应用程序。用户只需要在原有的虚拟化平台上启动支持新指令集的虚拟机，不需要对主架构虚拟机的应用程序和操作系统做全部的替换，就可以实现计算节点对不同指令集架构的虚拟机的兼容，相比于全部替换降低了成本。

为了让从架构虚拟机能正常运行，关键在于让从架构虚拟机的指令能够被硬件层的处理器执行，尤其是从架构下的异常或中断的处理。

由于处理器只支持单一指令集架构（主架构），无法识别从架构虚拟机的指令的含义，所以现有指令翻译方案是使用仿真器将从架构虚拟机的指令翻译成主指令集架构下的指令，再由仿真器通过纯软件模拟的方式处理翻译后的指令。仿真器可在宿主机层实现。对于一条从架构虚拟机的指令，指令翻译方案首先将从架构虚拟机的指令翻译成仿真器的抽象的内部表示，然后仿真器将抽象的内部表示翻译成语义等价的主架构的指令。翻译得到的主架构的指令与从架构虚拟机的指令具有相同的语义。

翻译得到的主架构指令的执行过程与从架构虚拟机的指令类型有关。翻译前后的指令的类型是相同的，也就是说如果从架构虚拟机的指令是非特权指令，那么翻译得到的主架构指令也是非特权指令，若从架构虚拟机的指令是特权指令，那么翻译得到的主架构指令也是特权指令。如果从架构虚拟机的指令是非特权指令，翻译得到的主架构指令可被直接执行实现仿真效果；如果从架构虚拟机的指令是特权指令，则需要仿真器协助处理翻译得到的主架构指令。其中，特权指令是与系统安全相关的一类指令。在计算机系统中，有的指令与系统安全相关，比如内存清零指令。如果某程序可以使用这类指令，这意味着该程序可以将其他程序的内存数据随意清零。因此计算机系统对指令加以分类，将指令分为特权指令（如内存清零指令）和非特权指令（如普通的运算指令）。特权指令是拥有特殊权限的指令，用于调用系统函数或系统软件等，例如清理内存、设置时钟、分配系统资源、修改虚拟内存的段表和页表、修改用户的访问权限等。

若翻译得到的主架构指令是特权指令，且用于请求访问虚拟硬件资源，支持主架构的仿真器需要按照从架构的硬件设备规范模拟从架构虚拟机访问虚拟硬件资源的执行过程。从架构的硬件设备规范和主架构的硬件设备规范因各自支持的指令集架构的不同而不同。

若翻译得到的主架构指令进一步触发了异常，则由主架构下的仿真器通过软件模拟的方式仿真从架构下处理异常的执行过程。需要说明的是，在指令翻译方案里由于从架构虚拟机的指令直接由仿真器来翻译并通过软件模拟得以被处理，仿真器是在宿主机实现的，是全模拟的虚拟化方案，并没有采用硬件辅助虚拟化方案中的硬件加速模块(例如 MMU 中用于虚拟地址到物理地址转换的页表、中断控制器)，所以给虚拟机呈现的硬件资源都是纯软件实现的。在硬件辅助虚拟化方案中，需要虚拟机监视器参与处理异常或中断，而仿真器实现的指令翻译方案中不涉及需要主架构的虚拟机监视器识别、处理在从架构下触发的异常或中断的问题。但是指令翻译方案首先会遇到指令数量增大的问题，需要多条主架构指令才能实现与从架构虚拟机指令等价的语义，例如一条从架构虚拟机的指令通过指令翻译后，会得到多条主架构的指令来表示与该条从架构虚拟机指令相同的语义；其次，仿真器需要判断翻译得到的主架构指令是否触发异常，仿真器需要额外

的判断逻辑；再者，仿真器是通过全软件模拟的方式来模拟从架构虚拟机访问虚拟硬件资源的处理逻辑，这种全软件模拟的方式的性能较差。

各厂商针对各自使用的指令集架构提出了硬件辅助虚拟化技术，硬件辅助虚拟化技术借助硬件模块加速虚拟机访问虚拟硬件资源的过程，提高虚拟机的性能。例如扩展页表（extended page tables，EPT）用以加速 GPA 到 HPA 的转换过程； Intel® 的 VT-X（virtualization technology）技术引入新的特权指令和运行模式，加速虚拟机特权指令的执行；中断控制器支持处理虚拟中断，可加速对虚拟机的中断的处理。在采用硬件辅助虚拟化技术的虚拟化场景下触发异常或中断时，由虚拟机监视器处理异常或中断。处理器从硬件寄存器中读取虚拟机监视器在寄存器中注册的异常或中断的信息，并通知虚拟机监视器来处理该异常或中断。虚拟机监视器需要根据异常或中断的状态信息来处理异常或中断。

异常的状态信息包括异常的类型、触发异常的指令和触发异常的地址。异常（exception）是指令执行失败后的处理请求，例如有一种类型的异常为缺页异常，是虚拟机在访问虚拟内存中发生的异常。虚拟机监视器需要处理该异常，就需要获取触发缺页异常的虚拟内存的地址。触发异常的指令是指当异常发生时，当前正在执行的指令，例如有一种类型的异常为非法指令异常，则该非法指令就是触发异常的指令。虚拟机监视器获取到触发异常的指令就可获知哪条指令触发了异常，以便处理该异常。

中断的状态信息包括中断的类型。常见的中断类型有核间中断、局部中断（包括时钟中断，性能监控中断（performance monitoring interrupt，PMI），外部中断（主要是磁盘、网卡等外设触发的中断）。例如从架构虚拟机执行某个指令时，若从架构处理逻辑收到来自外部设备的一个外部中断，或者从架构虚拟机的虚拟处理器之间发送的核间中断，这些中断可称为从架构的中断，即在从架构的软件或硬件环境中触发中断。

虚拟机监视器需要获取异常或中断的类型，才能针对具体类型分别处理。用于表示异常或中断的类型的编码在不同指令集架构的规范下具有不同的定义。对于同一类中断或异常，在不同指令集架构下的编码是不同的，比如编码的值可能不同，存储编码值的格式也可能不同。存储编码值的格式可以理解为存储编码值的字段和字段数量不同。例如在 RISC-V 架构下，表示 stage 2 指令缺页异常的编码及编码格式为“interrupt=（0），Exception Code=（20）”，编码值为 0 和 20，通过 interrupt 和 Exception Code 两个字段来表示，stage 2 指令缺页异常在 ARM 架构下的编码及编码格式为“RES0 = (0x0)，EC = (0x20)，IL = (0x1)，ISS = (0x8E)”，编码值为 0x0、0x20、0x1 和 0x8E，通过 RES0、EC、IL 和 ISS 四个字段来记录这些编码值。

进一步的，在不同指令集架构中，存储异常或中断的状态信息的寄存器也是不同的。例如在 ARM 架构下的异常或中断的状态信息存储在符合 ARM 架构规范的寄存器里；在 RISC-V 架构下的异常或中断的状态信息存储在符合 RISC-V 架构规范的寄存器里。支持 ARM 架构的虚拟机监控器从符合 ARM 架构规范的寄存器中获得编码，支持 RISC-V 架构的虚拟机监控器从符合 RISC-V 架构规范的寄存器中获得编码，分别确定异常或中断的具体类型并针对不同类型分别处理。

所以在支持异构指令集架构虚拟机的系统中，在从架构虚拟机发生中断或异常时，表示中断或异常的类型的编码无法被支持主指令集架构的主架构虚拟机监控器识别，主

架构虚拟机监控器无法准确的识别出该中断或异常的具体类型。并且主架构虚拟机监控器也获取不到处理异常或中断所需的状态信息。这导致主架构虚拟机监控器无法对从架构虚拟机触发的异常或中断进行处理，异常或中断的处理的缺失导致硬件辅助虚拟化技术不能应用在支持异构指令集架构虚拟机的场景中。

本申请实施例提供的系统架构中，处理器（例如计算节点 203 的处理器）可支持多种指令集架构，可以直接执行主架构指令集的指令和从架构指令集的指令。因此无需对从架构指令进行指令翻译，避免了指令翻译带来的指令膨胀问题。从架构下触发异常或中断时，本申请根据存储的映射关系对该异常或中断的编码进行转换，转换后的编码符合主架构规范，转换后的编码为主架构规范中对该类异常或中断的定义。本申请找到主架构规范中用于表示该类异常或中断的类型的编码，使得主架构的虚拟机监视器可以识别从架构下的异常或中断，并且可以复用主架构下的支持硬件辅助虚拟化的硬件模块进行处理。

图 3 为本申请实施例提供的计算节点 300 的另一种结构示意图。图 3 所示的计算节点 300 可以为图 1 和图 2 中的计算节点的部分或全部。

计算节点 300 可划分为软件层和硬件层。硬件层包括处理器 308 和中断控制器 312；软件层包括主架构虚拟机 301 和从架构虚拟机 302 以及虚拟机监视器 303。计算节点 300 上运行的主架构虚拟机和从架构虚拟机所支持的指令集架构不同，主架构虚拟机 301 支持主指令集架构（主架构），从架构虚拟机 302 支持从指令集架构（从架构）。从架构虚拟机 302 内执行的进程与主架构虚拟机 301 内运行的进程也支持不同指令集架构。虚拟机监视器 303 运行在主架构下，支持主指令集架构。从架构虚拟机 302 的运行环境由主架构的虚拟机监视器 303 准备、启动。

处理器 308 包括主架构处理逻辑 308 和从架构处理逻辑 309，处理器 308 可以是一个多核处理器。主架构处理逻辑 308 或从架构处理逻辑 309 可以是处理核，或称为 CPU 内核。主架构处理逻辑 308 可称为主处理核，从架构处理逻辑 309 可称为从处理核。处理逻辑还可以称为逻辑核，是同一个物理核内逻辑层面的核。对于多核处理器中主架构处理逻辑和从架构处理逻辑之间关系，可以参考后续图 10 及其对应描述。

主架构处理逻辑 309 和从架构处理逻辑 310 可以共同访问的地址空间（例如，共享内存）或者寄存器可以作为主架构和从架构之间的通信通道 311。处理器 308 中可用于执行主指令集架构的指令（主架构指令）的部分为主架构处理逻辑 309，用于执行从指令集架构的指令（从架构指令）的部分为从架构处理逻辑 310。因此处理器 308 可以识别两种指令集架构的指令的含义，无需将从架构指令翻译为可被处理器识别的主架构指令。对于没有触发中断或异常的指令，不论是特权指令还是非特权指令都可以被从架构处理逻辑 309 直接处理。

从架构虚拟机 302 触发异常或中断时，从架构处理逻辑 310 暂停从架构虚拟机 302 的执行，从架构处理逻辑 310 将异常或中断的状态信息存储在主从架构之间的通信通道 311 中，从架构处理逻辑 310 通知主架构的虚拟机监视器 303 处理异常或中断，等待虚拟机监视器 303 处理完中断或异常。（步骤①）从架构处理逻辑为支持从架构的处理单元，主架构处理逻辑为支持主架构的处理单元。

需要说明的是，从架构虚拟机触发的中断可包括从架构处理逻辑 310 接收的中断和

从架构虚拟机产生的虚拟中断；从架构处理逻辑 310 接收的中断是从架构虚拟机 302 运行过程中由主架构中断控制器 312 向从架构处理逻辑 310 发送的物理中断，或者称为硬件中断；从架构虚拟机产生的虚拟中断为从架构虚拟机的虚拟处理器触发的中断，例如虚拟处理器 VCPU1 向 VCPU2 发送的核间中断。从架构虚拟机 302 触发的异常指的是从架构虚拟机 302 运行过程中触发的异常，例如从架构虚拟机 302 访问虚拟内存触发的缺页异常或者从架构虚拟机 302 执行了非法指令所触发的非法指令异常。

虚拟机监视器 303 接收到通知后，转换模块 304 从通信通道 311 中获取从架构下触发的异常或中断的状态信息。此时转换模块 304 获取的异常或中断的状态信息中，表示异常或中断的类型的编码符合从架构规范，不符合主架构规范。转换模块 304 根据映射表查找到在主架构规范中用于表示该类型的中断或异常的编码，其中映射表中存储有主架构规范中表示不同异常或中断的类型的编码与从架构规范中表示不同异常或中断的类型的编码之间的对应关系。然后转换模块 304 将查找到的在主架构规范中用于表示该类型的中断或异常的编码以及触发异常的地址以及触发异常的指令保存在虚拟机监视器 303 可访问的主架构硬件寄存器或内存里。（步骤②）其中，主架构硬件寄存器指的是支持主架构硬件规范的硬件寄存器。

模拟模块 305 从上述主架构硬件寄存器或内存里获取状态信息，根据状态信息中的编码得知从架构虚拟机触发是异常还是中断以及中断或异常的具体类型。（步骤③）

若从架构虚拟机触发的是中断，中断模拟模块 307 从主架构硬件寄存器或内存里获取中断的具体类型，然后使用主架构的中断控制器 312 处理该类型的中断。（步骤④）

中断模拟模块 307 通过模拟符合从架构规范所定义的中断控制器的行为，向从架构虚拟机发送虚拟中断，例如管理虚拟中断的映射关系和优先顺序关系，以及虚拟中断的投递过程。中断模拟模块 307 可用于管理虚拟中断映射关系，虚拟中断映射关系包括从架构虚拟机的虚拟中断和从架构虚拟处理器的映射关系，以及从架构虚拟处理器和物理处理器（或物理核）的映射关系。中断模拟模块 307 可为每个从架构虚拟处理器管理多个虚拟中断的优先顺序关系。当有多个虚拟中断需要发送给虚拟处理器时，中断模拟模块 307 根据优先顺序关系获取具有最高优先级的虚拟中断。中断模拟模块 307 根据虚拟中断映射关系，获取从架构虚拟处理器所在的物理处理器（或物理核）。通过主架构中断控制器 312 发送中断通知该物理处理器（或物理核）。该物理处理器将该虚拟中断的信息存储到从架构处理逻辑的硬件寄存器内，完成该虚拟中断的投递。虚拟中断的信息包括：虚拟中断的类型。

若从架构虚拟机触发的是异常，异常模拟模块 306 从主架构硬件寄存器或内存里获取异常的具体类型，使用主架构处理逻辑 309 以及硬件辅助虚拟化能力处理从架构虚拟机的异常。（步骤⑤）

异常模拟模块 306 用于模拟从架构虚拟机运行过程中所触发的异常，根据主架构的异常和从架构异常的对应情况，分为三种类型：

1).若主架构虚拟机监视器 303 可处理与从架构的异常的类型相同的异常（映射表中可以找到与从架构的异常对应的主架构异常），例如访存异常，则可以直接复用主架构虚拟机监视器的异常处理方法，包括软件逻辑和硬件上的硬件辅助虚拟化能力。

2).若主架构虚拟机监视器 303 可处理与从架构的异常的类型相似的异常（映射表中可以找到与从架构的异常对应的主架构异常），例如 TLB 刷新指令触发的非法指令异常、

调试指令触发的非法指令异常，异常模拟模块 306 可以读取转换后的异常的状态信息，获取模拟该类型异常所需的信息，例如触发异常的从架构指令、从架构指令附带的参数信息。然后选取主架构相似功能的指令或者异常处理模块，并且将异常模拟所需的信息按照主架构指令的规范或者异常处理模块的输入参数规范进行填入，最后调用主架构的指令或异常处理模块处理。

3).若主架构虚拟机监视器 303 中没有与从架构异常相同或相似的异常类型（映射表中没有找到与从架构的异常对应的主架构异常），例如超级调用指令触发的异常、从架构特有的特权指令触发的异常、对虚拟设备的地址空间的访问异常，异常模拟模块 306 根据从架构的硬件规范采用纯软件模拟的方式处理。

模拟模块 305 完成从架构虚拟机的异常或中断的处理后，将处理结果存储在通信通道 311 里，并通知从架构处理逻辑 310 恢复执行。（步骤⑥）

从架构虚拟机被恢复执行后，从通信通道 311 中获取处理结果，继续执行后续指令。（步骤⑦）

下面以主架构为 ARM 架构，从架构可以为 RISC-V 架构为例，介绍本申请提供的异构指令集架构下虚拟机运行的方法，该方法可应用于计算节点 400，方法的执行流程如图 5 和图 6 所示。图 4 所示为计算节点 400 的结构，计算节点 400 的结构与图 3 所示的计算节点 300 类似。

计算节点 400 的硬件层包括处理器 411、ARM 中断控制器以及共享内存 412。

处理器 411 可包括能够处理 ARM 指令集的指令的 ARM 处理逻辑 409 和能够处理 RISC-V 指令集的指令的 RISC-V 处理逻辑 410。由于不同指令集架构下用于存储异常或中断的状态信息的硬件寄存器是不同的，因此本申请中处理器 411 里还包括符合 ARM 架构规范的一个或多个 ARM 寄存器 413，用于存储 ARM 架构下发生的异常或中断的类型、触发异常的地址和触发异常的指令；以及符合 RISC-V 架构规范的一个或多个 RISC-V 寄存器 414，用于存储 RISC-V 架构下的异常或中断的类型、触发异常的地址和触发异常的指令。其中，某指令集架构下的异常或中断或某指令集架构下触发的异常或中断指的是由支持该指令集架构的虚拟机触发的异常或者由支持该指令集架构的处理逻辑接收到的中断。存储有异常或中断的状态信息的 RISC-V 寄存器 414 或 ARM 寄存器 413 可具体为状态控制寄存器（Control and Status Registers，CSRs），状态控制寄存器可以有多个，分别用于存储异常或中断的类型、触发异常的指令和触发异常的地址。

由于 ARM 处理逻辑 409 不能直接读取 RISC-V 寄存器 414 来获得 RISC-V 架构下的异常或中断的状态信息，所以本申请实施例使用共享内存 412 来传输 RISC-V 架构下的异常或中断的状态信息。共享内存 412 可以被 ARM 处理逻辑和 RISC-V 处理逻辑访问，用于传输 RISC-V 架构下的异常或中断的类型、触发异常的地址和触发异常的指令，以及对 ARM 架构下 ARM Hypervisor 和硬件对该异常或中断的处理结果。

针对 RISC-V 架构下的状态控制寄存器 414、浮点寄存器和通用寄存器，共享内存 412 的设计规范里定义了用于存储各个寄存器的值的地址范围。换句话说，RISC-V 架构下的异常或中断的类型、触发异常的地址和触发异常的指令这些存储在一个或多个状态寄存器中的数据以及存储在浮点寄存器和通用寄存器中的数据在共享内存 412 中的存储地址是固定的。且 ARM 处理逻辑 409 可以从共享内存 412 中对应的存储地址获取 RISC-V

架构下的异常或中断的类型、触发异常的地址和触发异常的指令这些数据。在本申请实施例中具体是由 ARM Hypervisor 402 从共享内存 412 中获取这些数据。

计算节点 400 的软件层包括 ARM 虚拟机 401、RISC-V 虚拟机 403 以及 ARM Hypervisor 402。RISC-V 虚拟机 403 支持 RISC-V 指令集架构，ARM 虚拟机 401 支持 ARM 指令集架构，ARM Hypervisor 402 支持 ARM 指令集架构的 Hypervisor。

本申请中 ARM Hypervisor 402 中的 RISC-V 转换模块 404 可根据 ARM & RISC-V 映射表，确定与 RISC-V 架构下触发的异常或中断的类型对应的 ARM 架构下的异常或中断的类型，并且将 RISC-V 架构下的异常或中断的状态信息存储在对应的一个或多个 ARM 寄存器 413 里。

ARM & RISC-V 映射表可以包括异常映射表和中断映射表。

异常映射表用于记录 RISC-V 架构下一种或多种的异常的编码与 ARM 架构下一种或多种异常的编码的对应关系。异常映射表里记录的每一条对应关系是 ARM 规范中对一类异常的编码与 RISC-V 规范中对该类异常的编码。其中，异常的编码用于表示异常的类型，是一种类型的异常在一个指令集架构下唯一的编码，以区分不同类型的异常。RISC-V 架构下的一类异常的编码指的是在 RISC-V 指令集规范里，用于表示该类异常的类型；ARM 架构下的一类异常的编码指的是在 ARM 指令集规范里，用于表示该类异常的类型。因此，RISC-V 转换模块 404 可以根据该异常映射表将从架构下的异常的编码转换成可被 ARM Hypervisor 402 识别的异常的编码，ARM Hypervisor 402 可识别出从架构下的异常的具体类型。

异常映射表还用于记录一个或多个 RISC-V 寄存器 414 与一个或多个 ARM 寄存器 413 的对应关系。例如 RISC-V 架构下存储触发异常的指令的寄存器与 ARM 架构下用于存储触发异常的指令的状态控制寄存器的对应关系，RISC-V 架构下存储触发异常的地址的寄存器与 ARM 架构下用于存储触发异常的地址的状态控制寄存器的对应关系，以及 RISC-V 架构下存储异常编码的寄存器与 ARM 架构下用于存储异常编码的寄存器的对应关系。因此，处理异常所需的状态信息可被存储在 ARM 寄存器 413 中，以便 ARM Hypervisor 402 从 ARM 寄存器 413 获取处理异常所需的状态信息并处理该异常。

中断映射表用于记录 RISC-V 架构下一种或多种类型的中断的编码与 ARM 架构下一种或多种类型的中断的编码的对应关系。中断映射表里记录的每一条对应关系是 ARM 规范中对一类中断的编码与 RISC-V 规范中对该类中断的编码。其中，中断的编码表示中断的类型，是一类中断在一个指令集架构下唯一的编码，以区分不同类型的中断。RISC-V 架构下的一类中断的编码指的是在 RISC-V 指令集规范里，用于表示该类中断的类型；ARM 架构下的一类中断的编码指的是在 ARM 指令集规范里，用于表示该类中断的类型。因此，RISC-V 转换模块 404 可以根据该中断映射表将从架构下的中断的编码转换成可被 ARM Hypervisor 402 识别的中断的编码，ARM Hypervisor 402 可识别出从架构下的中断的具体类型。

中断映射表还用于记录 RISC-V 架构下用于存储中断编码的 RISC-V 寄存器 414 与 ARM 架构下存储中断编码的 ARM 寄存器 413 的对应关系。因此，处理中断所需的状态信息可被存储在 ARM 寄存器 413 中，以便 ARM Hypervisor 402 从 ARM 寄存器 413 获取处理中断所需的状态信息并处理该中断。

图 5 表示对 RISC-V 虚拟机触发的异常的处理流程。

步骤 501：当 RISC-V 虚拟机触发异常时，处理器 411 通知 ARM Hypervisor 处理该异常；：

RISC-V 虚拟机请求访问 GVA 获取存储在内存的数据时，由于虚拟机最终要根据 HPA 里获取数据，MMU 会将 GVA 转换成 GPA，再将 GPA 转换成 HPA。如果 GPA 到 HPA 的转换过程出现异常，例如请求访问的是非法的虚拟地址或者请求访问的虚拟地址虽然合法但该虚拟地址还未被分配物理页（没有建立 GPA 到 HPA 的映射表），可能会触发缺页异常（Page Fault）。由于 ARM Hypervisor402 不支持 RISC-V 架构且没有支持 RISC-V 规范的 MMU，所以该异常无法在 RISC-V 架构下被处理，需要由 ARM Hypervisor402 处理该异常。

步骤 502： RISC-V 虚拟机暂停执行，等待 ARM Hypervisor 处理异常。RISC-V 处理逻辑 410 将系统状态保存在共享内存中。

系统状态包括状态控制寄存器、通用寄存器以及浮点寄存器中的信息。通用寄存器和浮点寄存器中可存储有 RISC-V 虚拟机的 vCPU 的运行状态信息，例如指令运算过程中产生的数据，系统临时申请的局部变量以及浮点计算的中间结果。

代表异常类型为缺页异常的编码和触发缺页异常的虚拟地址（比如 GVA）可分别存储在不同的 RISC-V 状态控制寄存器（RISC-V 寄存器 414）里，触发缺页异常的地址指的是触发缺页异常时 RISC-V 虚拟机 403 请求访问的虚拟地址。

共享内存 412 里为状态控制寄存器、通用寄存器以及浮点寄存器中的信息分别设置了存储地址，例如相对于共享内存 412 的起始地址的偏移量为 0x1000-0x13e0 的地址范围内用于存储通用寄存器以及浮点寄存器中的信息；相对于共享内存的起始地址的偏移量为 0x2000-0x11140 的地址范围用于存储一个或多个状态控制寄存器中的信息，也就是用于存储缺页异常的编码和触发缺页异常的地址。具体的，共享内存 412 中设置了地址映射表，地址映射表中存储有状态控制寄存器、通用寄存器以及浮点寄存器的地址。ARM Hypervisor 就可以根据地址映射表中记录的各个寄存器的地址来读取寄存器中的值或者更新寄存器中的值。

步骤 503： ARM Hypervisor402 从共享内存 412 中获取异常的状态信息；

RISC-V 虚拟机产生缺页异常，处理器 411 通知 ARM Hypervisor402 处理该缺页异常后，ARM Hypervisor 在 ARM 架构下的硬件寄存器中写入一个异常处理函数入口地址，处理器 411 会从该硬件寄存器中读取异常处理函数入口地址，执行从共享内存 412 中获取异常的状态信息的操作。

ARM Hypervisor 根据共享内存 412 中记录的状态控制寄存器（RISC-V 寄存器 414）的地址，获取表示 RISC-V 虚拟机触发的异常的类型的编码。

步骤 504： RISC-V 转换模块 404 根据异常映射表，查找与 RISC-V 架构下的异常的类型对应的、ARM 架构下的异常类型。

由于 RISC-V 架构和 ARM 架构对于表示异常类型的编码的定义不同，用于表示 RISC-V 虚拟机触发的异常的类型的编码不符合 ARM 架构的规范，所以 ARM

Hypervisor402 无法直接根据从共享内存里获得的异常的编码来识别 RISC-V 虚拟机触发的异常的类型。所以 RISC-V 转换模块 404 根据异常映射表查找到与符合 RISC-V 架构规范的异常编码对应的、符合 ARM 架构规范定义的异常编码，从而识别出 RISC-V 虚拟机触发的异常的类型。这里说的 RISC-V 架构和 ARM 架构对于表示异常类型的编码的定义不同，指的是对于同一类型的异常，两种指令集架构的规范采用不同的编码值来表示，并且编码的格式也不同。RISC-V 架构规范中，采用两个字段的格式来表示异常，与 RISC-V 架构的规范对缺页异常的编码的格式的定义不同，ARM 架构规范采用四个字段表示缺页异常。

RISC-V 架构规范中，用于表示异常和中断的类型的编码存储在 scause 寄存器中。如图 7 所示为 scause 寄存器的结构图，scause 寄存器的 1 位的 interrupt 的值为 1 时表示中断，interrupt 值为 0 时表示异常。scause 寄存器的 0 位的异常码（Exception Code）就是用于表示异常或中断的具体类型。以 stage 2 指令缺页异常为例，scause 寄存器的 1 位的 interrupt 值为 0，scause 寄存器的 0 位中 Exception Code 的值为 20，也就是说 stage 2 指令缺页异常的编码为 interrupt=（0），Exception Code=（20）。

ARM 架构规范中，用于表示异常或中断的类型的编码存储在 esr_el2 寄存器中。esr_el2 寄存器的结构如图 8 所示。RES0 是预留字段，值通常为 0；EC（Exception Class）字段表示异常大类，IL（Instruction Length for synchronous exceptions）表示指令的长度，ISS（Instruction Specific Syndrome）字段用于记录异常大类下的子类以及一些特殊字段。以缺页异常为例，ARM 架构下 stage 2 指令缺页异常的编码为“RES0 = (0x0)，EC = (0x20)，IL = (0x1)，ISS = (0x8E)”。

RISC-V 架构规范中，用于存储触发异常的地址的寄存器为 htval 寄存器。ARM 架构规范中，用于存储触发异常的地址的寄存器为 hpfar_el2 寄存器。

scause 寄存器、esr_el2 寄存器、htval 寄存器和 hpfar_el2 寄存器是用于存储不同信息的状态控制寄存器。

异常映射表里记录了 RISC-V 架构多类型异常的编码与 ARM 架构下多类型异常的编码的对应关系，异常的编码代码异常的类型，表中一个 RISC-V 架构下的异常的编码对应一个 ARM 架构下的异常的编码，多条对应关系中的每一条对应关系表示 RISC-V 架构下一类异常的编码与 ARM 架构下该类异常的编码的对应关系。

以 stage 2 指令缺页异常为例，异常映射表里记录有异常编码“interrupt=（0），Exception Code=（20）”与异常编码“RES0 = (0x0)，EC = (0x20)，EL = (0x1)，ISS = (0x8E)”的对应关系。

RISC-V 转换模块 404 从共享内存 412 中获取了 RISC-V 架构下 stage 2 指令缺页异常的编码“interrupt=（0），Exception Code=（20）”，根据异常映射表查找到与编码“interrupt=（0），Exception Code=（20）”对应的编码“RES0 = (0x0)，EC = (0x20)，EL = (0x1)，ISS = (0x8E)”。编码“RES0 = (0x0)，EC = (0x20)，EL = (0x1)，ISS = (0x8E)”符合 ARM 架构规范，所以 ARM Hypervisor402 能够根据该编码识别出 RISC-V 架构下触发的异常类型是 stage 2 指令缺页异常。

步骤 505，异常模拟模块 406 将转换后的异常编码、触发异常的指令和触发异常的地址写入对应的 ARM 寄存器里或者写入到 ARM Hypervisor 可访问的内存中。

在一种实现方式中，考虑到对 ARM 架构的原始代码的改动要尽量小，避免做太多侵入式修改，所以由 ARM Hypervisor 中的转换模块将转换后的异常编码、触发异常的指令和触发异常的地址写入 ARM 寄存器里，以便 ARM Hypervisor 中的异常模拟模块 406 可以从 ARM 寄存器里获取处理异常所需的上述信息。

异常映射表记录了 RISC-V 架构下用于存储异常编码的寄存器与 ARM 架构下用于存储异常编码的寄存器的对应关系，以及 RISC-V 架构下用于存储触发异常的地址的寄存器与 ARM 架构下用于存储异常编码的寄存器的对应关系。以 stage 2 指令缺页异常为例，异常映射表记录了 scause 寄存器与 esr_el2 寄存器的对应关系，以及和 htval 寄存器与 hpfar_el2 寄存器的对应关系。因此异常模拟模块 406 可以将与 scause 寄存器里存储的编码对应的编码写到 esr_el2 寄存器中，即把编码“RES0 = (0x0)，EC = (0x20)，EL = (0x1)，ISS = (0x8E)”esr_el2 寄存器；并且可以把从共享内存中获取到的 htval 寄存器里的值（触发异常的地址）写入到 hpfar_el2 寄存器里。其中，编码和触发异常的地址按照 ARM 架构的规范写入到寄存器中相应的字段中。

在缺页异常里有个特殊的场景是访问内存地址空间 MMIO 的触发的缺页异常，要处理这类异常 ARM Hypervisor 需要获取触发异常的指令。在 RISC-V 架构下，触发异常的指令存储在 htinst 寄存器（一种状态控制寄存器）。由于 ARM 架构这边没有与 htinst 寄存器对应的专门用于存储触发异常的指令的寄存器，因此 ARM Hypervisor 可以把从 ARM & RISC-V 的共享内存读取的触发异常的指令的这一信息存储在通用寄存器或者划分给 ARM Hypervisor 的内存里。

在另一种实现方式中，ARM Hypervisor 将转换后的异常编码、触发异常的指令和触发异常的地址写入 ARM Hypervisor 可访问的内存里，以便异常模拟模块 406 可以直接获得这些处理异常所需的信息。

以 RISC-V 虚拟机执行 TLB 刷新指令或 EBREAK 时指令触发的非法指令异常为例，该异常的状态信息包括异常的类型、触发异常的指令、指令附带的参数信息（例如地址空间标识符（Address Space Identifier，ASID）和虚拟地址范围）。异常模拟模块按照 ARM 架构的规范，将上述信息写入 ARM 架构下用于存储这些信息的寄存器中或者存储在内存中。

步骤 506：ARM Hypervisor 根据异常的编码识别并处理该异常；

转换后的异常编码为 RISC-V 虚拟机触发的异常在 ARM 架构下的编码，因此 ARM Hypervisor 根据转换后得到的异常的编码识别并处理 RISC-V 虚拟机触发的异常。

以缺页异常为例，异常模拟模块 406 调用 ARM Hypervisor 具有的数据终止（data abort）异常处理逻辑来处理 RISC-V 架构下虚拟机访问 GVA 所触发的缺页异常。例如缺页异常是因为 GPA 到 HPA 的转换过程出现异常，则 ARM Hypervisor 建立 GPA 到 HPA 的映射关系，其中该映射关系可以记录在 stage 2 地址转换页表中，stage 2 地址转换页表在 ARM 架构下的 MMU 中用于将 GPA 转换至 HPA。处理器再次执行触发缺页异常的指令或重新访问触发缺页异常的地址，这时 MMU 可通过重新建立的表示 GPA 到 HPA 映射关系的 stage 2 地址转换页表找到 HPA，获得存储在内存中的数据。由此复用了硬件辅助虚拟化技术中的 stage 2 地址转换页表对访问内存数据进行加速处理。

例如 TLB 刷新指令触发的异常，ARM Hypervisor 可使用已有的处理方式完成对

RISC-V 虚拟机的 TLB 刷新操作。对于 ECALL 指标触发的非法指令异常，解析触发异常的 RISC-V 架构的 ECALL 指令，获取 ECALL 指令的具体调用类型，及指令的入参。采用纯软件模拟的方式，模拟 RISC-V 架构的 ecall 指令执行。

对于 ARM Hypervisor 中没有相同或相似异常处理功能的 RISC-V 架构的异常，例如 ECALL 指令触发的异常、RISC-V 架构特有的特权指令触发的异常、对虚拟设备的地址空间的访问异常。在 ARM Hypervisor 中按照 RISC-V 架构的硬件规范采用纯软件模拟方式，模拟 RISC-V 架构异常。以 ECALL 指令异常为例，该异常是要调用 RISC-V 架构中虚拟机监视器提供的一个接口来完成某个功能时触发的。由于系统中没有支持 RISC-V 架构的虚拟机监视器，且在 ARM Hypervisor 中也没有的该接口。那么就参考 RISC-V 架构中的定义，在 ARM Hypervisor 中再实现一个相同功能的接口。

步骤 507：ARM Hypervisor 将对异常的处理结果存储在共享内存中，传递给 RISC-V 虚拟机。

对于缺页异常，处理结果为触发缺页异常时 RISC-V 虚拟机所要访问的内存中的数据。ARM Hypervisor 将 RISC-V 虚拟机要访问的数据存储在共享内存中，传递给 RISC-V 虚拟机。RISC 处理逻辑恢复 RISC-V 虚拟机的状态，继续运行 RISC-V 虚拟机。

由于对于同一类异常，不同指令集架构中用于表示该异常的类型的编码是不同的，所以 ARM Hypervisor 不能直接识别出 RISC-V 虚拟机触发的异常的类型。在本申请实施例中，通过存储有 ARM 架构定义的多类异常各自的编码和 RISC-V 架构定义的多类异常各自的编码之间的一一对应关系，ARM Hypervisor 可以找到 RISC-V 虚拟机触发的异常的类型在 ARM 架构下的编码，从而 ARM Hypervisor 可以识别 RISC-V 虚拟机触发的异常的类型。换句话说，ARM Hypervisor 根据用于表示 RISC-V 虚拟机触发的异常的类型的 RISC-V 编码以及对应关系，找到表示 RISC-V 虚拟机触发的异常的类型的 ARM 编码；其中，RISC-V 编码是 RISC-V 架构的规范中对该类异常的编码，ARM 编码是 ARM 架构的规范中对该类异常的编码，ARM Hypervisor 是可以直接识别 ARM 编码来确定异常的类型。

图 6 表示对 RISC-V 虚拟机触发的中断的处理流程，具体包括：

步骤 601：当 RISC-V 虚拟机触发中断时，处理器 411 通知 ARM Hypervisor 处理中断；

在本申请的架构中，由于硬件层没有符合 RISC-V 规范的中断控制器，所以当 ARM 中断控制器 409 向 RISC-V 处理逻辑 410 发送物理中断或者 RISC-V 虚拟机的 vCPU1 向 vCPU2 发送核间中断时，RISC-V 处理逻辑 410 需要通知 ARM Hypervisor 402 来处理该中断。RISC-V 虚拟机触发的中断包括 RISC-V 处理逻辑 410 接收的物理中断，以及 RISC-V 虚拟机中虚拟处理器接收的虚拟中断。

步骤 602： RISC-V 虚拟机暂停执行，等待 ARM Hypervisor 处理完中断。RISC-V 处理逻辑将当前的系统状态保存在共享内存中。

系统状态包括 RISC-V 架构下的通用寄存器、浮点寄存器、状态控制寄存器等硬件寄存器中存储的信息。状态控制寄存器中存储的是处理中断所需的中断的状态信息，在 RISC-V 架构下，状态控制寄存器（RISC-V 寄存器 414）具体为虚拟化管理模式中断寄存器（Virtual Supervisor Interrupt Registers，VSIP）。通用寄存器、浮点寄存器中存储有

RISC-V 虚拟机当前的 vCPU 的状态信息，例如指令运算过程中的数据，临时申请的局部变量以及浮点计算的中间结果。通用寄存器、浮点寄存器中存储的 vCPU 的状态信息可以用于 RISC-V 虚拟机 403 切换 vCPU 时使用。

中断的状态信息包括中断的类型，具体来说，包括代表中断的类型的编码，且该编码符合 RISC-V 架构规范。RISC-V 处理逻辑从 RISC-V 状态控制寄存器 414 中读取代表中断类型的编码，并将该编码存储在共享内存 412 中。

共享内存 412 里为状态控制寄存器、通用寄存器以及浮点寄存器中的信息分别设置了存储地址，例如相对于共享内存 412 的起始地址的偏移量为 0x1000-0x13e0 的地址范围内用于存储通用寄存器以及浮点寄存器中的信息；相对于共享内存的起始地址的偏移量为 0x2000-0x11140 的地址范围用于存储一个或多个状态控制寄存器中的信息，也就是用于存储代表中断的类型的编码。具体的，共享内存 412 中设置了地址映射表，地址映射表中存储有状态控制寄存器、通用寄存器以及浮点寄存器的地址。ARM Hypervisor 就可以根据地址映射表中记录的各个寄存器的地址来读取寄存器中的值或者更新寄存器中的值。

步骤 603： ARM Hypervisor 从共享内存中获取中断的状态信息；

ARM Hypervisor 根据共享内存 412 中记录的状态控制寄存器（RISC-V 寄存器 414）的地址，获取表示 RISC-V 虚拟机触发的中断的类型的编码。

步骤 604：RISC-V 转换模块在中断映射表中查找与 RISC-V 架构下的中断的类型对应的、ARM 架构下的中断类型。

RISC-V 中断类型可识别为时钟中断，软件中断，以及外部中断。其中，核间中断属于软件中断中的一种。

由于 RISC-V 架构和 ARM 架构对于表示中断类型的编码的定义不同，用于表示 RISC-V 虚拟机触发的中断的类型的编码不符合 ARM 架构的规范，所以 ARM Hypervisor402 无法直接根据从共享内存里获得的中断的编码来识别 RISC-V 虚拟机触发的中断的类型。所以 RISC-V 转换模块 404 根据中断映射表查找到与符合 RISC-V 架构规范的中断编码对应的、符合 ARM 架构规范定义的异常编码，从而识别出 RISC-V 虚拟机触发的中断的类型。这里说的 RISC-V 架构和 ARM 架构对于表示中断类型的编码的定义不同，指的是对于同一类型的中断，两种指令集架构的规范采用不同的编码值来表示，并且编码的格式也不同。

中断映射表中记录了 RISC-V 架构下多类型中断的编码与 ARM 架构下多类型中断的编码的对应关系。中断的编码代表中断的类型，中断映射表中一个 RISC-V 架构下的中断的编码对应一个 ARM 架构下的中断的编码。

步骤 605，将转换后的中断编码写入对应的 ARM 寄存器里或者写入到 ARM Hypervisor 可访问的内存中。

在一种实现方式中，考虑到对 ARM 架构的原始代码的改动要尽量小，避免做太多侵入式修改，所以由 ARM Hypervisor 中的转换模块将转换后的中断编码写入 ARM 寄存器里，以便 ARM Hypervisor 中的中断模拟模块 407 可以从 ARM 寄存器里获取处理异常所

需的上述信息。

中断映射表还记录了RISC-V架构下用于存储中断编码的寄存器与ARM架构下用于存储中断编码的寄存器的对应关系。

步骤606：ARM Hypervisor根据转换后的中断的编码识别并处理该中断。

转换后的中断编码为RISC-V虚拟机触发的中断在ARM架构下的编码，因此ARM Hypervisor根据转换后得到的中断的编码识别并处理RISC-V虚拟机触发的中断。

中断模拟模块获取RISC-V虚拟处理器的编号，由虚拟处理器的编号获取虚拟处理器所在的物理处理器或物理核，通过ARM中断控制器通知该物理处理器或物理核。中断模拟模块维护有RISC-V虚拟中断和RISC-V虚拟处理器的映射关系，以及RISC-V虚拟处理器和物理处理器/物理核的映射关系。

中断模拟模块同时为每个RISC-V虚拟处理器维护RISC-V虚拟中断的优先顺序关系。当有多个中断同时发送给该虚拟处理器时，中断模拟模块根据维护优先顺序关系获取具有最高优先级的中断。中断模拟模块根据虚拟中断映射关系，获取RISC-V虚拟处理器所在的物理处理器/物理核。通过ARM中断控制器发送中断通知该物理处理器/物理核。ARM Hypervisor将具有最高优先级的中断设置到RISC-V处理器逻辑的硬件寄存器内，完成该虚拟中断的投递。

当ARM Hypervisor识别出RISC-V虚拟机触发的是核间中断，中断模拟模块解析获得的中断的状态信息，识别出是核间中断，且要发送给VCPU2。中断模拟模块根据虚拟中断映射关系得知vCPU2位于物理核2上，则通过ARM中断控制器发送核间中断至物理核2，从而完成RISC-V虚拟机触发的核间中断。

当ARM Hypervisor识别出RISC-V虚拟机触发的是时钟中断，ARM Hypervisor将RISC-V的VSIP（RISC的状态控制寄存器）的管理模式时钟中断等待位（Supervisor-level Time Interrupt Pending，STIP）置位，完成RISC-V时钟中断的投递。

当ARM Hypervisor识别出RISC-V虚拟机触发的是软件中断，则ARM Hypervisor将RISC-V处理逻辑的VSIP寄存器的管理模式软件中断等待位（Supervisor-level Software Interrupt Pending，SSIP）置位，完成RISC-V软件中断的投递。

当ARM Hypervisor识别出RISC-V虚拟机触发的是外部中断，则ARM Hypervisor将RISC-V处理逻辑的VSIP寄存器的管理模式外部中断等待位（Supervisor-level external interrupt pending，SEIP）置位，完成RISC-V外部中断的投递。

由于共享内存412中设置了地址映射表，地址映射表中存储有状态控制寄存器（例如VSIP）的地址，SSIP、STIP和SEIP位于VSIP中不同的比特位，置位SSIP、STIP或SEIP表示将对应比特位的值设为1，SSIP、STIP和SEIP这些比特位上的值为1代表对于的中断在等待处理。ARM Hypervisor可以根据地址映射表中记录的VSIP的地址来将VSIP中SSIP、STIP或SEIP的值设为1，以使得RISC处理逻辑根据VSIP中比特位上的值触发对应的中断。例如将STIP的位置位，是ARM Hypervisor根据地址映射表中VSIP的地址，将VSIP中STIP的值设置为1，表示时钟中断等待处理。RISC处理逻辑通过读取VSIP中的STIP的值，得知RISC-V虚拟机触发的中断类型为时钟中断。

步骤607:RISC处理逻辑恢复RISC-V虚拟机的状态，继续运行RISC-V 虚拟机。

RISC处理逻辑通过读取VSIP中的SSIP、STIP或SEIP的值，得知RISC-V触发的

中断的类型。

由于对于同一类中断，不同指令集架构中用于表示该中断的类型的编码是不同的，所以 ARM Hypervisor 不能直接识别出 RISC-V 虚拟机触发的中断的类型。在本申请实施例中，通过存储有 ARM 架构定义的多类中断各自的编码和 RISC-V 架构定义的多类中断各自的编码之间的一一对应关系，ARM Hypervisor 可以找到 RISC-V 虚拟机触发的中断的类型在 ARM 架构下的编码，从而 ARM Hypervisor 可以识别 RISC-V 虚拟机触发的中断的类型。换句话说，ARM Hypervisor 根据用于表示 RISC-V 虚拟机触发的中断的类型的 RISC-V 编码以及对应关系，找到表示 RISC-V 虚拟机触发的中断的类型的 ARM 编码；其中，RISC-V 编码是 RISC-V 架构的规范中对该类中断的编码，ARM 编码是 ARM 架构的规范中对该类中断的编码，ARM Hypervisor 是可以直接识别 ARM 编码来确定中断的类型。

在另一种实现方式中，若对于 RISC-V 虚拟机触发的异常或中断，ARM 架构的规范中没有对于该类异常或中断的编码。则可以采用以下的处理方式：

1）按照 ARM 的编码规范，自定义 ARM 架构下该类异常或中断的编码。RISC-V 架构下该类异常或中断的编码与 ARM 架构下自定义的该类异常或中断的编码的对应关系增加至异常映射表或中断映射表中。

2）直接在 ARM Hypervisor 中新增功能模块，新增的功能模块类似支持 RISC-V 架构的 Hypervisor。所以新增的功能模块可以参考 RISC-V Hypervisor 直接处理异常或中断时的处理逻辑来处理 RISC-V 虚拟机触发的异常或中断，也无需进行编码的映射。

图 9 为计算节点 700 的结构示意图。计算节点 700 可以为前述计算节点 100、203、300 或 400 的部分或全部。计算节点 700 包括处理器 701、存储器 702 和通信接口 703。

处理器 701 是计算节点 700 的控制中心，利用各种接口和总线连接计算节点 700 的各个部件。在一些实施例中，处理器 701 可包括一个或多个处理单元，或称为物理核，例如图 1 中的处理器 114 包括核 0 和核 1。在一些实施例中，处理器 701 可包括主架构处理逻辑和从架构处理逻辑，如图 3 中的主架构处理逻辑 309 和从架构处理逻辑 310 以及图 4 中的 ARM 处理逻辑 409 和 RISC-V 处理逻辑 410。处理器 701 还可以包括寄存器，寄存器可用于存储触发的异常或中断的状态信息。并且在一些实施例中，主架构处理逻辑和从架构处理逻辑具有各自的寄存器，主架构处理逻辑的寄存器可用于存储转换后的异常或中断的转换后的状态信息。处理器 701 可以是中央处理单元(Central Processing Unit，CPU)，该处理器 701 还可以是其他通用处理器、数字信号处理器（Digital Signal Processor，DSP)、专用集成电路（Application Specific Integrated Circuit，ASIC)、现成可编程门阵列(Field-Programmable Gate Array，FPGA）或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件等。通用处理器可以是微处理器或者该处理器 701 也可以是任何常规的处理器等。

存储器 702 中存储有计算机程序。处理器 701 被配置用于执行存储器 702 中的计算机程序，从而实现该计算机程序定义的功能。存储器 702 为非易失性存储介质，一般包括内存和外存。内存包括但不限于随机存取存储器（Random Access Memory，RAM），只读存储器（Read-Only Memory, ROM），或高速缓存（cache）等。外存包括但不限于闪存（flash memory）、硬盘、光盘、通用串行总线（universal serial bus, USB）盘等。计

算机程序通常被存储在外存上，处理器在执行计算机程序前会将该程序从外存加载到内存。 存储器 702 可以是独立的，通过总线与处理器 701 相连接；存储器 702 也可以和处理器 701 集成到一个芯片子系统。

存储器 702 上存储有操作系统 704、虚拟化软件程序 703 以及其他程序模块。操作系统 704 可以为计算节点 700 自身的操作系统，例如图 1 所示的宿主机操作系统 111。虚拟化软件程序 705 被处理器 701 读取和运行之后实现计算节点 700 的虚拟化，以及实现本申请各个实施例提供的支持异构指令集架构的虚拟机对异常或中断的处理方法。虚拟化软件程序 705 可实现图 1 中的虚拟机监视器 110、图 2 中的虚拟化平台、图 3 中的虚拟机监视器 303 或图 4 中的 ARM Hypervisor 402 的部分或全部功能。

通信接口 703 使用例如收发器一类的收发装置，来实现执行计算节点 700 与其他设备或通信网络之间的通信。

图 10 为前述各个实施例中提供的多核处理器的结构示意图，该多核处理器 10 可以位于任意一个电子设备中，如电脑、计算机、手机、平板等各类设备中。该多核处理器 10 具体可以是芯片或芯片组或搭载有芯片或者芯片组的电路板。该芯片或芯片组或搭载有芯片或芯片组的电路板可在必要的软件驱动下工作。

多核处理器 80 可包括主处理核 801、以及耦合于所述主处理核 801 的一个或多个从处理核 802。主处理核相当于前述的主架构处理逻辑，从处理核相当于前述的从架构处理逻辑。可有 N 个从处理核 802，包括从处理核 1（Core1）、从处理核 2（Core2）、从处理核 3（Core3）、从处理核 4（Core4）、……从处理核（N-1）（Core（N-1））和从处理核 N（CoreN）。该 N 个从处理核 802 均包含配置接口，分别为配置接口 1、配置接口 2、配置接口 3、……配置接口 N-1、配置接口 N。主处理核 801 可通过上述配置接口对相应的从处理核 802 进行相关的配置和控制。可选的，主处理核 801 和所述一个或多个从处理核 802 可位于一个或多个 IC 中，例如，主处理核 801 和所述一个或多个从处理核 802 可位于一个集成电路(Integrated Circuit，IC)中，或者，主处理核 801 位于一个 IC 中，所述一个或多个从处理核 802 中的部分或全部位于另一个 IC 中，本发明实施例对此不作具体限定。可以理解的是，主处理核 801 与 N 个从处理核 802 之间可通过总线或其他方式耦合通信，图 10 中所示意的连接关系并不对其之间的耦合关系构成限制。

主处理核 801 与 N 个从处理核 802 支持不同的指令集架构，主处理核 801 支持第一指令集，从处理核 802 支持第二指令集。可选的，主处理核 801 和 N 个从处理核 102 中的任意一个从处理核 802 之间为异构，即主处理核 801 支持的第一指令集和任意一个从处理核 802 支持的第二指令集不同。而 N 个从处理核 802 之间可以是同构也可以是异构，还可以是部分同构、部分异构，也即是 N 个从处理核 802 分别支持的指令集可以相同也可以不同，还可以部分相同、部分不同，本发明实施例对此不作具体限定。例如，在一种应用场景中，主处理核 801 为通用处理器内核，N 个从处理核 802 分别为多个特定功能的处理内核，比如，主处理核为通用 CPU，从处理核 802 分别为 FPGA、DSP 等。即每个从处理核 802 都具有各自独特的结构，因而每个从处理核都具有自己特有的指令集，而特定的指令集决定了每个从处理核的特定应用，从而使得每个从处理核都有自己擅长处理的一类程序。因此，主处理核 801 可将不同类型计算任务分配到不同类型的从处理器核 802 上并行处理，也因此在同一个处理器中同时实现了不同特定应用的功能，从而为不同需求的应用提供更加灵活、高效的处理机制。

可选的，所述第一指令集是 ARM 指令集，所述第二指令集是 RISC-V 指令集。例如，主处理核 801 和 N 个从处理核 802 均为 CPU 内核，但支持不同指令集，例如，主处理核 101 支持 ARM 指令集，从处理核 102 支持 RISC-V 指令集。因此，主处理核 801 可用于安装运行基于 ARM 指令集的虚拟机监视器和虚拟机；而从处理核 802 则可在主处理核 101 的控制下运行基于 RISC-V 指令集的应用、比如基于 RISC-V 指令集的虚拟机，从而在同一个处理器即多核处理器 80 中同时支持异构指令集的虚拟机。

本申请的说明书和权利要求书及所述附图中的术语“第一”、“第二”、“第三”和“第四”等是用于区别不同对象，而不是用于描述特定顺序。此外，术语“包括”和“具有”以及它们任何变形，意图在于覆盖不排他的包含。例如包含了一系列步骤或单元的过程、方法、系统、产品或设备没有限定于已列出的步骤或单元，而是可选地还包括没有列出的步骤或单元，或可选地还包括对于这些过程、方法、产品或设备固有的其它步骤或单元。在本文中提及“实施例”意味着，结合实施例描述的特定特征、结构或特性可以包含在本申请的至少一个实施例中。在说明书中的各个位置出现该短语并不一定均是指相同的实施例，也不是与其它实施例互斥的独立的或备选的实施例。本领域技术人员显式地和隐式地理解的是，本文所描述的实施例可以与其它实施例相结合。

在本说明书中使用的术语“部件”、“模块”、“系统”等用于表示计算机相关的实体、硬件、固件、硬件和软件的组合、软件、或执行中的软件。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、执行线程、程序和/或计算机。通过图示，在计算设备上运行的应用和计算设备都可以是部件。一个或多个部件可驻留在进程和/或执行线程中，部件可位于一个计算机上和/或分布在 2 个或更多个计算机之间。此外，这些部件可从在上面存储有各种数据结构的各种计算机可读介质执行。部件可例如根据具有一个或多个数据分组（例如来自与本地系统、分布式系统和/或网络间的另一部件交互的二个部件的数据，例如通过信号与其它系统交互的互联网）的信号通过本地和/或远程进程来通信。

权 利 要 求 书

1、一种异常的处理方法，其特征在于，所述方法应用于物理主机，所述物理主机的处理器包括支持主指令集架构的主处理核和支持从指令集架构的从处理核，所述物理主机上运行有支持所述主指令集架构的虚拟机监视器，以及支持所述主指令集架构的主架构虚拟机和支持所述从指令集架构的从架构虚拟机，所述方法包括：

所述从架构虚拟机触发异常时，所述虚拟机监视器获得所述异常的状态信息；所述异常的状态信息包括所述异常的第一编码；所述异常的第一编码表示在所述从指令集架构下所述异常的类型；

所述虚拟机监视器从异常映射关系中获取所述异常的第二编码，所述异常的第二编码表示在所述主指令集架构下所述异常的类型；所述异常映射关系包括多类异常中每类异常的第一编码与第二编码的对应关系；

所述虚拟机监视器根据所述异常的第二编码，识别出所述异常的类型并处理所述异常。

2、根据权利要求 1 所述的方法，其特征在于，所述物理主机还包括从架构寄存器，所述虚拟机监视器获得所述异常的状态信息，包括：

所述虚拟机监视器从共享内存中获取所述异常的状态信息；所述共享内存被所述从处理核和所述主处理核共享；所述异常的状态信息为所述从处理核从所述从架构寄存器中复制到所述共享内存中的；所述从架构寄存器为符合所述从指令集架构规范的、用于存储所述异常的状态信息的寄存器。

3、根据权利要求 1 所述的方法，其特征在于，所述物理主机还包括共享寄存器，所述虚拟机监视器获得所述异常的状态信息，包括：

所述虚拟机监视器从所述共享寄存器中获取所述异常的状态信息，所述共享寄存器被所述从处理核和所述主处理核共享；所述共享寄存器存储有所述异常的状态信息。

4、根据权利要求 1 或 2 所述的方法，其特征在于，所述物理主机还包括从架构寄存器和主架构寄存器，所述异常的状态信息还包括触发所述异常的指令或触发所述异常的地址中的至少一个，所述异常映射关系还包括所述从架构寄存器与所述主架构寄存器的对应关系，所述主架构寄存器为符合所述主指令集架构规范的寄存器，所述从架构寄存器为符合所述从指令集架构规范的、用于存储所述异常的状态信息的寄存器；

所述虚拟机监视器根据所述异常的第二编码，识别出所述异常的类型，并处理所述异常之前，所述方法还包括：

所述虚拟机监视器根据所述异常映射关系查找与所述从架构寄存器对应的所述主架构寄存器；

所述虚拟机监视器将所述异常的第二编码写入所述主架构寄存器，以及将所述触发异

常的指令和所述触发异常的地址中的至少一个写入所述主架构寄存器；

对应的，所述虚拟机监视器根据所述异常的第二编码，识别出所述异常的类型，并处理所述异常，包括：

所述虚拟机监视器从所述主架构寄存器中读取所述异常的第二编码，以及所述触发异常的指令和所述触发异常的地址中的至少一个；

所述虚拟机监视器根据所述异常的第二编码，以及所述触发异常的指令和所述触发异常的地址中的至少一个处理所述异常。

5、根据权利要求 1-4 任一项所述的方法，所述物理主机中包括支持硬件辅助虚拟化的硬件设备，且所述硬件设备支持所述主指令集架构，所述虚拟机监视器处理所述异常，包括：

所述虚拟机监视器通过所述支持硬件辅助虚拟化的硬件设备处理所述异常。

6、一种中断的处理方法，其特征在于，所述方法应用于物理主机，所述物理主机的处理器包括支持主指令集架构的主处理核和支持从指令集架构的从处理核，所述物理主机上运行有支持所述主指令集架构的虚拟机监视器，以及支持所述主指令集架构的主架构虚拟机和支持所述从指令集架构的从架构虚拟机，所述方法包括：

所述从架构虚拟机触发中断时，所述虚拟机监视器获得所述中断的状态信息，所述状态信息包括所述中断的第一编码；所述中断的第一编码表示在所述从指令集架构下所述中断的类型；

所述虚拟机监视器从中断映射关系中获取与所述中断的第二编码，所述中断的第二编码表示在所述主指令集架构下所述中断的类型；所述中断映射关系包括多类中断中每类中断的第一编码与第二编码的对应关系；

所述虚拟机监视器根据所述中断的第二编码，识别出所述中断的类型并处理所述中断。

7、根据权利要求 6 所述的方法，其特征在于，所述物理主机包括从架构寄存器，所述虚拟机监视器获得所述中断的状态信息，包括：

所述虚拟机监视器从共享内存中获取所述中断的状态信息，所述共享内存被所述从处理核和所述主处理核共享，所述中断的状态信息为所述从处理核从所述从架构寄存器中复制到所述共享内存中的；所述从架构寄存器为符合所述从指令集架构规范的寄存器，用于存储所述中断的状态信息。

8、根据权利要求 6 所述的方法，其特征在于，所述物理主机还包括共享寄存器，所述虚拟机监视器获得所述中断的状态信息，包括：

所述虚拟机监视器从所述共享寄存器中获取所述中断的状态信息，所述共享寄存器被所述从处理核和所述主处理核共享；所述共享寄存器存储有所述中断的状态信息。

9、根据权利要求 6 或 7 所述的方法，其特征在于，所述物理主机包括从架构寄存器

和主架构寄存器，所述中断映射关系还包括所述从架构寄存器与所述主架构寄存器的对应关系，所述主架构寄存器为符合所述主指令集架构规范的寄存器；所述从架构寄存器为符合所述从指令集架构规范的、用于存储所述中断的状态信息的寄存器；

所述虚拟机监视器根据所述中断的第二编码，识别出所述中断的类型并处理所述中断之前，所述方法还包括：

所述虚拟机监视器根据所述中断映射关系查找与所述从架构寄存器对应的所述主架构寄存器；

所述虚拟机监视器将所述中断的第二编码写入所述主架构寄存器；

对应的，所述虚拟机监视器根据所述中断的第二编码，识别出所述中断的类型并处理所述中断，包括：

所述虚拟机监视器从所述主架构寄存器中读取所述中断的第二编码，根据所述中断的第二编码，识别出所述中断的类型并处理所述中断。

10、 根据权利要求 6-9 任一项所述的方法，其特征在于，所述物理主机中包括支持硬件辅助虚拟化的硬件设备，且所述硬件设备支持所述主指令集架构，所述虚拟机监视器处理所述中断，包括：

所述虚拟机监视器通过所述支持硬件辅助虚拟化的硬件设备处理所述中断。

11、 一种计算设备，其特征在于，所述计算设备包括处理器和存储器，所述处理器包括支持主指令集架构的主处理核和支持从指令集架构的从处理核，所述存储器存储计算机指令，所述主处理核运行所述计算机指令以执行如权利要求 1-10 任一项所述的方法。

12、 一种计算机可读存储介质，其特征在于，所述计算机可读存储介质存储计算机指令，所述计算机指令被处理器调用以执行如权利要求 1-10 任一项所述的方法。

13、 一种物理主机，其特征在于，所述物理主机包括转换模块和模拟模块，

所述转换模块用于：从架构虚拟机触发异常时，获得所述异常的状态信息；所述异常的状态信息包括所述异常的第一编码；所述异常的第一编码表示在从指令集架构下所述异常的类型；所述从架构虚拟机为所述物理主机上运行的支持从指令集架构的虚拟机；从异常映射关系中获取所述异常的第二编码，所述异常的第二编码表示在主指令集架构下所述异常的类型；所述异常映射关系包括多类异常中每类异常的第一编码与第二编码的对应关系；

所述模拟模块用于：根据所述异常的第二编码，识别出所述异常的类型；处理所述异常。

14、 根据权利要求 13 所述的物理主机，其特征在于，所述物理主机还包括从架构寄存器，所述转换模块还用于从共享内存中获取所述异常的状态信息；所述共享内存被所述从处理核和所述主处理核共享；所述异常的状态信息为从所述从架构寄存器中复制

到所述共享内存中的；所述从架构寄存器为符合所述从指令集架构规范的、用于存储所述从架构虚拟机触发的异常的状态信息的寄存器。

15、 根据权利要求 13 所述的物理主机，其特征在于，所述物理主机还包括共享寄存器；所述转换模块还用于从所述共享寄存器中获取所述异常的状态信息，所述共享寄存器被所述从处理核和所述主处理核共享；所述共享寄存器存储有所述异常的状态信息。

16、 根据权利要求 13 或 14 所述的物理主机，其特征在于，所述物理主机还包括从架构寄存器和主架构寄存器，所述异常的状态信息还包括触发所述异常的指令或触发所述异常的地址中的至少一个，所述异常映射关系还包括所述从架构寄存器与所述主架构寄存器的对应关系，所述主架构寄存器为符合所述主指令集架构规范的寄存器，所述从架构寄存器为符合所述从指令集架构规范的、用于存储所述异常的状态信息的寄存器；

所述转换模块还用于：根据所述异常映射关系查找与所述从架构寄存器对应的所述主架构寄存器；将所述异常的第二编码写入所述主架构寄存器，以及将所述触发所述异常的指令和所述触发异常的地址中的至少一个写入所述主架构寄存器；

所述模拟模块用于：从所述主架构寄存器中读取所述异常的第二编码，以及所述触发异常的指令和所述触发异常的地址中的至少一个；根据所述异常的第二编码，以及所述触发异常的指令和所述触发异常的地址中的至少一个处理所述异常。

17、 根据权利要求 13-16 任一项所述的物理主机，所述物理主机中包括支持硬件辅助虚拟化的硬件设备，且所述硬件设备支持所述主指令集架构；

所述模拟模块用于：通过所述支持硬件辅助虚拟化的硬件设备处理所述异常。

18、 一种物理主机，其特征在于，所述物理主机包括转换模块和模拟模块，

所述转换模块用于：从架构虚拟机触发中断时，获得所述中断的状态信息，所述状态信息包括所述中断的第一编码；所述中断的第一编码表示在从指令集架构下所述中断的类型；所述从架构虚拟机为所述物理主机上运行的，支持所述从指令集架构的虚拟机；从中断映射关系中获取与所述中断的第二编码，所述中断的第二编码表示在所述主指令集架构下所述中断的类型；所述中断映射关系包括多类中断中每类中断的第一编码与第二编码的对应关系；

所述模拟模块用于：根据所述中断的第二编码，识别出所述中断的类型；处理所述中断。

19、 根据权利要求 18 所述的物理主机，其特征在于，所述物理主机还包括从架构寄存器，

所述转换模块用于：从共享内存中获取所述中断的状态信息，所述共享内存被所述从处理核和所述主处理核共享，所述中断的状态信息为从所述从架构寄存器中复制到所述共享内存中的；所述从架构寄存器为符合所述从指令集架构规范的寄存器，用于存储所述从架构虚拟机触发的中断的状态信息。

20、 根据权利要求 18 所述的物理主机，其特征在于，所述物理主机还包括共享寄存器，

所述转换模块还用于：从所述共享寄存器中获取所述中断的状态信息，所述共享寄存器被所述从处理核和所述主处理核共享；所述共享寄存器存储有所述中断的状态信息。

21、 根据权利要求 18 或 19 所述的物理主机，其特征在于，所述物理主机还包括从架构寄存器和主架构寄存器，所述中断映射关系还包括所述从架构寄存器与所述主架构寄存器的对应关系，所述主架构寄存器为符合所述主指令集架构规范的寄存器；所述从架构寄存器为符合所述从指令集架构规范的、用于存储所述中断的状态信息的寄存器；

所述转换模块还用于：根据所述中断映射关系查找与所述从架构寄存器对应的所述主架构寄存器；将所述中断的第二编码写入所述主架构寄存器；

所述模拟模块用于：从所述主架构寄存器中读取所述中断的第二编码；根据所述中断的第二编码处理所述中断。

22、 根据权利要求 18-21 任一项所述的物理主机，所述物理主机中包括支持硬件辅助虚拟化的硬件设备，且所述硬件设备支持所述主指令集架构；所述处理单元用于：通过所述支持硬件辅助虚拟化的硬件设备处理所述中断。

23、 一种芯片，其特征在于，所述芯片上包括支持主指令集架构的主处理核和支持从指令集架构的从处理核，所述主处理核被配置为执行如权利要求 1-10 任一项所述的方法。

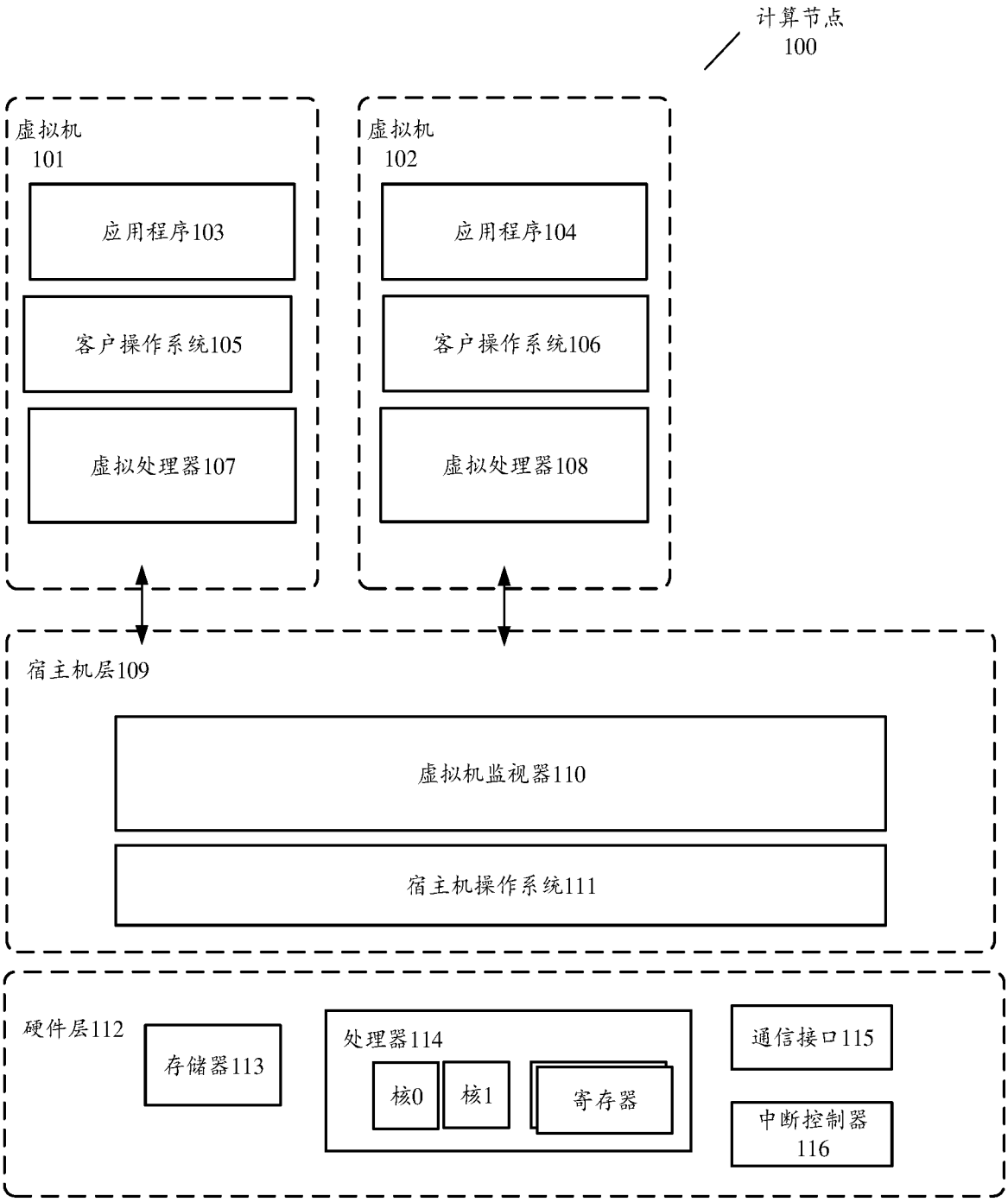
计算节点
100
虚拟机
101
应用程序103
客户操作系统105
虚拟处理器107
虚拟机
102
应用程序104
客户操作系统106
虚拟处理器108
宿主机层109
虚拟机监视器110
宿主机操作系统111
硬件层112
存储器113
处理器114
核0
核1
寄存器
通信接口115
中断控制器
116

图 1

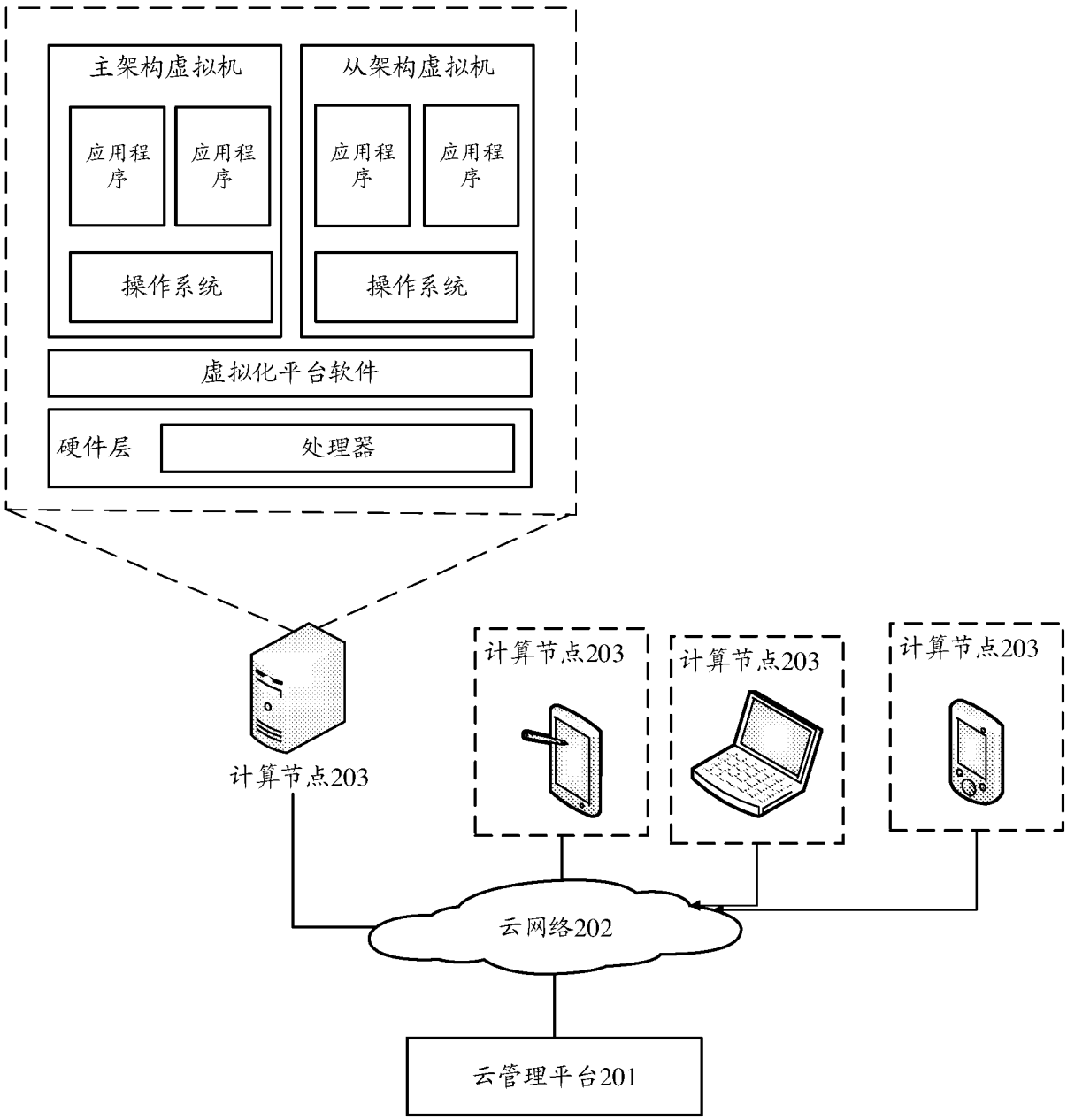
主架构虚拟机
从架构虚拟机
应用程序
应用程序
应用程序
应用程序
操作系统
操作系统
虚拟化平台软件
硬件层
处理器
计算节点203
计算节点203
计算节点203
计算节点203
云网络202
云管理平台201

图 2

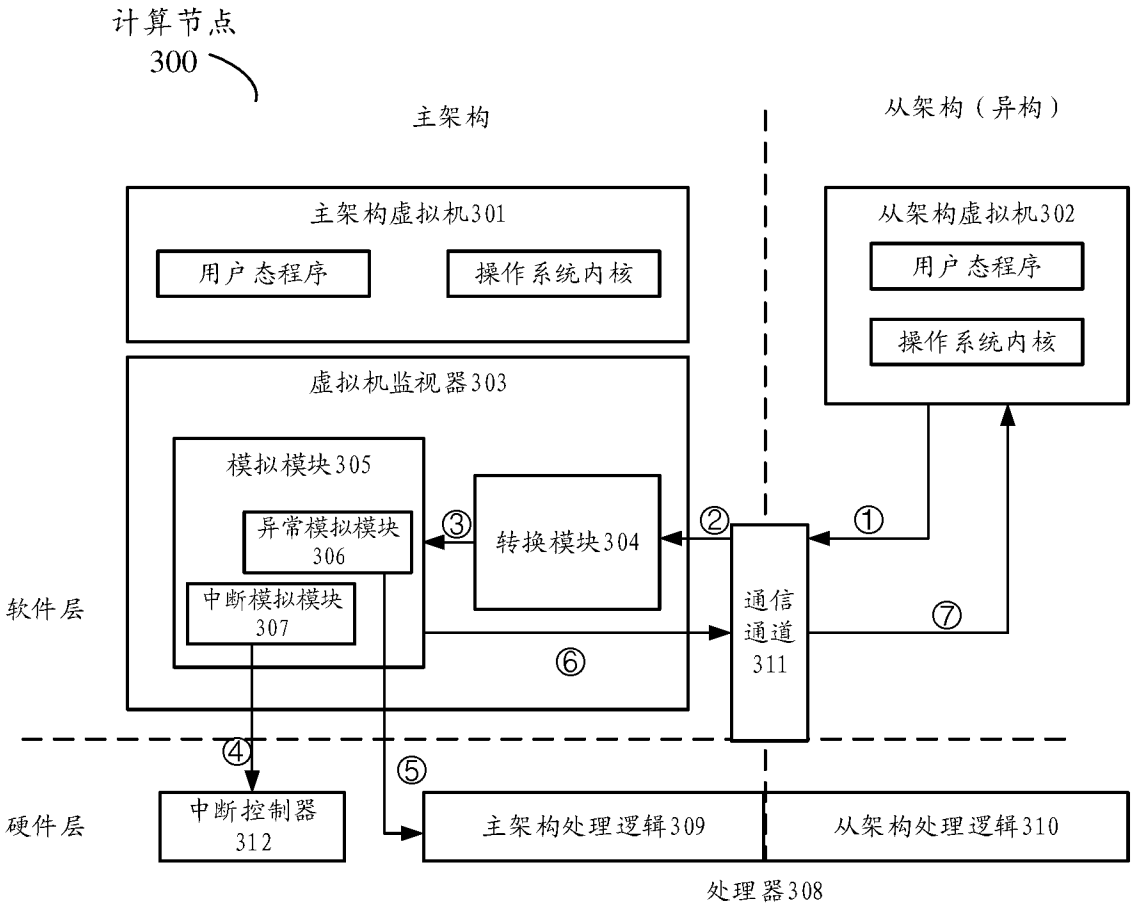
计算节点
300
主架构
从架构（异构）
主架构虚拟机301
用户态程序
操作系统内核
从架构虚拟机302
用户态程序
操作系统内核
虚拟机监视器303
模拟模块305
异常模拟模块306
中断模拟模块307
转换模块304
通信通道311
软件层
硬件层
中断控制器312
主架构处理逻辑309
从架构处理逻辑310
处理器308
①
②
③
④
⑤
⑥
⑦

图 3

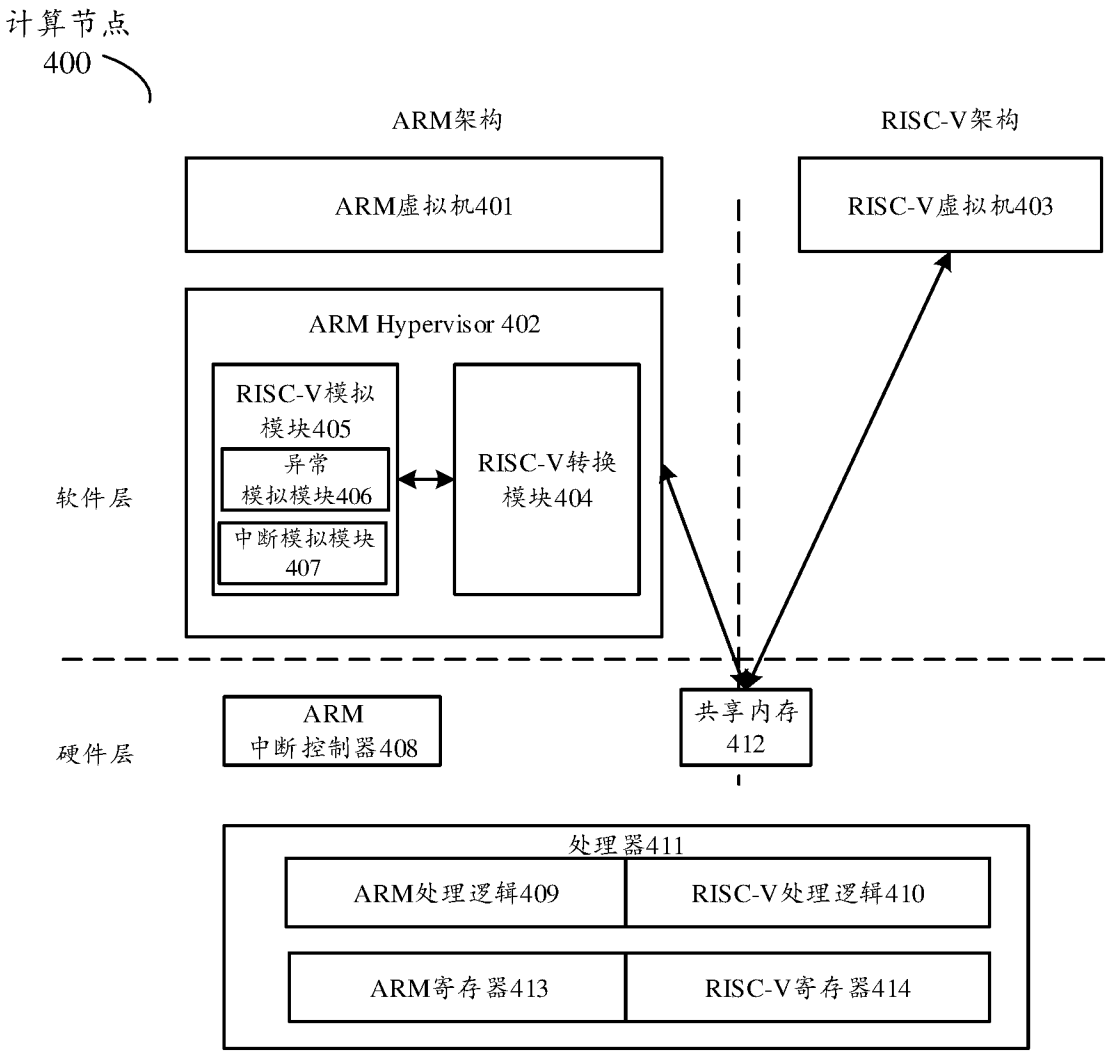
计算节点
400
ARM架构
RISC-V架构
ARM虚拟机401
RISC-V虚拟机403
ARM Hypervisor 402
RISC-V模拟模块405
异常模拟模块406
中断模拟模块407
RISC-V转换模块404
软件层
硬件层
ARM中断控制器408
共享内存412
处理器411
ARM处理逻辑409
RISC-V处理逻辑410
ARM寄存器413
RISC-V寄存器414

图 4

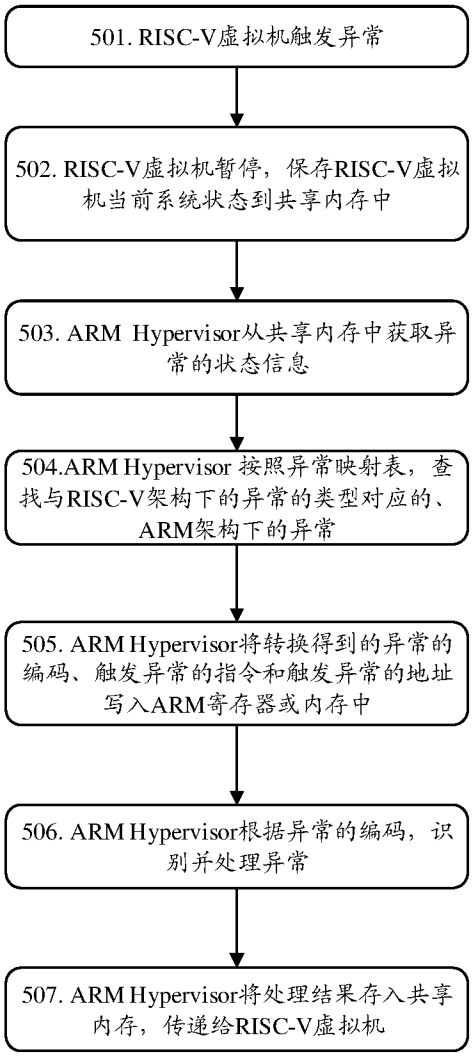
501. RISC-V虚拟机触发异常
502. RISC-V虚拟机暂停，保存RISC-V虚拟机当前系统状态到共享内存中
503. ARM Hypervisor从共享内存中获取异常的状态信息
504.ARM Hypervisor 按照异常映射表，查找与RISC-V架构下的异常的类型对应的、ARM架构下的异常
505. ARM Hypervisor将转换得到的异常的编码、触发异常的指令和触发异常的地址写入ARM寄存器或内存中
506. ARM Hypervisor根据异常的编码，识别并处理异常
507. ARM Hypervisor将处理结果存入共享内存，传递给RISC-V虚拟机

图 5

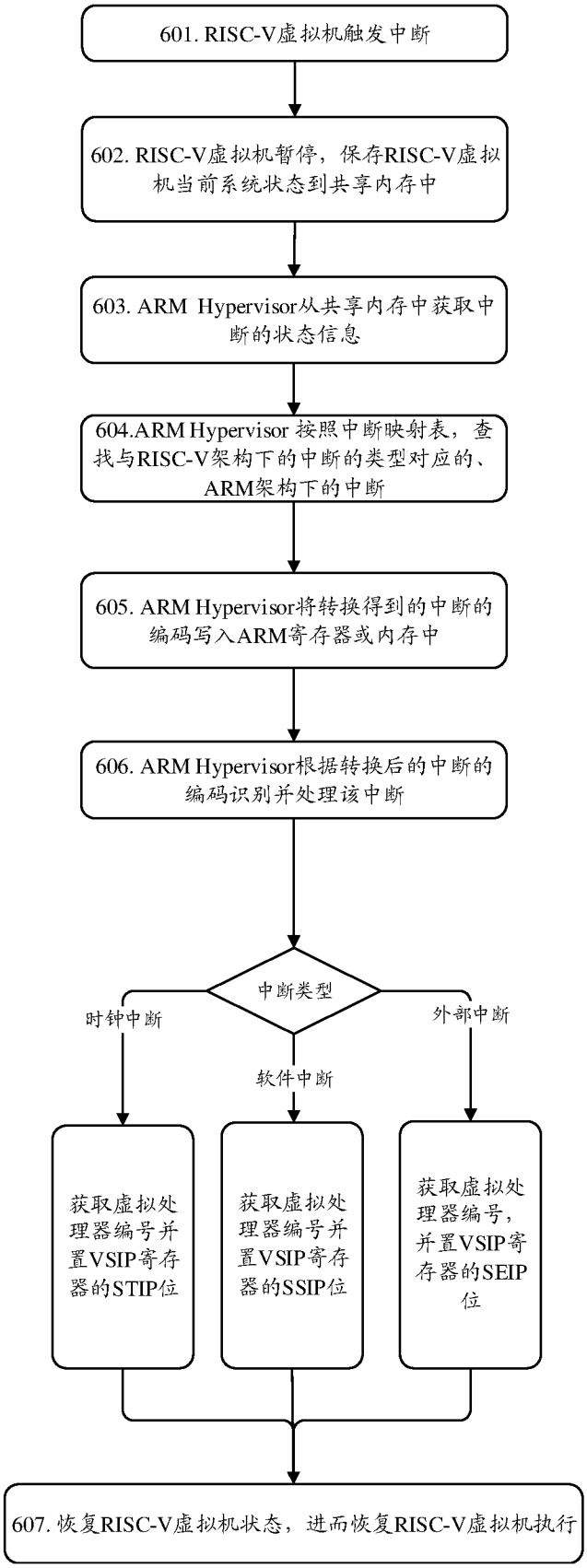
601. RISC-V虚拟机触发中断
602. RISC-V虚拟机暂停，保存RISC-V虚拟机当前系统状态到共享内存中
603. ARM Hypervisor从共享内存中获取中断的状态信息
604.ARM Hypervisor 按照中断映射表，查找与RISC-V架构下的中断的类型对应的、ARM架构下的中断
605. ARM Hypervisor将转换得到的中断的编码写入ARM寄存器或内存中
606. ARM Hypervisor根据转换后的中断的编码识别并处理该中断
中断类型
时钟中断
软件中断
外部中断
获取虚拟处理器编号并置VSIP寄存器的STIP位
获取虚拟处理器编号并置VSIP寄存器的SSIP位
获取虚拟处理器编号，并置VSIP寄存器的SEIP位
607. 恢复RISC-V虚拟机状态，进而恢复RISC-V虚拟机执行

图 6


Interrupt
Exception Code
1
0

图 7

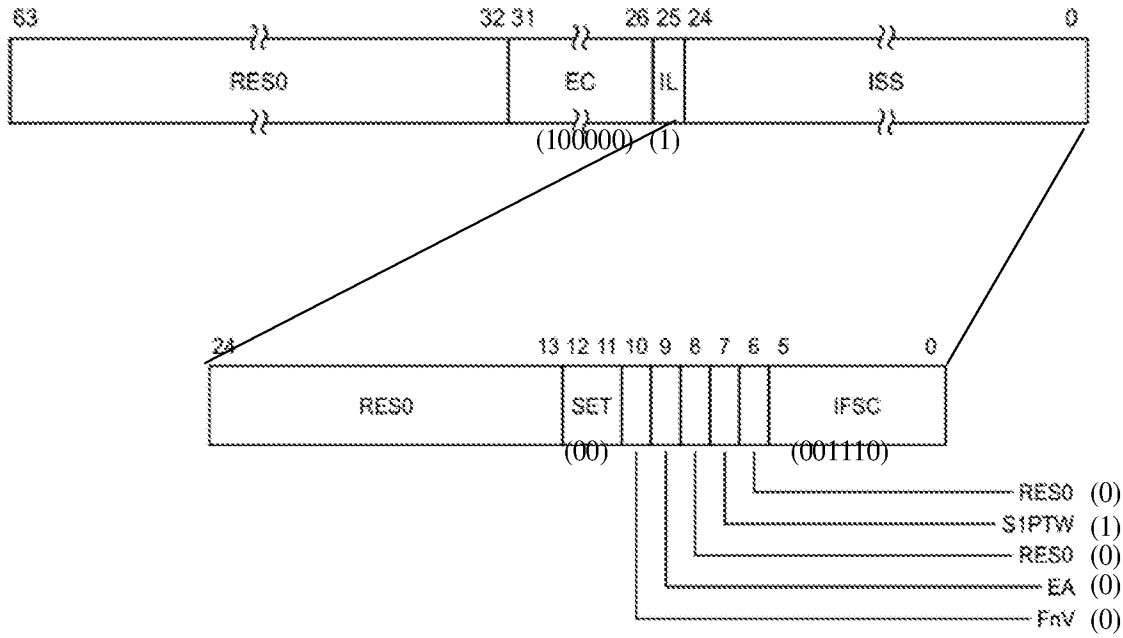
63
32 31
26 25 24
0
RES0
EC
IL
ISS
(100000)
(1)
24
13 12 11 10 9 8 7 6 5
0
RES0
SET
IFSC
(00)
(001110)
RES0 (0)
S1PTW (1)
RES0 (0)
EA (0)
FnV (0)

图 8

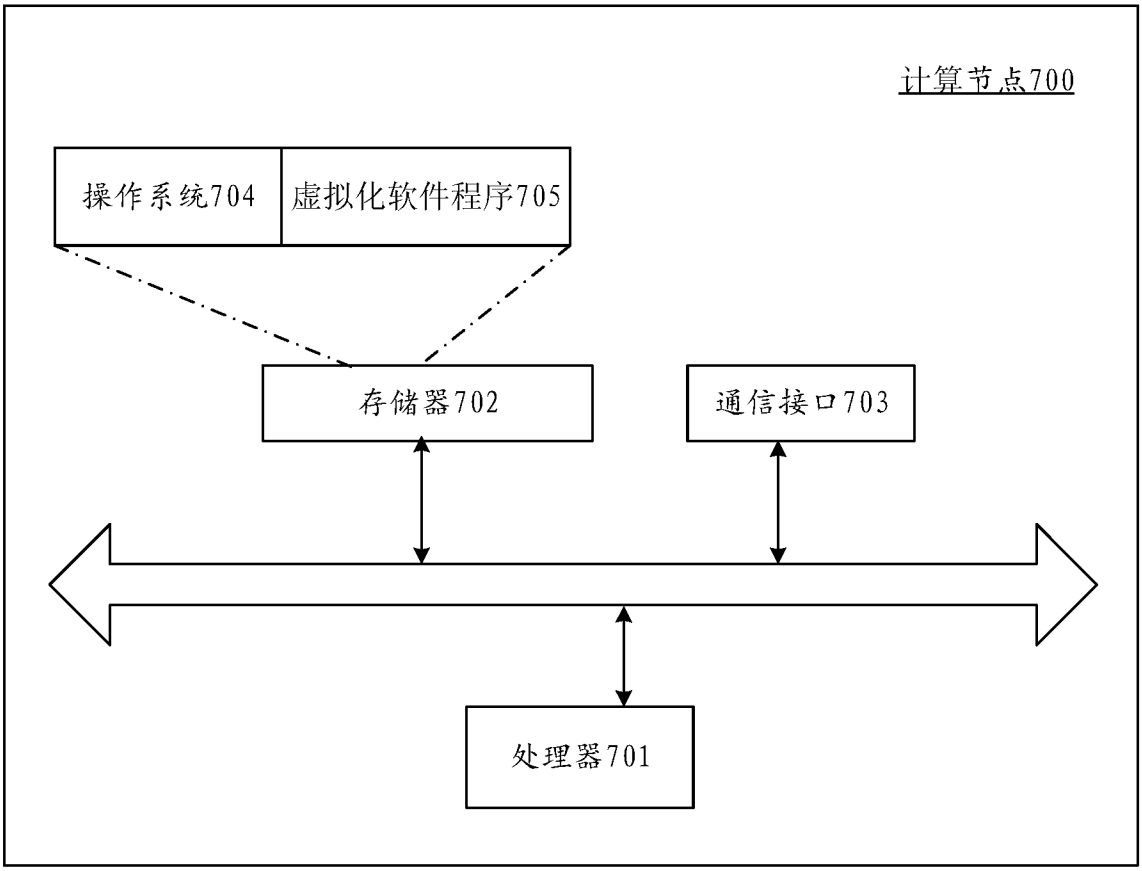

图 9

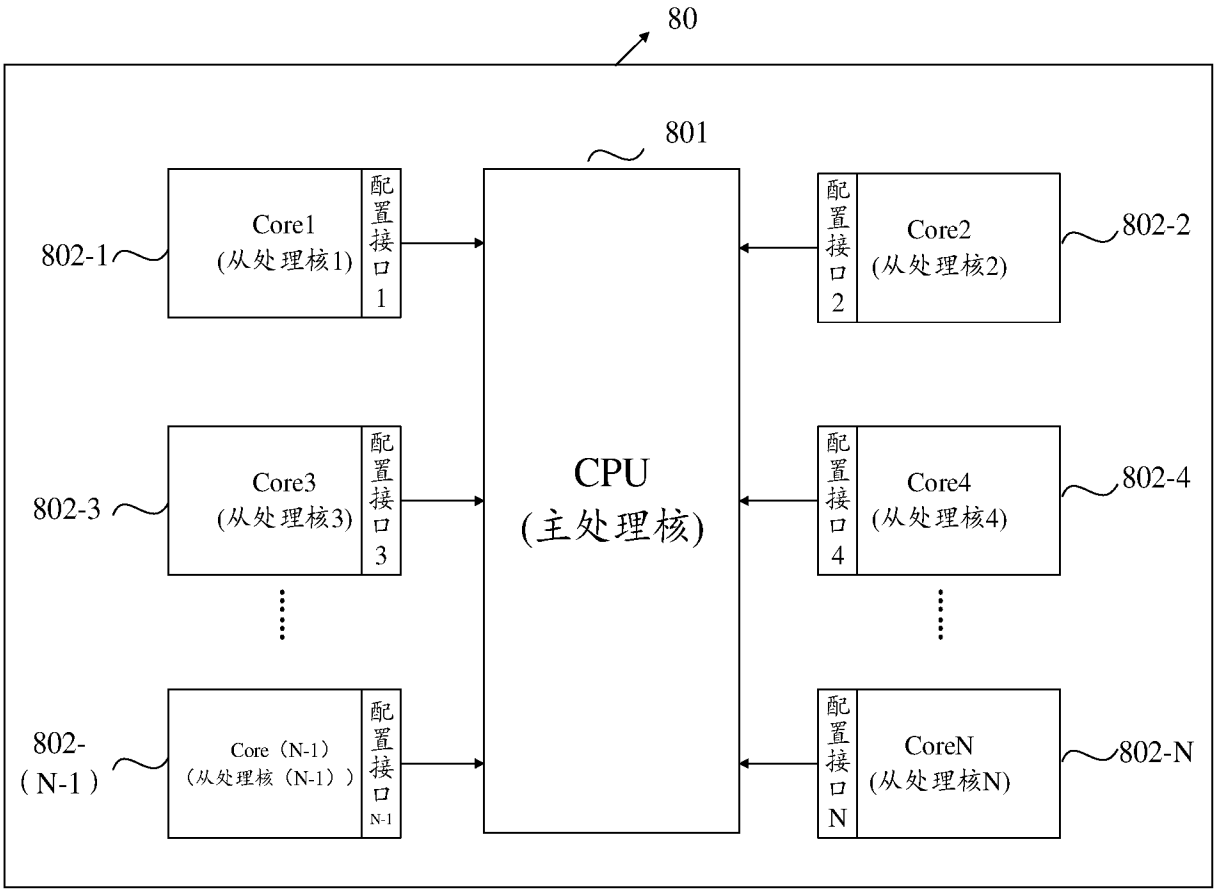
80
801
802-1
Core1
(从处理核1)
配置接口1
802-2
Core2
(从处理核2)
配置接口2
802-3
Core3
(从处理核3)
配置接口3
CPU
(主处理核)
802-4
Core4
(从处理核4)
配置接口4
802-(N-1)
Core（N-1）
（从处理核（N-1））
配置接口N-1
802-N
CoreN
(从处理核N)
配置接口N

图 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2021/096075

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/455(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNABS; TWABS; CNTXT; VEN; USTXT; CNKI: 异构, 不同, 相异, 指令集, 指令组, 异常, 故障, 中断, 编号, 编码, 转换, 翻译, 解释, 映射, 虚拟机, 监视器, 管理器, hetero, different, instruction set, ISA, exception, failure, error, fault, interrupt, number, coding, translat+, convert+, conversion, map+, VMM, virtual machine, monitor, hypervisor

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	CN 101663644 A (INTERNATIONAL BUSINESS MACHINES CORPORATION) 03 March 2010 (2010-03-03) description page 2 paragraph 2 to page 18 paragraph 4, figures 1, 7	1-23
Y	CN 103038749 A (NEODANA, INC.) 10 April 2013 (2013-04-10) claims 1-10, description paragraph 88, figure 6	1-23
A	CN 101454753 A (INTEL CORPORATION) 10 June 2009 (2009-06-10) entire document	1-23
A	CN 102918516 A (INTERNATIONAL BUSINESS MACHINES CORPORATION) 06 February 2013 (2013-02-06) entire document	1-23
A	US 6075938 A (THE BOARD OF TRUSTEES OF THE LELAND STANFORD JUNIOR UNIVERSITY) 13 June 2000 (2000-06-13) entire document	1-23
A	US 5854913 A (IBM) 29 December 1998 (1998-12-29) entire document	1-23

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 July 2021	**28 July 2021**
Name and mailing address of the ISA/CN **China National Intellectual Property Administration (ISA/CN)** **No. 6, Xitucheng Road, Jimenqiao, Haidian District, Beijing 100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2021/096075

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	101663644	A	03 March 2010	TW	I509516	B	21 November 2015
				EP	2165258	A1	24 March 2010
				JP	5077605	B2	21 November 2012
				GB	0707528	D0	30 May 2007
				EP	2165258	B1	16 March 2016
				TW	200907809	A	16 February 2009
				GB	2448523	A	22 October 2008
				CN	101663644	B	20 March 2013
				GB	2448523	B	17 June 2009
				US	2008263342	A1	23 October 2008
				JP	2010525440	A	22 July 2010
				WO	2008129315	A1	30 October 2008
CN	103038749	A	10 April 2013	US	2018246760	A1	30 August 2018
				US	2013145375	A1	06 June 2013
				US	10579426	B2	03 March 2020
				TW	201214139	A	01 April 2012
				TW	I480738	B	11 April 2015
				US	9959139	B2	01 May 2018
				US	9477524	B2	25 October 2016
				CN	103038749	B	15 September 2017
				US	2016378538	A1	29 December 2016
				CN	107608755	A	19 January 2018
				WO	2012003486	A1	05 January 2012
CN	101454753	A	10 June 2009	US	2008005546	A1	03 January 2008
				WO	2008002978	A1	03 January 2008
				CN	102981800	A	20 March 2013
				CN	102981800	B	05 August 2015
				US	7487341	B2	03 February 2009
				DE	112007001466	T5	07 May 2009
				CN	101454753	B	28 November 2012
CN	102918516	A	06 February 2013	CN	102918516	B	12 August 2015
				US	8504754	B2	06 August 2013
				JP	5649200	B2	07 January 2015
				JP	2013536485	A	19 September 2013
				WO	2011160705	A1	29 December 2011
				EP	2585932	A1	01 May 2013
				EP	2585932	B1	10 April 2019
				US	2011320662	A1	29 December 2011
US	6075938	A	13 June 2000		None		
US	5854913	A	29 December 1998	KR	970002607	A	28 January 1997
				KR	100195666	B1	15 June 1999
				EP	0747808	A3	15 January 1997
				EP	0747808	A2	11 December 1996
				JP	H08339325	A	24 December 1996
				JP	3451595	B2	29 September 2003

国际检索报告

国际申请号
PCT/CN2021/096075

A. 主题的分类

G06F 9/455(2006.01)i

按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

G06F

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称，和使用的检索词（如使用））

CNABS;TWABS;CNTXT;VEN;USTXT;CNKI: 异构，不同，相异，指令集，指令组，异常，故障，中断，编号，编码，转换，翻译，解释，映射，虚拟机，监视器，管理器，hetero, different, instruction set, ISA, exception, failure, error, fault, interrupt, number, coding, translat+, convert+, conversion, map+, VMM, virtual machine, monitor, hypervisor

C. 相关文件

类 型*	引用文件，必要时，指明相关段落	相关的权利要求
Y	CN 101663644 A (国际商业机器公司) 2010年 3月 3日 (2010-03-03) 说明书第2页第2段至第18页第4段，图1、7	1-23
Y	CN 103038749 A (纽戴纳公司) 2013年 4月 10日 (2013-04-10) 权利要求1-10，说明书第88段，图6	1-23
A	CN 101454753 A (英特尔公司) 2009年 6月 10日 (2009-06-10) 全文	1-23
A	CN 102918516 A (国际商业机器公司) 2013年 2月 6日 (2013-02-06) 全文	1-23
A	US 6075938 A (UNIV LELAND STANFORD JUNIOR) 2000年 6月 13日 (2000-06-13) 全文	1-23
A	US 5854913 A (IBM) 1998年 12月 29日 (1998-12-29) 全文	1-23

☐ 其余文件在C栏的续页中列出。

☑ 见同族专利附件。

* 引用文件的具体类型：

“A” 认为不特别相关的表示了现有技术一般状态的文件

“E” 在国际申请日的当天或之后公布的在先申请或专利

“L” 可能对优先权要求构成怀疑的文件，或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件（如具体说明的）

“O” 涉及口头公开、使用、展览或其他方式公开的文件

“P” 公布日先于国际申请日但迟于所要求的优先权日的文件

“T” 在申请日或优先权日之后公布，与申请不相抵触，但为了理解发明之理论或原理的在后文件

“X” 特别相关的文件，单独考虑该文件，认定要求保护的发明不是新颖的或不具有创造性

“Y” 特别相关的文件，当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时，要求保护的发明不具有创造性

“&” 同族专利的文件

国际检索实际完成的日期	国际检索报告邮寄日期
2021年 7月 20日	2021年 7月 28日
ISA/CN的名称和邮寄地址 中国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088 传真号 (86-10)62019451	受权官员 张静 电话号码 (86-27) 59371867

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2021/096075

检索报告引用的专利文件			公布日 (年/月/日)	同族专利			公布日 (年/月/日)
CN	101663644	A	2010年 3月 3日	TW	I509516	B	2015年 11月 21日
				EP	2165258	A1	2010年 3月 24日
				JP	5077605	B2	2012年 11月 21日
				GB	0707528	D0	2007年 5月 30日
				EP	2165258	B1	2016年 3月 16日
				TW	200907809	A	2009年 2月 16日
				GB	2448523	A	2008年 10月 22日
				CN	101663644	B	2013年 3月 20日
				GB	2448523	B	2009年 6月 17日
				US	2008263342	A1	2008年 10月 23日
				JP	2010525440	A	2010年 7月 22日
				WO	2008129315	A1	2008年 10月 30日
CN	103038749	A	2013年 4月 10日	US	2018246760	A1	2018年 8月 30日
				US	2013145375	A1	2013年 6月 6日
				US	10579426	B2	2020年 3月 3日
				TW	201214139	A	2012年 4月 1日
				TW	I480738	B	2015年 4月 11日
				US	9959139	B2	2018年 5月 1日
				US	9477524	B2	2016年 10月 25日
				CN	103038749	B	2017年 9月 15日
				US	2016378538	A1	2016年 12月 29日
				CN	107608755	A	2018年 1月 19日
				WO	2012003486	A1	2012年 1月 5日
CN	101454753	A	2009年 6月 10日	US	2008005546	A1	2008年 1月 3日
				WO	2008002978	A1	2008年 1月 3日
				CN	102981800	A	2013年 3月 20日
				CN	102981800	B	2015年 8月 5日
				US	7487341	B2	2009年 2月 3日
				DE	112007001466	T5	2009年 5月 7日
				CN	101454753	B	2012年 11月 28日
CN	102918516	A	2013年 2月 6日	CN	102918516	B	2015年 8月 12日
				US	8504754	B2	2013年 8月 6日
				JP	5649200	B2	2015年 1月 7日
				JP	2013536485	A	2013年 9月 19日
				WO	2011160705	A1	2011年 12月 29日
				EP	2585932	A1	2013年 5月 1日
				EP	2585932	B1	2019年 4月 10日
				US	2011320662	A1	2011年 12月 29日
US	6075938	A	2000年 6月 13日		无		
US	5854913	A	1998年 12月 29日	KR	970002607	A	1997年 1月 28日
				KR	100195666	B1	1999年 6月 15日
				EP	0747808	A3	1997年 1月 15日
				EP	0747808	A2	1996年 12月 11日
				JP	H08339325	A	1996年 12月 24日
				JP	3451595	B2	2003年 9月 29日