[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 11/36 (2006.01)

[12] 发明专利说明书

专利号 ZL 200410035161.0

[45] 授权公告日 2008 年 11 月 5 日

[11] 授权公告号 CN 100430904C

[22] 申请日 2004.4.21
[21] 申请号 200410035161.0
[30] 优先权
[32] 2003.4.21 [33] US [31] 10/419，384
[73] 专利权人 微软公司
地址 美国华盛顿州
[72] 发明人 J·M·斯陶尔
M·M·马格鲁德尔
[56] 参考文献
US20020129337A1 2002.9.12
CN1221146A 1999.6.30
JP2001-60161A 2001.3.6
审查员 陈 颖

[74] 专利代理机构 上海专利商标事务所有限公司
代理人 陈 斌

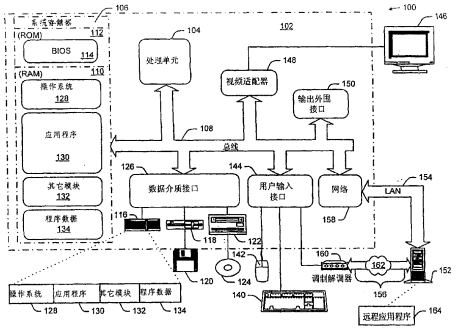
权利要求书 4 页 说明书 12 页 附图 3 页

[54] 发明名称

Just-My-Code 调试技术

[57] 摘要

讨论了用于 Just-My-Code(JMC)调试的系统和方法。在一个方面，调试探头可自动插入在表示感兴趣代码的各个编程结构中。调试探头是在计算机产生原始代码的计算机程序编译操作的期间被插入的。随后作为一个进程来执行原始代码。该进程可以包括在一个 JMC 单步操作期间自由运行通过不感兴趣代码的执行的一个或多个线程。只能在调试探头的一个有效探头被线程遇上且该线程是通过该线程的 JMC 单步步进时，才能暂停一个或多个线程中的一个线程。



1．一种 适 用 于 Just-My-Code 即 JMC 调试的方法，其特征在于，该方法包括：

自动识别一组表示感兴趣代码的编程结构；

自动将调试探头插入到表示感兴趣代码的各个编程结构中，所述调试探头在计算机的计算机程序编译操作期间被插入，以产生原始代码；

作为一个进程执行原始代码，该进程包括在一个 JMC 单步步进操作期间自由运行通过不感兴趣代码的执行的一个或多个线程，在一个或多个线程中的一个线程只能在调试探头的一个有效探头被该线程遇上且该线程是以 JMC 单步步进通过该进程时才可在感兴趣的代码中暂停；和

其中，JMC 单步步进包括 JMC 单步进入操作、JMC 单步退出操作、或者 JMC 单步跳过操作。

2．如权利要求 1 所述的方法，其特征在于，所述各个编程结构是类方法和/或静态函数。

3．如权利要求 1 所述的方法，其特征在于，每个调试探头包括一个编译时间常数和一个对调试服务的调用，所述方法还包括：

响应于遇上所述一个有效探头，间接引用对应于所述一个有效探头的编译时间常数，以确定执行的一个或多个线程中的任何一个是否为 JMC 单步步进的；

如果存在至少一个线程是 JMC 单步步进的，则调用所述调试服务来确定所述一个或多个线程中的所述一个线程是否为 JMC 单步步进的；

响应于确定所述一个线程是 JMC 单步步进的，将所述一个线程暂停在所述一个有效探头之后一行代码处；和

响应于确定所述一个线程不是 JMC 单步步进的，不将断点自动插入在代码中来暂停所述一个线程。

4．如权利要求 1 所述的方法，其特征在于，每个调试探头包括一个编译时间常数和一个对调试服务的调用，所述方法还包括：

响应于遇上所述一个有效探头，间接引用对应于所述一个有效探头的编译时间常数，以确定执行的一个或多个线程中的任何一个是否为 JMC 单步步进的；

如果存在至少一个线程是 JMC 单步步进的，则调用所述调试服务来确定所述一个或多个线程中的所述一个线程是否为 JMC 单步步进的；和

响应于确定所述一个线程是 JMC 单步进入的，将一个断点设置在紧跟着对所述一个线程 JMC 单步进入的编程结构的调用后的一行代码中。

5．如权利要求 1 所述的方法，其特征在于，每个调试探头包括一个编译时间常数和一个对调试服务的调用，所述方法还包括：

响应于遇上所述一个有效探头，间接引用对应于所述一个有效探头的编译时间常数，以确定执行的一个或多个线程中的任何一个是否为 JMC 单步步进的；

如果至少一个线程是 JMC 单步步进的，则调用所述调试服务来确定所述一个线程是否为 JMC 单步步进的；和

响应于确定所述一个线程是 JMC 单步退出的：

识别对一个感兴趣编程结构的一个第一堆栈帧的一个返回地址；和

将一个断点插入在所述返回地址上。

6．如权利要求 1 所述的方法，其特征在于，每个调试探头包括一个编译时间常数和一个对调试服务的调用，所述方法还包括：

响应于遇上所述一个有效探头，间接引用对应于所述一个有效探头的编译时间常数，以确定执行的一个或多个线程中的任何一个是否为 JMC 单步步进的；

如果存在至少一个线程是 JMC 单步步进的，则调用所述调试服务来确定所述一个或多个线程中的所述一个线程是否为 JMC 单步步进的；和

响应于确定所述一个线程是 JMC 单步跳过的：

确定所述一个线程是否接近于方法的结束端；和

响应于确定所述一个线程接近于方法的结束端，将所述一个线程暂停在被执行的下一个感兴趣代码的第一行处。

7．一种 适用于 Just-My-Code 即 JMC 调 试的计算机设备，该计算机设备包括：

自动识别一组表示感兴趣代码的编程结构的装置；

用于自动将调试探头插入到表示感兴趣代码的各个编程结构中的装置，所述调试探头在计算机的计算机程序编译操作期间被插入，以产生原始代码；和

用于作为一个进程执行原始代码的装置，该进程包括在一个 JMC 单步步

进操作期间自由运行通过不感兴趣代码的执行的一个或多个线程，在一个或多个线程中的一个线程只能在调试探头的一个有效探头被该线程遇上且该线程是以 JMC 单步步进通过该进程时才可在感兴趣的代码中暂停，

其中，所述 JMC 单步步进包括 JMC 单步进入操作、JMC 单步退出操作、或者 JMC 单步跳过操作。

8．如权利要求 7 所述的计算机设备，其特征在于，所述各个编程结构是类方法和/或静态函数。

9．如权利要求 7 所述的计算机设备，其特征在于，每个调试探头包括一个编译时间常数和一个对调试服务的调用，所述计算机设备还包括：

响应于遇上所述一个有效探头，间接引用对应于所述一个有效探头的编译时间常数，以确定执行的一个或多个线程中的任何一个是否为 JMC 单步步进的装置；

如果存在至少一个线程是 JMC 单步步进的，则调用所述调试服务来确定所述一个或多个线程中的所述一个线程是否为 JMC 单步步进的装置；

响应于确定所述一个线程是 JMC 单步步进的，将所述一个线程暂停在所述一个有效探头之后一行代码处的装置；和

响应于确定所述一个线程不是 JMC 单步步进的，不将断点插入在代码中来暂停所述一个线程的装置。

10．如权利要求 7 所述的计算机设备，其特征在于，每个调试探头包括一个编译时间常数和一个对调试服务的调用，所述计算机设备还包括：

响应于遇上所述一个有效探头，间接引用对应于所述一个有效探头的编译时间常数，以确定执行的一个或多个线程中的任何一个是否为 JMC 单步步进的装置；

如果存在至少一个线程是 JMC 单步步进的，则调用所述调试服务来确定所述一个或多个线程中的所述一个线程是否为 JMC 单步步进的装置；

响应于确定所述一个线程是 JMC 单步进入的，将一个断点设置在紧跟着对所述一个线程 JMC 单步进入的编程结构的调用后的一行代码中的装置。

11．如权利要求 7 所述的计算机设备，其特征在于，每个调试探头包括一个编译时间常数和一个对调试服务的调用，所述计算机设备还包括：

响应于遇上所述一个有效探头，间接引用对应于所述一个有效探头的编译时间常数，以确定执行的一个或多个线程中的任何一个是否为 JMC 单步步进的

装置；

如果存在至少一个线程是 JMC 单步步进的，则调用所述调试服务来确定所述一个或多个线程中的所述一个线程是否为 JMC 单步步进的装置；

响应于确定所述一个线程是 JMC 单步退出的：

识别对一个感兴趣编程结构的一个第一堆栈帧的一个返回地址的装置；和

用于将一个断点插入在所述返回地址上的装置。

12．如权利要求 7 所述的计算机设备，其特征在于，每个调试探头包括一个编译时间常数和一个对调试服务的调用，所述计算机设备还包括：

响应于遇上所述一个有效探头，间接引用对应于所述一个有效探头的编译时间常数，以确定执行的一个或多个线程中的任何一个是否为 JMC 单步步进的装置；

如果存在至少一个线程是 JMC 单步步进的，则调用所述调试服务来确定所述一个或多个线程中的所述一个线程是否为 JMC 单步步进的装置；和

响应于确定所述一个线程是 JMC 单步跳过的：

确定所述一个线程是否接近于方法的结束端的装置；和

响应于确定所述一个线程接近于方法的结束端，将所述一个线程暂停在被执行的下一个感兴趣代码的第一行处的装置。

Just-My-Code 调试技术

技术领域

本发明涉及软件开发。

技术背景

调试一般都涉及到调试器的使用，允许软件开发者能够观察计算机程序的运行时间的行为并定位语法错误。诸如停止命令之类的某些调试命令允许程序员可以任何时间暂停执行一个正在运行的进程。然而，在已经到达代码中的预定位置时，程序员所人工插入的断点允许暂停该进程。被调试程序可以自由地运行，直至它碰到在指令流中的断点操作代码，在该断点上，操作系统（OS）将会暂停被调试程序，直至调试器继续该被调试程序。于是，当调试一个计算机程序时，程序可以是运行的（例如，作为一个进程在执行），也可以是暂停的。一些调试命令，例如，单步进入、单步跳过和单步退出的命令，只有在断点模式中才是已作用（即，当被调试程序停时），并且允许程序员步进通过程序状态，来观察和/或更改变量、指针和/或其它类似的内容。

传统的步骤是通过在经过指令流的静态分析中所获得的战略点上设置断点操作代码并随后自由运行直至遇上（碰上）相应的插入码来完成的。例如，单步进入把插入码放在被单步进入的函数的开始处，单步跳过把插入码放置在正需跳过的一行之后，以及单步退出将插入码放置在一旦将被执行当前函数就返回的指令处上。不幸的是，这类常规单步命令并不允许程序员自动跨越“不感兴趣的代码”的。相反，正如以上所讨论的那样，程序员需要在所感兴趣的代码中人工插入断点并且执行直至碰到一个断点，和/或人工单步通过所不感兴趣的代码以得到所感兴趣的代码。换句话说，程序员可以需要迭代（一行又一行）单步进入和单步通过所不感兴趣的代码，单步跳过在不感兴趣代码中的一个函数和可能区域，和/或单步退出在不感兴趣代码中的一个函数和可能区域。

例如，单步进入和单步跳过命令只是在它们处理函数调用方面是不同的。这两个命令都可以指令调试器执行源代码的下一行。如果该行包含一函数调用，则单步进入命令就只执行调用其本身，随后在该函数中代码的第一行处暂

停，且与代码的第一行是否为不感兴趣的代码无关。单步退出命令可执行整个功能，随后在出走该函数外的第一行处暂停，且与该第一行是否为不感兴趣的代码无关。在嵌套的函数调用中，单步进入命令可使步骤进入到最深的嵌套函数中。例如，如果单步进入命令用于像 F1（F2（））的调用，则调试器就步进入功能 F2。单步退出命令可用于通过继续程序执行直到函数返回来单步退出该函数，并随后在调用函数中的返回点处中断执行，这与该返回点是否对应于不感兴趣的代码无关。

另外，尽管可以指令调试器对代码中每个相应行迭代执行单步进入命令，以能够单次通过代码中的一部分代码，这一过程实际上降低了调试过程的性能，以及实际上也增加了过程的死锁（即，由于调试而引起的同一缓存线的竞争）的风险。例如，为了仿真多级迭代单步进入的操作，调试器，对代码的各个行，都在下一指令的位置上插入一个断点，运行过程，缓存有关执行，并随后检查指令指针，以确定预先配置的停止点是否已经到达（或者直至调试器已被人工停止）。应该意识的是，这类迭代操作，如果调试器在另一例外退出之前不能执行多于一个单行的代码（即，代码只允许在一个非常短的时间内“自由运行”），则就会相当妨碍调试器的性能并增加过程死锁的可能性。为了讨论的目的，自由运行的代码可以是一个代码的逻辑块，例如，一种允许无妨碍执行到遇上断点的方法或遇上所不感兴趣代码的方法。

当正在调试的计算机程序设计成集成了大量代码且程序员对调试不感兴趣的复杂环境中操作时，这类调试器的限制就变得特别成问题。这类不感兴趣的代码可以包括，例如，程序员不能写入、已经调试过的代码，其它共享的代码、网络的服务代码、互用性框架代码，和/或类似其它等等。在这种情况下，现有技术调试不允许程序员轻易跳过不感兴趣代码的程序就需要花费时间和密集的人力，这对新手或有经验程序员来说，同样都是非常困难的。

因此，就非常希望允许用户只调试感兴趣的代码，而无需人为地在感兴趣的代码中设置断点，和/或人工地步进通过不感兴趣代码以达到感兴趣的代码的系统和方法。

发明内容

本发明讨论了适用于 Just-My-Code 代码（JMC）调试的系统和方法。一个方面，调试探头可以自动插入在表示感兴趣代码的各个程序结构中。调试探头

是在计算机产生原始代码的计算机程序编译操作过程中插入的。原始代码随后作为一个过程来执行。该过程可以包括一个或多个在一个 JMC 步进操作过程中自由运行通过不感兴趣代码的执行线程。只有当线程遇到一个有效的调试探头时，一个或多个线程中的一个线程才能在感兴趣的代码中暂停，同时该线程是 JMC 单步通过该过程。

附图的简要说明

以下详细描述将参考附图进行讨论。在附图中，每个组件标号数的最左边的数表示该组件第一次出现的特定的附图。

图 1 显示了一例示例性计算环境，在该计算环境中可以实现适用于调试 Just-My-Code 代码的系统和方法。

图 2 显示了图 1 所示的系统存储器的另一示例性方面，它包括适用于调试 Just-My-Code 代码的应用程序和程序数据。

图 3 显示了一例适用于调试 Just-My-Code 代码的示例性流程。

详细描述

概述

针对常规调试技术中上述讨论的限制，本发明讨论了适用于调试 Just-My-Code 代码的系统和方法，它可以在调试操作过程中自动地跳过不感兴趣的代码。为了讨论的目的，不感兴趣的代码是一种代码，但是无论怎样的原因，这一代码都不能认为是在调试时参与程序员调试工作有关的代码（例如，程序员不能写入的代码，已经调试过的代码，等等）。“感兴趣的代码”是相对于不感兴趣的代码的，因为它是程序员在调试操作过程中所感兴趣的。

特别是，本发明的各个实施例都允许一个调试应用（一个调试器）可以将任意方法标记在所感兴趣的汇编程序中。在从该汇编程序产生原始代码的 Just-In-Time（JIT）编译器操作的过程中，调试探头自动地插入到所感兴趣的方法中。使用该调试器，用户可以将原始代码下载到过程中，以便于调试。响应 JMC 步进的操作，只有在 JMC 单步线程遇到了所感兴趣的方法时，过程执行才可以自动暂停。所有非 JMC 步进线程都可以通过所感兴趣的和所不感兴趣的代码自由运行。这意味着用户不再需要在所感兴趣的代码中人工设置断点，

也不再需要程序员费力地通过（即，执行常规的单步操作）所不感兴趣的代码而达到所感兴趣的代码。

示例性操作系统

回到附图，在该附图中，相同的标号数指示着相同的元件，并以一个适当的计算环境的实现来说明本发明。尽管这并不是所必需的，但是本发明的讨论采用了计算机可执行指令的主要内容，例如，由一台个人计算机执行的程序模块。程序模块一般可包括，例程、程序、对象、元件、数据结构、等等，它们可用于执行特殊的任务或实现特殊的抽象数据类型。

图 1 说明了一例适当的计算环境 100 实例，该实例依次讨论了所能够（全部或部分）实现的适用于 Just-My-Code 代码（JMC）的系统、装置和方法。示例性计算环境 100 仅仅只是一例适当的计算环境的实例，并不试图建议对本文所讨论的系统和方法的使用及功能作出任何限制。计算环境 100 不应该解释成其具有与计算环境 100 所说明的元件的任何一个或其组合有关的独立性或必要性。

本文所讨论的方法和系统可以众多其它通用或专用计算系统环境或配置进行操作。众所周知，适用于用户的计算系统、环境和/或配置的实例可以包括，但并不限制于，个人计算机、服务器计算机、微处理器系统、基于微处理器的系统、网络 PC、小型计算机、大型计算机、包括任何上述系统或设备的分布式计算环境，以及其它等等。也可以在有限资源的客户机中实现框架紧缩型的或其子集的方式，例如，手持计算机或者其它计算设备。本发明也可以分布式计算环境来实现，在这类计算环境中，可以由通过通讯网络连接的远程处理设备来执行任务。在一个分布式计算环境中，程序模块可以放置于本机和远程存储器存储设备中。

正如图 1 所示，计算环境 100 包括一个采用计算机 102 形式的通用计算设备。计算机 102 的元件可以包括，但并不限制于，一个或多个处理器或处理器单元 104，一个系统存储器 106，以及一个总线 108，其中，该总线将包括系统存储器 106 的各种不同的系统元件与处理器 104 相耦合。系统总线 108 表示几种类型的总线结构中的任何一种或多种，它包括，一种存储器总线或存储器控制器，一种外围总线，一种加速图形端口，以及使用多种总线结构中任一种的一个处理器或本机总线。举例来说，但并不限制于，这类架构可以包括工业标准结构（ISA）总线、微通道结构（MCA）总线、视频电子标准协会（VESA）

本机总线、以及外围元件互连（PCI）总线（也称之为 Mezzanine 总线）。

计算机 102 典型地包括多种计算机可读介质。这类介质可以是计算机 102 可以访问的任何一种有效介质，并且它可以包括易失性或非易失性介质，可移动或不可移动的介质。在图 1 中，系统存储器 106 包括易失性存储器形式，例如，随机存取存储器（RAM）110；以及非易失性存储器形式，例如，只读存储器（ROM）112，的计算机可读介质。基本输入/输出系统（BIOS）114 存储于 ROM 112 中，它包含着诸如在启动过程中有助于在计算机 102 各元件之间传递信息的例程。RAM 110 一般包含着数据和/或程序模块，它可以由处理器 104 进行即时访问和/或当前正被操作。

计算机 102 还可以包括其它可移动/非移动性，易失性/非易失性计算机存储介质。例如，图 1 说明了一个适用于读写的非移动且非易失性的磁性介质的硬盘驱动器 116（未显示且一般可称之为“硬盘驱动器”），一个适用于读写的可移动且非易失性磁盘 120 的磁盘驱动器 118（即，软盘），以及一个适用于读写的可移动且非易失性光盘 124 的光盘驱动器 122，光盘可包括 CD－ROM/R/RW，DVD－ROM/R/RW/＋R/RAM 或者其它光介质。硬盘驱动器 116，磁盘驱动器 118 和光盘驱动器 122 每一个采用一个或多个接口 126 与总线 108 相连接。

驱动器和相关的计算机可读介质提供了计算机可读指令、数据结构、程序模块、以及其它适用于计算机 102 的数据的非易失性存储。虽然本文所讨论的示例性环境采用了一个硬盘，一个可移动的磁盘 120 以及一个可移动的光盘 124，但是，本领域的熟练技术人士应该理解的是，在该示例性计算环境中也可以使用其它类型能够存储计算机所能访问数据的计算机可读介质，例如，磁带盒、闪存存储卡、数字视频光盘、随机存取存储器（RAMs）、只读存储器（ROM），以及其它类似等等。

用户可以通过输入设备，例如，键盘 140 和定位设备 142（例如，“鼠标器”）向计算机 102 提供命令和信息。其它输入设备（未显示）可以包括麦克风、操纵杆、游戏垫、碟型卫星天线、串行端口、扫描仪、摄像机、等等。这些和其它输入设备都可以通过与总线 108 相耦合的用户输入界面 144 连接着处理单元 104，但是也可以通过其它接口和总线结构来连接，例如，并行端口、游戏端口或通用串行总线（USB）。

监视器 146 或者其它类型的显示设备也通过一个接口（例如，视频适配器

148）与总线 108 相连接。除了监视器 146 之外，个人计算机一般包括其它外围输出设备（未显示），例如，扬声器和打印机，它们可以通过输出外围接口 150 相连接。

计算机 102 可以在使用与一个或多个远程计算机（例如，远程计算机 152）逻辑连接的网络环境中操作。远程计算机 152 可以包括与计算机 102 有关的本文所讨论的许多或所有元件和性能。在图 1 中所显示的逻辑连接可以是一个局域网（LAN）154 和一般的广域网（WAN）156。这类网络环境在办公室中是常见的，包括企业内部范围的计算机网络，企业内的互联网和因特网。

当在 LAN 网络环境中使用时，计算机 102 通过网络接口或适配器 158 与 LAN 154 相连接。当在 WAN 网络环境中使用时，计算机一般可包括一个调制解调器 160 或者适用于建立通过 WAN 156 通讯的其它部件。调制解调器 160 可以是内置的或者是外置的，它可以通过用户输入界面 144 或者其它合适的机制与系统总线 108 相连接。图 1 中所说明的是一个通过因特网的 WAN 的特殊实现。这里，计算机 102 采用调制解调器 160 来建立通过因特网 162 与至少一个远程计算机 152 的通讯。

在一个网络环境中，所表示的程序模块与计算机 102 有关或者部分与计算机 102 有关，该程序模块存储一个远程存储器存储设备中。即，正如图 1 所说明的那样，远程应用程序 164 可以驻留在远程计算机 152 的一个远程设备中。应该理解的是，所显示和所讨论的网络连接都是示例性的，也可以使用在计算机之间建立通讯链路的其它装置。

大量的程序模块可以存储于硬盘、磁盘 120、光盘 124、ROM 112、或者 RAM 110，还可包括一个提供一个运行时间的环境、适用于 Just-My-Code 代码（JMC）调试的应用程序 130、其它程序模块 132（例如，设备驱动器等等），以及诸如源代码的程序数据 134、中间汇编、等等的操作系统。

图 2 是一个进一步显示图 1 所示系统存储器 106 的示例性方面的方框图，它可以包括适用于 Just-My-Code 代码（JMC）调试应用程序 130 和程序数据 134。应用程序 130 和程序模块 134 包括，例如，一个过程 202，一个初级编译器 204，一个适用于调试该过程 202 的调试程序（“调试器”）206，以及一个共享的运行时间组件（“运行时间”）208。过程 202 是已经编译成汇编程序（未显示）的源代码（未显示），并依次已经编译成原始代码 210；年个 202 表示通过调试器 206 所执行的原始代码 210。

源代码可以表示成以任何类型的计算机编程语言写成的计算机编程代码。具有调试开关（即，“/debug”） 被打开的初级编译器 204，将源代码编译成汇编程序（未显示）。这样一个初级变压器可以是任何类型的计算机程序语言编译器，例如，C、C＋＋、C#，VB，和/或已经改进成了实现 JMC 调试的其它类型编译器。汇编程序是众所周知，并且可表示成在源代码变换成平台所指定的适用于执行的原始代码中的一个中间阶段。为此，这样一个汇编程序可以包括，例如，独立于中间语言（IL）和相关元数据的平台/处理器。

用户可以将汇编程序下载到调试器 206。适用于将一个汇编程序下载到一个应用的技术是众所周知的。例如，一个汇编程序可以通过指定（即，通过命令行或者 UI）汇编程序的名称或者通过选择汇编程序下载到调试器 206。在汇编程序下载的操作过程中，调试器 206 在汇编程序的不同位置上产生一列表感兴趣的代码，这感兴趣代码是与不感兴趣代码相比较而言。用户代码（迭代：源代码变换成一个汇编程序，以及，最后，正如以下所讨论的，变换成原始代码，该原始代码可作为一个过程来执行）一般都包括某些感兴趣代码和不感兴趣代码的组合。然而，用户代码并不包括属于运行时间 208 的主机代码。在一个实施例中，感兴趣的代码可以通过用户输入到用户输入界面 144 的任何一个方面来产生（图 1）。例如，用户可以选择（即，通过一个定位设备、键盘、语音激励技术，等等）来指定在调试服务的对话框中所显示的多种方法中的一种方法，或者用户可以将编程结构的名称打入到命令行的界面，和/或其它等等。调试器 206 通过所暴露的调试 API 216 通知所识别的感兴趣的代码的调试服务 214，而调试 API 216 也可以两个部分来实现，一个部分可以作为一个运行时间 208 的服务，而另一部分可以作为执行 216 的公共部分组件且可以由调试器 206 来消费。

在该点上，用户可指示调试器 206 开始 JMC 调试操作，例如，可通过使能一个 JMC 菜单项目、命令行，或者其它 JMC 使能指令，以及发布一个运行或启动命令。这就使得调试器 206 指示 JIT 编译器 220 将汇编程序变换成原始代码 210，以便于以后能作为过程 202 来执行。

在汇编程序变换的操作过程中，JIT 编译器 220 将 JMC 使能元件、标志 222 和调试探头 224 插入原始代码 210 的编程结构中（即，执行文件/二进制文件、动态连接库、分类目标和方法、静态函数，和/或其它类似）。在该实施例中，在分类方法和/或静态函数的前言之后立即插入该 JIT 使能元件。JIT 编译器 220

通过分解感兴趣的代码列表/标识 212 来识别这类分类方法和/或静态函数，正如以上所讨论的，感兴趣的代码列表/标识 212 是由调试器 206 产生的。

调试器探头 224 参考一个相关的标志 222，并通过一个编译时间常数指针值，来确定是否请求调试服务 214 来仿真已经遇上有效探头 224 所执行的特殊线程是否执行一个 JMC 单步操作，并且如果是的话，则暂停该过程 202。正如我们现在所讨论的，这样的确定对于不激活的标志是不会产生，而是仅对激活的标志产生的。一个典型的调试探头 24 可以下列方式来表示：

If(*pFlag){call JMC _Probe}，

式中，pFlag 是一个指向对应标志 222 的编译时间常数指针（*表示该地址将再作为运行时间的间接引用以产生相关标志 222 的数值）。探头 224 的“JMC_Probe”部分是一个对 JMC_Probe 函数 226 的调用。通过指针间接指向的使用，与所插入的编程结构有关的多个编程的 PFlags 可以全都指向相同的标志 222。

正如所说明的，调用 JMC_Probe 226 是一个条件，它取决于参考标志 222 是否激活或者不激活。为了讨论的目的，一个激活的标志 222 具有一个非零的数值，其中一个不激活的标志具有一个零数值或空数值。当标志 222 是激活的，我们就说该调试探头 224 是激活的（即，是一个有效的探头），反之亦然。例如，当执行的一个线程遇上了一个有效的探头 224，则产生对 JMC_Probe 的调用。相类似，遇上一个不激活的探头 224 就不会产生这类调用。这就意味着翻转触发一个信号标志 222 可以分别激活或不激活各个相关的探头 224。现在，我们讨论这些新颖的 JMC 调试探头 224 在调试过程中是怎样使用的。

当过程 202 已经暂停（通过任何传统的单步机制，例如，通过采用一个传统的断点）时，调试器 206 允许用户通过 JMC 单步命令 228 使得 JMC 步进通过过程 202（即，JMC 单步进入，JMC 单步退出，和 JMC 单步跳过命令）。这类命令可以由用户来选择，例如，通过命令行和/或用户界面 144 的 UI 部分（图 1）。响应于由调试器 206 对调试服务 214 发出的一个 JMC 单步命令 228（即，JMC 单步进入，单步退出，和单步跳过命令 228 的任何一个），调试服务大量使能标志 222，并从而激活相关的探头 224。用户不需要知道那一个探头是激活的。

激活调试探头 224 允许过程 202 的执行线程可以自由运行通过不感兴趣的代码（即，可以基本上全速和不需要暂停）。遇上一个激活探头 224 的任何线

程，只要它是处于感兴趣代码中的，就只有在所遇到的探头 224 通过对 JMC_Probe 226 的调用才能暂停（通过 TriggerMethodEntry 230），以确定该线程是 JMC 单步通过代码的。为了能达到这一点，JMC_Probe 226 通过对调用堆栈的解码来滤去不能进行 JMC 单步操作的线程（见，其它数据 232），从而识别与所遇到的（“已触发的”）调试探头 224 相关方法的指令指针（ip）和帧指针（fp）。如果当前的线程是进行 JMC 单步操作，则 JMC_Probe 226 就将 ip 和 jp 参数发送给 TriggerMethodEntry 230，它可以随后将 ip 映射回该方法以确定该方法是否是感兴趣的方法，正如在感兴趣代码列表/标识 212 中所指示的。（运行时间 208 能够为 JIT 编译器 220 作充分的薄记工作，以完全保证一个调试探头 224 可以通过一个查找表的操作来确定该探头是否处于所感兴趣的代码中）。如果该方法是感兴趣的，则 JMC_Probe 226 就可以在调试探头 224 之后插入一个中断的操作码（即，一个断点 234，JMC_Probe 可以注入所有的断点 234），并且继续过程的执行，使得执行的线程将会击中断点 234 以及在所感兴趣的代码中停止下来。

另外，如果 JMC 单步命令 228 是：

•JMC 单步进入，调试服务 214 可激活所有的 JMC_Probe 224 并且在调用了单步进入的方法之后设置一个断点（不是 JMC_Probe 被插入的断点）。因为 JMC_Probe 224 是处在所有感兴趣代码的起始位置上，并且激活探头，如果调用最后调用了任何“感兴趣”代码（也许非直接地，通过调用最终包括“感兴趣”代码的不感兴趣/框架代码），则该线程将击中一个激活的探头，该探头将调用 TriggerMethodEntry 230—停止该线程，在感兴趣的代码中单步完成。如果该调用未调用任何感兴趣的代码，则该线程将击中所插入的断点，它插入在单步进入方法的调用之后（即，不是 JMC_Probe 插入的断点 234），并且 JMC 单步进入的操作将完成。采用这样的方法，进程 202 可以在 JMC 单步进入起始和它结束之间的时间内全速运行（即，不会被中间断点操作码中断）。

•JMC 单步退出，JMC_Probe 226 自动将一个断点 234 插入在由第一堆栈帧所标识的对一个感兴趣的方法的返回地址上。于是，断点就没有插入在调试探头 224 能够遇上的相同方法中。为了定位具有对一个感兴趣的方法的一个返回地址的第一堆栈帧，调试服务 214 完全走过过程堆栈（即，展开堆栈）。由于“其它数据”232 的“感兴趣方法的数据”部分，调试服务 214 可采用一个或多个运行时间 208 的服务，以确定一个堆栈帧的地址是否为一个感兴趣方法的

返回地址。这类感兴趣方法数据包括，例如，适用于各种感兴趣方法的各自基本上唯一的 ID（即，一个 GUID）。在一个实施例中，在调试探头 224 插入操作的过程中，由 JIT 编译器 220 产生感兴趣方法的数据。

•JMC 单步跳过和线程不在该方法结束的地方执行一个指令的话，就会引起 JMC_Probe 226 产生类似于一个传统单步跳过的行为。如果该线程是在该方法结束的地方，则 JMC_Probe 226 就会产生类似于单步进入和单步退出的行为，使得该线程将停止在所执行的感兴趣代码的第一个实例（第一行）中。

响应 JMC 单步命令 228 的完成，这是由正在暂停的这个 JMC 单步线程所证明的，调试服务 214 禁止所有的探头（调试器 206 仅理解 JMC 单步操作；它不知道有关探头的其它事情）。不激活探头 224 是避免对 JMC_Probe 226 不必要调用的性能最优化，否则这种调用就可能在没有执行的线程是 JMC 单步指令的情况下发生。

在本实施例中，标志 222 是一个按模块基础上的使能/禁止（即，每个模块中被插入一个（1）标志）。采用这样的方式，在该模块中的各种方式都对 pFlag 具有相同的数值。这一特殊的实施例是基于代码是在按模块的基础上的感兴趣和不感兴趣的合理性的。应该意识到，不同的合理性可以以不同的实施例作为一个特殊实施例设计的一个函数来加以实现。例如，标志 222 也可能插入在按方法的基础上；可以对更多的标志 222 作些折中，可以通过改变/触发一单个标志 222 的值来使能较少的探头。

因此，遇到一个有效探头 224 的所有线程都会产生对 JMC_Probe 226 的调用，即使当前的线程不是 JMC 单步执行。只有当 JMC_Probe 226 确定该线程时一个 JMC 单步时，该线程才被暂停。类似的，如果所遇上的线程调试探头 224 不是有效的，则对 JMC_Probe 226 的调用就完全可以跳过，并且该线程可以正常（即，自由地运行）地继续执行通过该进程。

在该实施例中，对 JMC_Probe 功能的调用 226 可以在尺寸上得到充分的优化，因为它没有采用任何参数且没有返回数值。这就提供了用于优化所有的“调用位置”（产生对 JMC_Probe 函数 226 的各个调用的位置）。于是，如果对来自调试探头 224 的 JMC_Probe 的调用是来自探头的（即，当*pFlag!＝0），则该调用就将不再是一个重的加权调用。这就意味着该调用并不需要对堆栈进行压入和弹出函数参数。这依次意味着与需要压入和弹出参数的函数相比较，JMC_Probe 调用位置的加权要轻得多。一线程在遇到一个无效探头 224 要经受

的唯一处理开销是它要进行处理以评估标志 222（通过一个*pFlag）以及跳到插入的调试探头 224 之后的下一个指令。采用这一方式，所讨论的 JMC 调试操作允许线程以基本全速的方式自由运行，以通过所有不感兴趣的代码直至遇上所感兴趣的代码。

在本实施例中，运行时间 208 是基于一种公用语言基础结构（CLI），该公用语言基础结构提供了适用于可执行代码的规范以及它所运行的虚拟执行环境。于是，尽管在图 2 中分别进行了说明，但是过程 208 和运行时间 202 都是相同执行环境中的一个部分（即，过程 202 控制着运行时间 208）。

示实例过程

图 3 显示了一例适用于 JMC 调试的示例性过程。为了讨论的目的，这些过程操作都可以参考图 1 和图 9 所示的程序模块和数据特性来讨论。在方框 302，调试器 206（图 2）识别所感兴趣的计算机程序编程结构（即，对象方法和/或静态功能）。在一个实施例中，这是通过用户输入到用户输入界面 144 的任何方面来完成的（图 1）。例如，用户可以选择（即，通过一个指点设备、键盘、语音激励技术，等等）来指定在调试服务的对话方框中所显示的多种方法中的一种方法，或者用户可以将编程结构的名称键入一个命令行的界面上，和/或类似的。这些用户输入可以存储于所感兴趣的代码列表/标识 212 中（图 2），它可以由运行时间编译器 220 读取（图 2），以识别感兴趣的编程结构（即，感兴趣的代码）。

在方框 304，调试器 206（图 2）指令一个编译器，例如，图 2 所示的 Just-In-Time（JIM）编译器 220，以便于在编译操作过程中将 JMC 使能元件 222 和 224 自动插入计算机程序的各个独立的编程结构（即，方法）。在本实施例中，JIT 编译器 220 编译一个汇编程序，以产生原始代码 210（图 2），并自动地将各个标志 222 和调试探头 224 插入到原始代码 210 中。在方框 306，在调试器 206 的控制下作为进程 202（图 2）来执行原始代码 210（图 2）。该进程 202 包括任何数量的作为该进程的特殊实施的函数的执行线程。

在方框 308，调试器 206（图 2）允许用户输入一个 JMC 单步命令 228 来 JMC 单步步进的一个线程以通过该进程 202。因为该线程是 JMC 单步步进的，所以该线程可称之为一个“感兴趣的线程”。该进程 202 可以是多线程的并且包括一个不是 JMC 单步步进的线程，该线程可以称之为一个“不感兴趣的线程”。一个感兴趣的线程可以自由运行（无暂停的）通过不感兴趣的代码（即，

与在方框 302 中所选择的编程结构中的所选择的/单独的结构所识别的感兴趣代码不相对应的任何代码）。感兴趣线程可由调试服务 214 自动暂停，无论何时只要它首次遇上一个有效的调试探头 224。确定过程 202 的一个线程（图 2）是否为一个感兴趣的线程以及所感兴趣的线程是否为一个在所选择的编程结构中的执行/访问的代码是由所遇上的调试探头 224 在运行时间确定的。于是，所有非 JMC 单步线程都可以自由运行通过在感兴趣代码和不感兴趣代码中所有遇上的调试探头 224。

结论

所讨论的系统和方法提供了 JMC 调试。尽管已经讨论的系统和方法在语言上特定于结构性能和操作方法，但是在权利要求中所限定的主题内容并不一定要限制于上述的特定的性能和操作。相反，只是以实现权利要求的主题内容的示例性形式披露了特定的性能和操作。

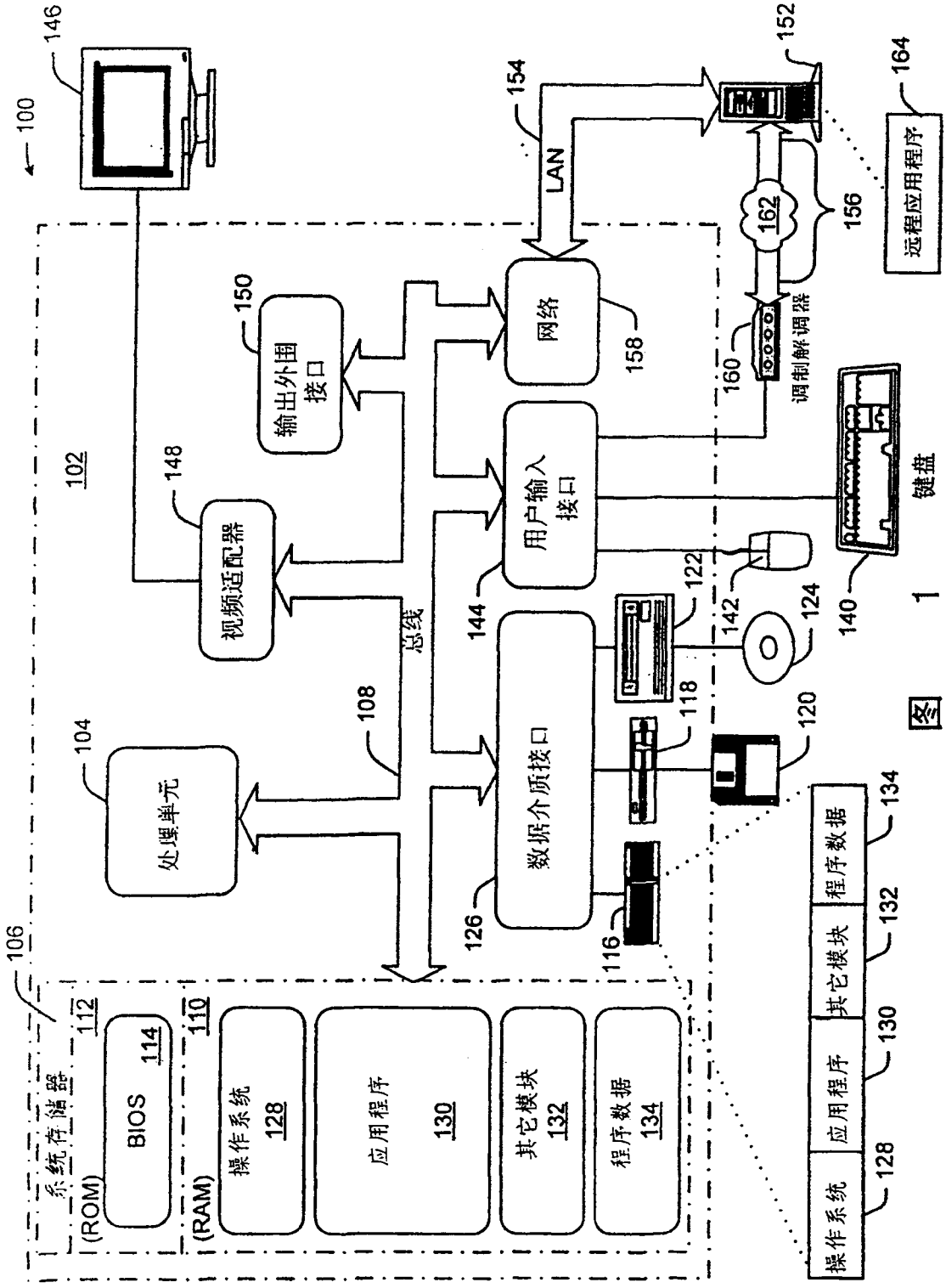

100
102
104
处理单元
106
系统存储器
(ROM)
BIOS 114
112
(RAM)
110
操作系统 128
应用程序 130
其它模块 132
程序数据 134
108
总线
148
视频适配器
150
输出外围接口
146
154
LAN
网络
158
152
162
156
远程应用程序
164
160
调制解调器
用户输入接口
144
键盘
140
122
142
124
118
120
126
116
数据介质接口
操作系统 128
应用程序 130
其它模块 132
程序数据 134

图 1

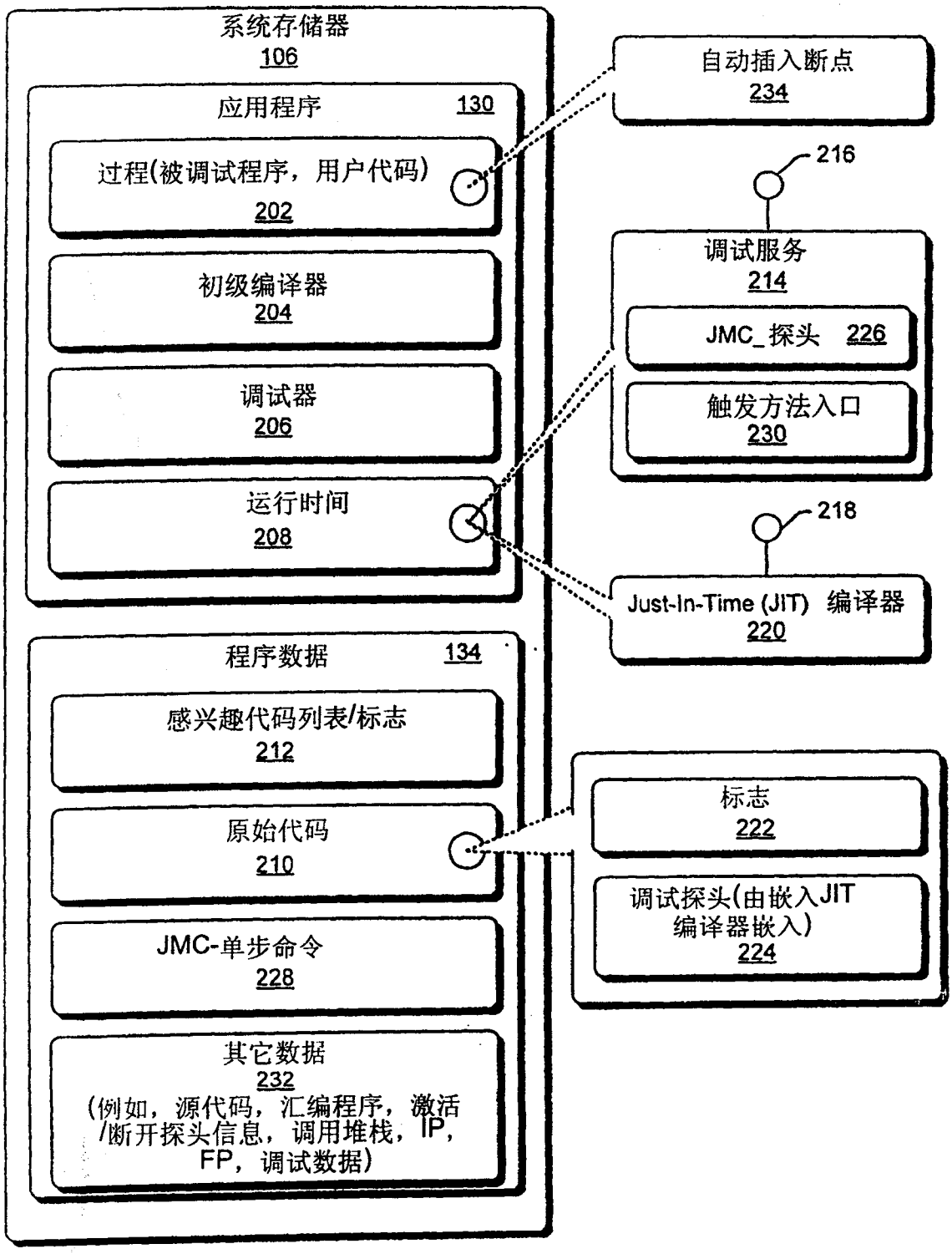

系统存储器
106
应用程序 130
过程(被调试程序，用户代码)
202
初级编译器
204
调试器
206
运行时间
208
程序数据 134
感兴趣代码列表/标志
212
原始代码
210
JMC-单步命令
228
其它数据
232
(例如，源代码，汇编程序，激活/断开探头信息，调用堆栈，IP，FP，调试数据)
自动插入断点
234
216
调试服务
214
JMC_探头 226
触发方法入口
230
218
Just-In-Time (JIT) 编译器
220
标志
222
调试探头(由嵌入JIT编译器嵌入)
224

图 2

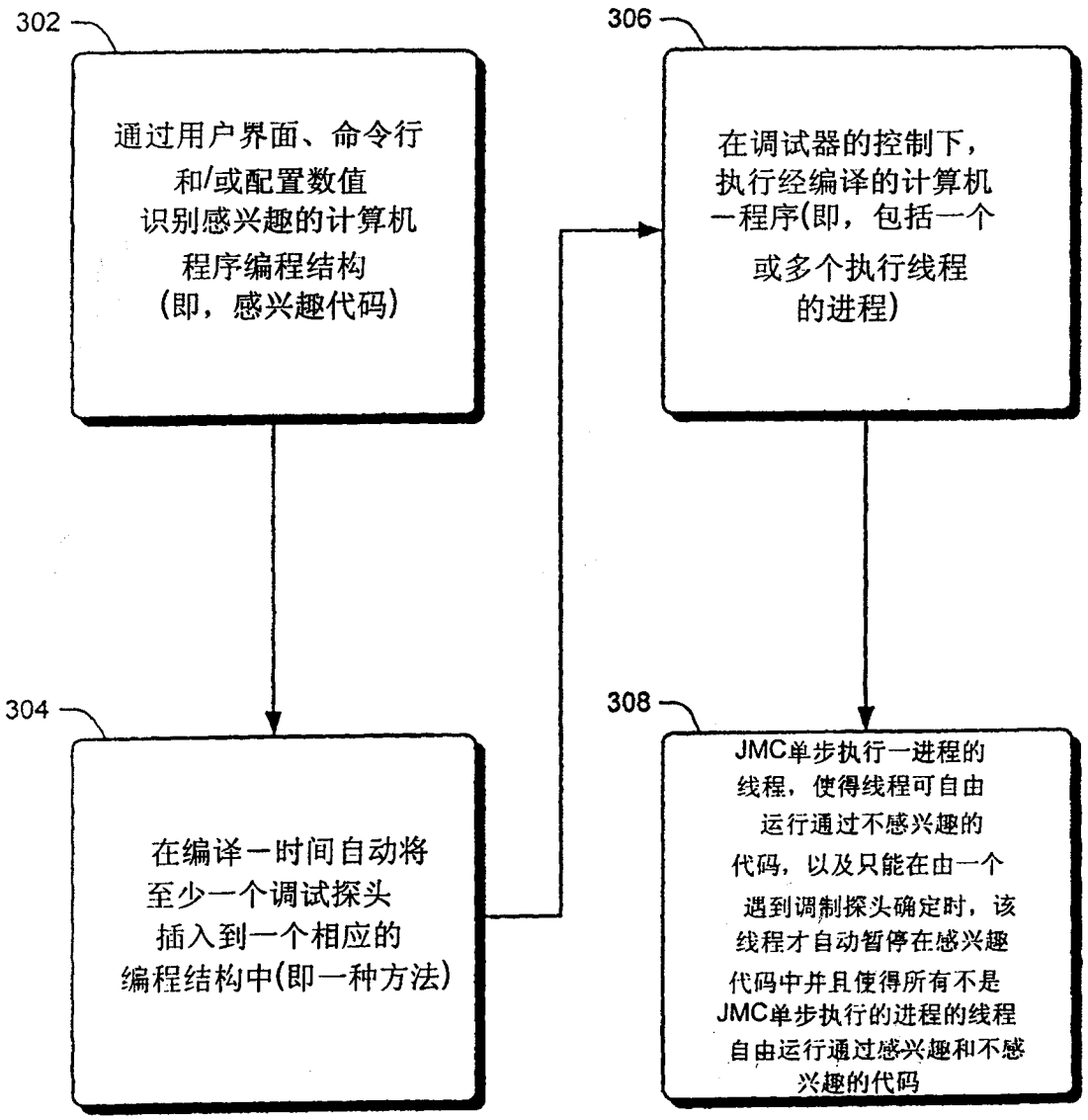

302
通过用户界面、命令行
和/或配置数值
识别感兴趣的计算机
程序编程结构
(即，感兴趣代码)
304
在编译一时间自动将
至少一个调试探头
插入到一个相应的
编程结构中(即一种方法)
306
在调试器的控制下，
执行经编译的计算机
一程序(即，包括一个
或多个执行线程
的进程)
308
JMC单步执行一进程的
线程，使得线程可自由
运行通过不感兴趣的
代码，以及只能在由一个
遇到调制探头确定时，该
线程才自动暂停在感兴趣
代码中并且使得所有不是
JMC单步执行的进程的线程
自由运行通过感兴趣和不感
兴趣的代码

图 3