[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/38 (2006.01)

[12] 发明专利说明书

专利号 ZL 200310101080.1

[45] 授权公告日 2006 年 1 月 18 日

[11] 授权公告号 CN 1237443C

[22] 申请日 2003.10.15
[21] 申请号 200310101080.1
[30] 优先权
[32] 2002.10.24 [33] US [31] 10/280，677
[71] 专利权人 国际商业机器公司
地址 美国纽约
[72] 发明人 迈克尔·K.·克施温德
凯瑟琳·M.·奥布莱恩
约翰·K.·奥布莱恩
瓦伦蒂纳·萨拉普罗
审查员 胡徐兵

[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 付建军

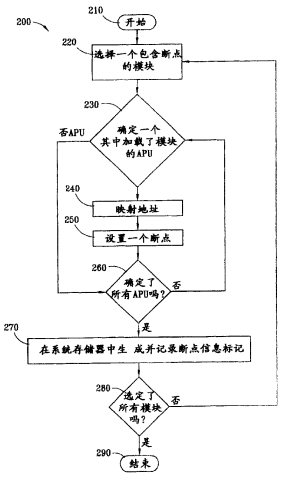
权利要求书 4 页 说明书 9 页 附图 4 页

[54] 发明名称

在调试异构体系结构中的集成可执行程序时设置断点的方法和设备

[57] 摘要

本发明用来在并行处理系统中插入和删除断点。 断点被插入到加载到附加处理器单元的执行环境中的一个模块中。 该断点可以直接插入。 此外，该模块的未加载映像也可以有一个与其关联的断点。 该断点可以直接插入到模块映像中，也可以生成断点请求，然后当该模块加载到附加处理器单元的执行环境时插入该断点。



1. 一种在第二执行环境中设置断点的方法，包括:

选择与断点相关的映射标记;

返回一个对应于映射标记的存储器地址;

选择与至少一个第二执行环境中的执行相关联的一个代码模块，其中所选择的代码模块包含所选择的映射标记;

确定是否将所选择的代码模块从至少一个第一执行环境加载到至少一个第二执行环境;

如果至少一个第二执行环境包含所选择的代码模块，在该第二执行环境中的返回的存储器地址设置断点;

如果第二执行环境包含所选择的代码模块，生成与设置断点相关联的断点标记; 以及

记录该断点标记，其中该断点标记包含可用来将该模块加载到至少一个第二执行环境的信息。

2. 根据权利要求1的方法，其特征在于，针对在第二执行环境中的执行选择至少一个代码模块的步骤还包括，选择针对在第二个执行环境中执行的多个代码模块。

3. 根据权利要求1的方法，其特征在于，存储器地址嵌入在第一执行环境中，并且还包括将存储器地址的内容复制到一个独立的信息存储位置的步骤。

4. 根据权利要求1的方法，其特征在于，设置断点的步骤包括插入一个陷阱函数。

5. 根据权利要求1的方法，其特征在于，设置断点的步骤还包括在驻留在第一执行环境的相应模块映像的相应存储器地址中插入一个操作代码。

6. 根据权利要求1的方法，其特征在于，所加载的代码模块被加载到第二执行环境，并且还包括确定第二执行环境中所加载的代码模块的加载存储器地址。

7. 根据权利要求5的方法，其特征在于，插入操作代码的步骤包括使用由映射地址返回的地址和所加载的代码模块的加载存储器地址二者的和。

8. 根据权利要求1的方法，还包括：如果存储器地址对应于第一执行环境，则在第二执行环境中对应的存储器地址插入断点。

9. 根据权利要求1的方法，其特征在于，记录信息包括修改共享存储器中所选择代码模块的相应的映像。

10. 根据权利要求1的方法，其特征在于，记录信息包括向断点请求表写入信息。

11. 一种在将模块从第一执行环境加载到第二执行环境时插入断点的方法，包括：

将模块加载到第二执行环境；

确定是否为所加载的模块选择了断点；

确定该断点在第二执行环境中的映射地址；以及

在所述映射地址插入断点。

12. 根据权利要求11的方法，其特征在于，确定步骤包括读取一个断点请求表。

13. 根据权利要求11的方法，还包括将对应于加载模块的断点的存储器地址的内容复制到单独的信息存储位置。

14. 权利要求11的方法，进一步包括：

选择一个要删除的断点；

确定所选择的要删除的断点是否与第一执行环境或第二执行环境中的存储器地址相关；

删除与所确定的执行环境相关的断点；

确定是否已经为与所确定的执行环境相关联的断点记录了断点请求；以及

如果已经记录了断点请求，则删除该断点请求。

15. 根据权利要求14的方法，还包括在断点插入原始信息。

16. 根据权利要求14的方法，其特征在于，删除所确定的环境

中的断点的步骤还包括，删除存储在共享存储器中的主模块中的断点。

17. 根据权利要求14的方法，其特征在于，删除所确定环境中的断点的步骤还包括，删除与一个加载模块对应的模块映像中的断点，其中对应于该加载模块的模块映像存储在共享存储器中。

18. 根据权利要求14的方法，其特征在于，所确定的执行环境是第二执行环境，并且还包括一个步骤，如果在其中嵌入了断点的模块加载到第二执行环境，则查询所确定的执行环境。

19. 根据权利要求14的方法，其特征在于，删除所确定的第二执行环境中的断点的步骤包括，确定与该断点相关联的模块是否加载到第二执行环境中。

20. 一种在并行处理环境中设置断点的方法，包括:

在代码中选择一个断点;

确定所请求的断点是否与要在第一执行环境或第二执行环境中执行的代码地址相关;

将断点映射到其对应的代码存储器地址;

如果断点与面向第一执行环境的代码相关，在映射返回的地址插入一个操作代码;

如果断点与加载到第二执行环境的模块相关，在第二执行环境中设置代码的断点。

21. 根据权利要求20的方法，还包括记录与所请求的断点相关联的信息，其中该信息包括将模块加载到至少一个第二执行环境时可以使用的标记。

22. 根据权利要求20的方法，其中操作代码包括一个陷阱函数。

23. 根据权利要求20的方法，其中确定步骤还包括确定所请求的断点是否映射到第一执行环境和第二执行环境。

24. 根据权利要求20的方法，还包括用操作代码重写第一执行环境中的信息的步骤。

25. 根据权利要求24的方法，还包括在重写信息的步骤之前，将要重写的信息复制到单独的信息存储位置。

26. 一个并行处理系统，包括:

一个第一执行环境;

一个第二执行环境;

一个断点请求表，其中该断点请求表位于第一执行环境;

一个断点请求标记，其中该断点请求标记在从断点请求表提取时使用，断点请求标记还可以在将一个断点插入在与断点请求标记相关的地址时使用；以及

一个所请求的断点标记与之相关联的模块，其中该模块还可以在从第一执行环境向第二执行环境传送时使用。

在调试异构体系结构中的集成可执行程序时设置断点的方法和设备

技术领域

本发明一般涉及多处理，具体而言，本发明涉及在面向并行处理器计算系统中的多个处理器的代码和数据上使用调试程序。

背景技术

在计算机技术中，并行处理非常重要。并行处理通常包括使用多个连接到同一系统的微处理器，用来并发处理一批数据。并行处理一般主要包括三种。这些并行处理系统使用共享存储器或分布式存储器或二者的组合。一般情况下，共享存储器是可以由多个处理器以单个操作，如“加载”或“读”命令来访问的存储器。分布式存储器是被本地化成单个处理器的存储器。换句话说，每个处理器都可以在单个的访问操作中访问与其本身相关联的存储器，但无法在单个操作中访问与其他处理器相关联的存储器。最后，还有一种混合或“异构”的并行处理，其中既有共享存储器又有分布存储器。

典型的这样一种混合并行处理器系统包括一个精减指令集（RISC）主处理器单元（MPU），如PowerPC™处理器，和一个专用或“附加”处理器（APU），如Synergistic™ APU (SPU)。一般而言，使用MPU来执行通用代码，其中通用代码包括复杂控制流并协调总体的混合并行处理功能。MPU能够访问所有的系统存储器。APU 通常用来执行数据流操作。也就是说，APU计算高度重复的多媒体、图形、信号或网络处理工作量，这些工作量的特点是具有高的计算与控制决策比例。在常规的混合系统中，APU无法访问系统存储器，并且其本身的存储器，即局部存储器，通常比共享存储器小。

一般情况下，使用混合系统虽然提供了高的计算性能，但对编

程模型提出了重大的挑战。这类问题与APU相关。APU无法直接寻址系统存储器。因此，任何代码要在APU上运行，必须先传送到与APU相关联的局部存储器，然后才能在APU上执行此代码。此外，APU和MPU还可以具有不同的指令集。

此外，还存在与调试要被编译和链接以在独立的执行环境中运行的有关软件调试其他问题。为帮助解决软件设计和实现过程中存在的各种问题，程序员使用调试程序。一般而言，调试程序使用的低级操作按照三种原语之一进行分类。第一种调试程序原语涉及在一个已经定义的位置停止程序。这要求调试程序（1）识别与函数名称、文件/行号或其他唯一标识源代码结构关联的地址，以及（2）设置一个断点。

第二个调试程序原语是关于将一个程序位置映射到函数名称、文件/行号或其他唯一标识源代码结构关联的地址。这要求调试程序将一个存储器地址映射到这种源结构。所映射的存储器地址通常是程序计数器PC的当前地址，程序计数器PC涉及到由调试程序再次读取PC寄存器的值。本领域技术人员应该理解，程序计数器包括当前正在执行的指令的存储器地址。

第三个调试程序原语允许读取和写入程序数据。这要求调试程序识别与数据对象或变量相关的存储器地址。一般情况下，设置断点与读访问或写访问地址存储器位置的内容结合使用。

通常，上述三个原语中的每一个包括一个映射步骤和一个操作步骤。映射步骤识别可执行对象代码和源代码之间的相关性，而操作步骤包括由调试程序执行的其他操作。为了执行映射步骤，调试程序使用最初由编译器生成并由链接器更新的映射表和调试表，对每个程序对象的位置、每个标签、文件/行号和对象地址之间的相关性、变量的布置、堆栈的布置等等进行描述。

通常，在非异构体系结构中为调试程序设置断点采用如下两种方式。第一种方式是在用陷阱指令，或者将要中断程序的正常执行并将控制传送的调试程序的其他序列，替换位于“断点”的一个选择的指令或数据。第二种方式是用断点的地址值（或地址范围）初始化断点寄存器。硬件比较程序计数器，即包含执行的指令地址的寄存器，与一个或多个断点寄存器的值。如果值匹配，则在断点寄存器中的值与程序计数器匹配时，将控制转移到调试程序。匹配可以包括各种匹配函数，如“ 等于”，“ 落入一个范围”，“ 小于”，“ 大于”或其它

布尔函数。

但是，在异构体系结构中设置断点会更加复杂。例如，单独处理器组件的指令集可以不同。这在调试中会带来问题。例如，根据是否将一个模块加载到第二执行环境，该模块的各种代码配置是不同的。此外，当在可以加载和卸载的模块中设置断点时，在这种加载和卸载活动中与所请求的断点相一致，正确地保持断点是很重要的。因此，当一个模块由另一个模块取代，并在之后重新加载时，必须注意在断点已经设置时保持所有断点。

因此，希望有一个能够在混合并行处理系统使用的调试程序，能够克服常规调试程序的局限性。

发明内容

本发明提供了一个第一执行环境，一个第二执行环境，以及一个断点请求。模块带有与之相关联的请求断点。该模块在执行从第一执行环境向第二执行环境的传送时使用。还提供了一个断点请求表。读取该断点请求表，从该断点请求表提取断点标记。断点标记在将一个断点插入到与该断点请求相关的地址时使用。

本发明提供了一种在第二执行环境中设置断点的方法，包括：选择与断点相关的映射标记；返回一个对应于映射标记的存储器地址；选择与至少一个第二执行环境中的执行相关联的一个代码模块，其中所选择的代码模块包含所选择的映射标记；确定是否将所选择的代码模块从至少一个第一执行环境加载到至少一个第二执行环境；如果至少一个第二执行环境包含所选择的代码模块，在该第二执行环境中的返回的存储器地址设置断点；如果第二执行环境包含所选择的代码模块，生成与设置断点相关联的断点标记；以及记录该断点标记，其中该断点标记包含可用来将该模块加载到至少一个第二执行环境的信息。

本发明提供了一种在将模块从第一执行环境加载到第二执行环境时插入断点的方法，包括：将模块加载到第二执行环境；确定是否为所加载的模块选择了断点；确定该断点在第二执行环境中的映射地址；以及在所述映射地址插入断点。

附图说明

为了更全面地理解本发明及其各种优点，下面将结合附图对本

发明进行详细说明，其中附图为：

图1概括地描述了一种其中使用调试程序的混合并行处理环境；

图2描述了在与APU相关的局部存储器中设置断点的方法；

图3描述了当从系统存储器向与APU相关的局部存储器加载模块时，在与APU相关的存储器中设置断点的方法；

图4描述从局部存储器和系统存储器删除断点的方法。

具体实施方式

在下文中阐述了各种特定的细节以便于全面地理解本发明。但是，本领域普通技术人员应该认识到本发明不需要这些特定细节的内容也可以实现本发明。在其他的实例中，对公知的部件用示意图或框图的形式来说明，以免不必要的细节妨碍对本发明的清楚理解。此外，对于大部分内容，关于网络通信、电磁信号技术等的细节内容均略去，因为这些内容对完整地理解本发明是不必要的，并且这些内容应该是相关领域的普通技术人员能够理解的。

还应提起注意的是，除非另外指明，这里描述的所有功能都可以即用硬件又用软件，或者用二者组合的形式实现。但在优选实施例中，除非另外指明，这些功能是由处理器根据计算机程序代码、软件和/或被编程为执行这些功能的集成电路执行的，如计算机或电子数据处理器。

参考图1，引用标记100通常指异构并行处理体系结构，提供了一种环境，用来使用存根函数传送信息。该体系结构100包括一个分布式计算环境100和一个系统存储器160，二者由一个接口150以电子方式连接。环境100包括多个APU 120，每个都带有其各自的局部存储器125。环境100还包括一个MPU130，如RISC处理器及其一级缓冲存储器135。在一个实施例中，MPU130通过一个信号路径连接到系统存储器160。在一个实施例中，APU包括一个SPU。在另一个实施例中，使用了多个MPU。

环境100还包括一个存储器流控制器（MFC）140。通常，MFC140用来实现MPU130和APU120处理器之间的数据移动和同步功能，并提供主系统存储器160和局部存储器125之间的数据传送。在图1中，MFC140通过接口150连接到系统存储器160。

通常，MFC140根据主处理器130或APU120的请求，在系统存储

器160和APU120的局部存储器125之间实现文本（即代码）和数据这些信息的移动。因为APU120不能直接访问系统存储器160，MFC140根据传送函数，如在APU120或MPU130上运行的存根函数的请求，在系统存储器160和APU120的局部存储器125之间传送信息。在一个实施例中，MFC140包括一个直接存储器访问（DMA）设备。

体系结构100是可执行程序在其中运行的一种环境，其中可执行程序带有内嵌在其中的存根函数。在存根函数中，要由APU120使用的代码和数据被封装为一个软件“对象”。通常，存根函数命令MFC140在两个单独的执行环境，如系统存储器160和局部存储器125之间传送信息。存根函数允许MPU130使代码和数据流动到APU120的局部存储器125进行处理，以便让APU120执行处理，然后让APU120将经过处理的数据流动回MPU130。由APU120执行的数据和代码处理对MPU130是不可见的，并允许MPU130并行执行其他数据处理或程序流控制任务。

一般情况下，存根函数命令MFC140将代码和数据从系统存储器160中的指定地址流动到APU120的局部存储器125。存根函数还命令MFC140命令APU120处理数据。通常，存根函数是到APU120的单个输入点。换句话说，APU120或者与其相关的局部存储器125，通常只能通过存根函数访问。在另一个实施例中，流动代码中还包含其他存根函数。当APU120完成数据时，此流动存根函数允许APU120命令MFC140将经过处理的数据发送回共享存储器160。当存根对象存储在APU120的局部存储器125中时，对应的存储器地址改变为不同的映射标记。

嵌入在系统存储器160和局部存储器125中的存根对象可以在其代码或数据中插入一个断点。通常，该断点允许编程人员在特定的代码或数据位置停止执行集成可执行程序。这可以由环境100利用一个陷阱指令，或其它将中断程序的正常执行并将控制传送到调试程序的类似序列实现。而且，环境100还可以用断点的地址值（或地址范围）来初始化一个断点寄存器。硬件比较程序计数器与一个或多个断

点寄存器的值。调试程序还包括装置，用于在控制已传送到调试程序时读取各个存储器位置的内容，并将该信息传递给编程人员。

现在看图2，图中所示为方法200，用于在可在第二执行环境，如局部存储器125中执行的模块中插入一个断点。在开始步骤210后，方法200定位一个模块，该模块包含选定的映射标记，如变量、行号和文件名等。通常，该位置通过查询一个映射标记表来执行，该映射标记表记录了在所需要的映射标记和所需要的模块之间的对应关系。映射标记的地址然后对应于至少一个断点的放置位置。

在步骤230中，方法200然后确定与所需要的断点相关的模块是否存储在至少一个第二执行环境，如局部存储器125中。如果在至少一个第二执行环境中没有加载与所需要的断点相关联的选定模块，方法200执行步骤260。但是，如果在至少一个第二执行环境中加载了一个选定模块，则方法200确定该选定模块被加载到附加APU120的哪个局部存储器125。通常，通过查询一个映射标记表来进行定位，在该映射标记表中记录了模块存储在哪个局部存储器125中的信息，如果局部存储器中有模块的话。

在步骤240中，方法200将地址从映射标记映射到所需要的断点。映射地址240使用从加载模块开始的所需要的偏移量定位加载模块中所需要的断点。在一个实施例中，映射地址240使用映射标记表进行这种确定。

在步骤250，方法200在所选定的加载模块中，在步骤240中确定的映射地址设置一个断点。设置断点可以包括插入一个操作码，如一个陷阱指令集，写入APU120的一个断点寄存器，或其他本领域技术人员知道的设置断点的方法。在一个实施例中，当使用陷阱指令时，设置断点还包括在插入断点之前，将映射地址的内容复制到存储器的一个单独的部分。这使得以后的时间，比如，删除断点时，将内容恢复到断点位置。

在步骤260，方法200确定包含选定模块的所有APU的局部存储器125已经由步骤240和250确定和处理过。如果没有，方法200重复步

骤230. 如果所有包含选定模块的APU120都经过确定和处理，则执行步骤270.

在步骤270，生成并记录与对应于局部存储器125的断点相关联的断点信息和标记。此断点信息和标记通常包括存储在共享存储器160中信息。通常，断点标记用于将来将同一模块加载到第二执行环境。

在第一执行环境中，断点信息，如映射地址240所生成的信息，包括一个陷阱指令，该陷阱指令写入系统存储器160中的未加载模块的映像，从而重写其中嵌入的一些信息。因此，当复制选定模块将其加载到第二执行环境中，将预先加载该断点。同样，在步骤250，设置断点还包括，在插入调试之前，将映射地址的内容复制到存储器的一个单独部分。

在另一个实施例中，未加载模块映像本身中的信息不重写。相反，创建一个断点请求，将该断点请求加载到断点请求总表中。使用断点请求可以归因于下列因素，如，存储在只读存储器中的选定模块映像，为了指示断点，使用断点寄存器取代将陷阱指令插入到指令流中，设法阻止包含一个或多个其它装置的专利，等等。在此第二实施例中，当模块映像从系统存储器160加载到局部存储器125时，环境100查询断点总表确定是否应在此加载模块中插入一个断点，并且，如果插入，插入在什么位置。

在步骤280，方法200确定是否所有带有请求映射标记，并且与请求断点相关的模块都已被选定，并由方法200处理。如果未选定带有请求映射标记的所有模块，并且未由方法200处理，则开始步骤220。如果选定并处理，则方法200在步骤290结束。如果带有请求映射标记的全部模块都已被选定并由方法200处理，则开始步骤220。

现在看图3. 图中所示为方法300，用于将模块加载到第二环境中时通过使用断点总表来插入断点。通常，在要加载的模块的主拷贝在系统存储器160中未曾修改时使用方法300. 在开始步骤310后，步骤320将一个模块从第一执行环境加载到APU120的局部存储器125

中。在步骤330中，方法300读取断点请求表，以查看在断点请求总表中是否有断点请求。步骤340确定是否请求了断点。如果没有请求断点，则结束步骤350。

但是，如果有断点请求，则执行步骤360。步骤360读取断点请求以确定需要哪个映射标记。然后，方法300访问映射表，其中存储了标记的偏移值。此偏移值是将在加载模块中插入断点的位置。

在步骤370，在步骤360所指示的位置，在加载模块中插入一个断点。与方法200相似，设置断点进一步包括，在插入调试之前，将映射地址的内容复制到存储器的单独部分。在步骤370，断点的地址将写入与特定第二执行环境，如APU120的局部存储器125相关联的断点寄存器。在一个实施例中，每个这些单独的执行环境都有一个独立的断点寄存器集合。

在步骤380，方法300确定是否对应于当前模块的断点请求表中的所有断点都已插入到加载模块中。如果没有，执行步骤360，然后确定下一个断点的映射地址。如果对应于当前模块的断点请求表中的所有断点都已插入到加载模块中，方法300结束。

现在看图4，其中公开了用于删除断点的方法400。尽管方法400说明了删除主存储器代码（即以在第一执行环境中执行为目标的代码）以及相关的附加代码模块中的断点，但不要求在主代码模块中设置或删除断点。

在开始步骤410后，方法400确定是否存在一个断点，该断点与要在MPU130的主模块中删除断点的当前的映射标记相关。如果不存在这样的断点，则执行步骤450。但是，如果主模块中有这样的断点，则在步骤430，删除相应的陷阱指令，并在步骤440恢复原始代码。

在步骤450，方法400确定是否存在与针对APU120的模块相关的断点。在一个实施例中，通过将映射标记映射到一个动态事件列表来确定，即通过列举可能的断点来确定，这些断点与相关断点要被删除的映射标记相关联。在另一个实施例中，可以用用户指定的断点号或

其他标识符来访问断点请求表。如果没有与APU120相关的断点，则方法400在步骤460结束，但是，如果有与APU120相关的断点，则方法400执行步骤470。

但是，在步骤470，如果模块仍加载到存储器中，则访问映射表，然后确定该断点的存储器偏移。在一个实施例中，在步骤480删除选定加载模块中的陷阱指令，然后在步骤485恢复模块中的原始代码。在另一个实施例中，通过刷新包含与此断点相关的地址的断点寄存器来替换步骤480和485。

在步骤490，还删除了断点的标记。在一个实施例中，这包括从主存储器160中相应的模块映像删除陷阱指令。在另一个实施例中，从断点请求总表删除对应的断点请求。因此，无需在加载时间插入此特定的断点，就可以加载或卸载选定的模块。在步骤490之后，执行步骤492。通常，步骤492确定是否还有模块带有为断点设置，并加载到另一个单独的执行环境，如第二APU120的局部存储器125，的标记。如果有多个加载了断点的模块，则执行步骤470。如果没有，则结束步骤495。

应该理解，本发明可以采用许多形式和实施例。因此，在不背离本发明的实质或范围的前提下，可以对上述内容进行各种改变。本文所概括的功能允许用各种可能的编程模型实现。不应把本文公开的内容看成优于任何特定的编程模型，这些内容都是针对构建这些编程模型的基础机制。

在引用某些优选实施例对本发明进行说明之后，应该认识到所公开的实施例都是说明性的，而不在实质上构成限制，各种各样的改变、修改、变更和替换都已涵盖在上述公开内容中。在一些情况下，可以使用本发明的一些特征，而相应地不使用另外一些特征。根据上述关于优选实施例的说明，许多改变和调整对本领域技术人员来讲都是明显而必须的。因此，以与发明的范围相一致的方式，用合理宽的范围解释本说明书所附加的权利要求书是适当的。

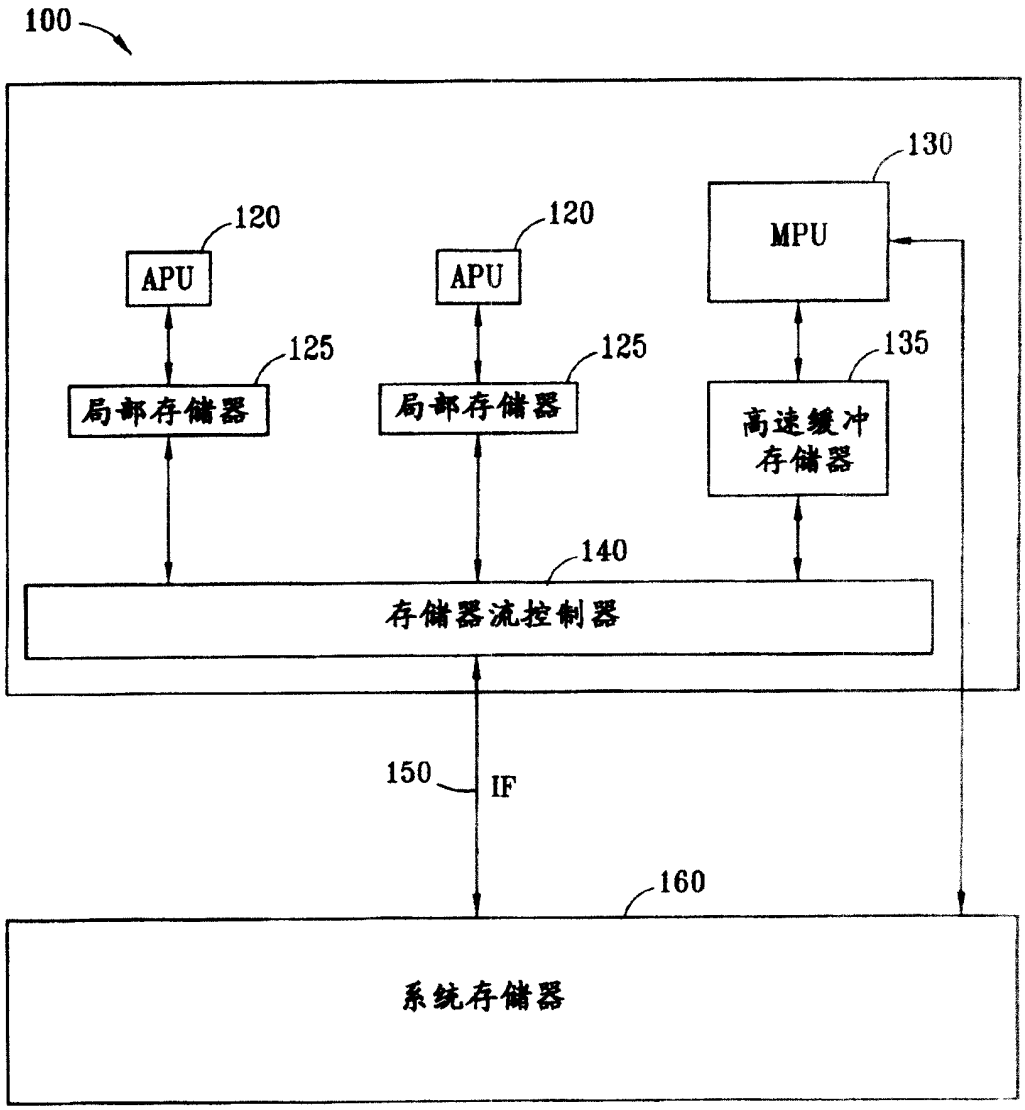

100
130
MPU
120
APU
120
APU
125
局部存储器
125
局部存储器
135
高速缓冲存储器
140
存储器流控制器
150
IF
160
系统存储器

图1

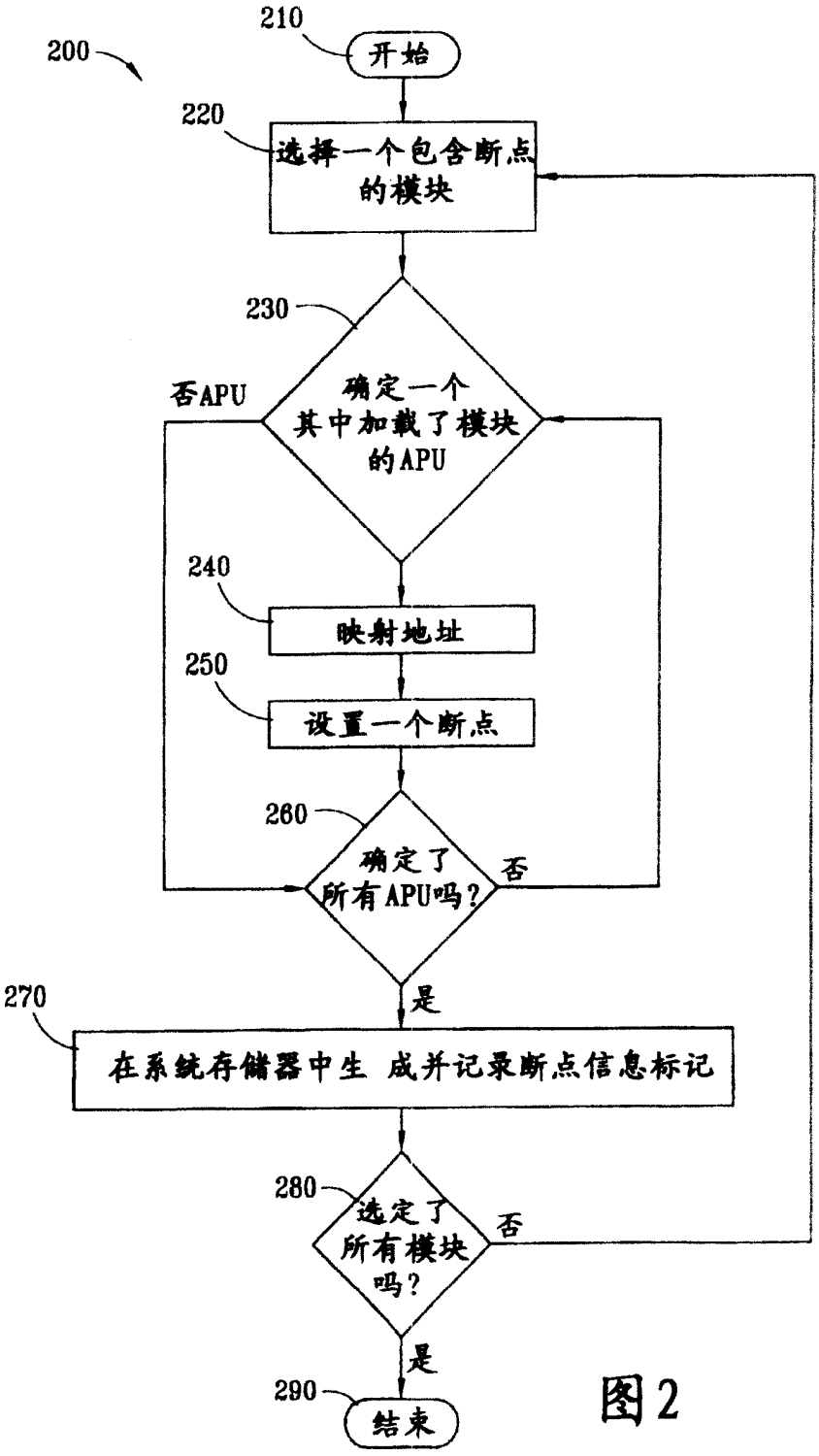

200
210
开始
220
选择一个包含断点的模块
230
确定一个其中加载了模块的APU
否APU
240
映射地址
250
设置一个断点
260
确定了所有APU吗?
否
是
270
在系统存储器中生 成并记录断点信息标记
280
选定了所有模块吗?
否
是
290
结束

图2

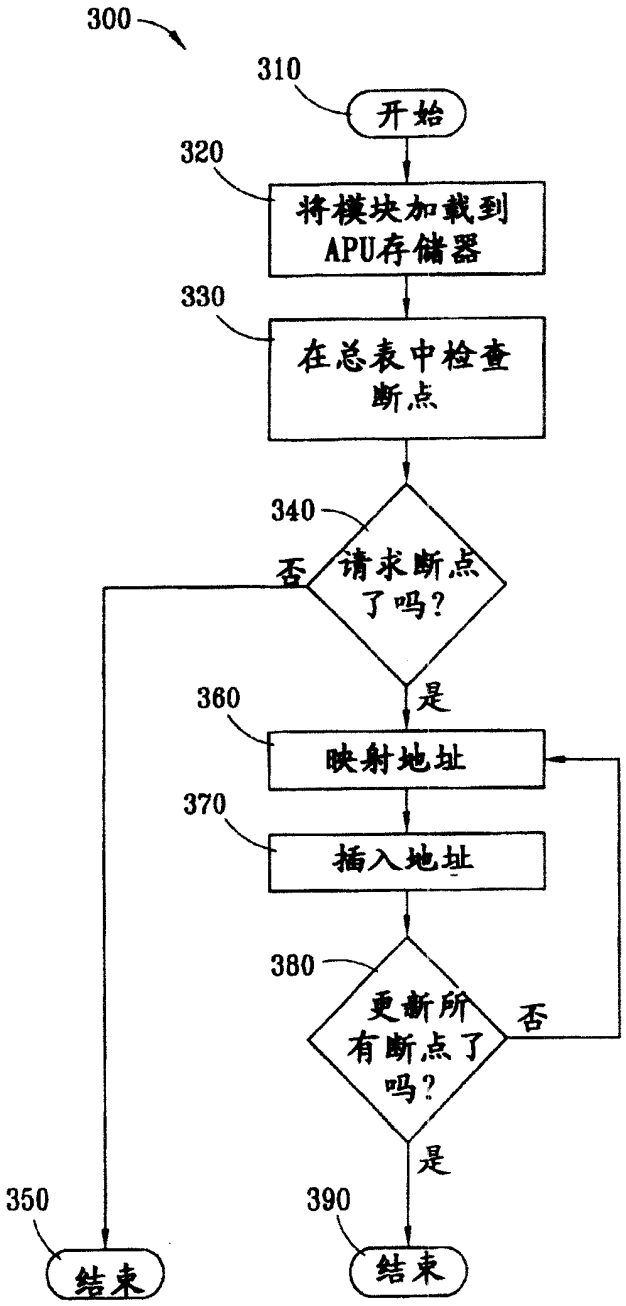

300
310
开始
320
将模块加载到APU存储器
330
在总表中检查断点
340
请求断点了吗?
否
是
360
映射地址
370
插入地址
380
更新所有断点了吗?
否
是
350
结束
390
结束

图3

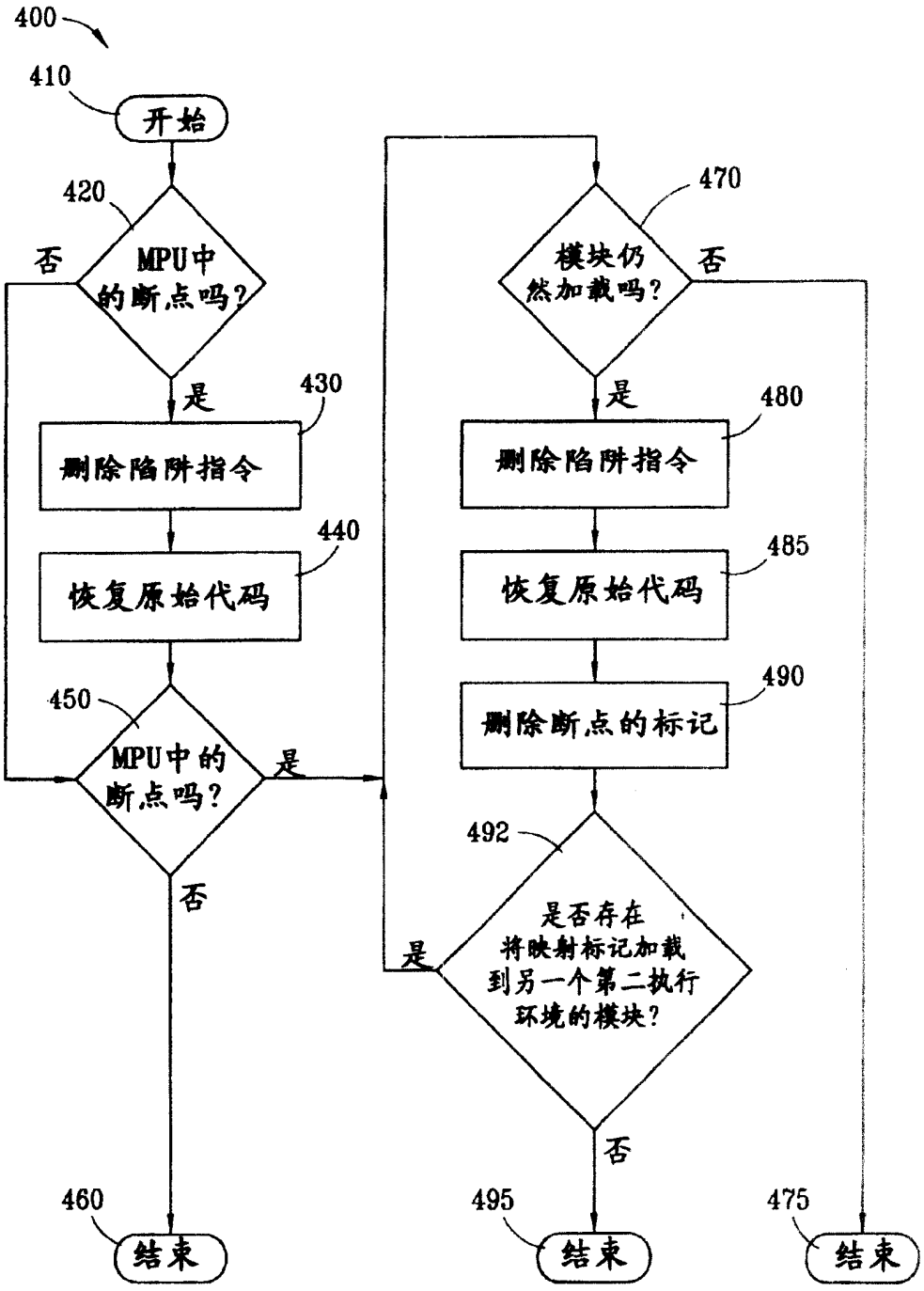

400
410
开始
420
MPU中的断点吗?
否
是
430
删除陷阱指令
440
恢复原始代码
450
MPU中的断点吗?
是
否
460
结束
470
模块仍然加载吗?
否
是
480
删除陷阱指令
485
恢复原始代码
490
删除断点的标记
492
是否存在将映射标记加载到另一个第二执行环境的模块?
是
否
495
结束
475
结束

图4