[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 200410082676.6

[45] 授权公告日 2007 年 11 月 28 日

[11] 授权公告号 CN 100351810C

[22] 申请日 2004.9.27
[21] 申请号 200410082676.6
[30] 优先权
[32] 2003.11.6 [33] US [31] 10/704,117
[73] 专利权人 国际商业机器公司
地址 美国纽约州
[72] 发明人 小吉米·E·德威特
弗兰克·E·莱文
克里斯托弗·M·理查森
罗伯特·J·厄克特
[56] 参考文献
US6009514A 1999.12.28
US5987598A 1999.11.16
US5991708A 1999.11.23
审查员 张 妍

[74] 专利代理机构 北京市柳沈律师事务所
代理人 邸万奎 黄小临

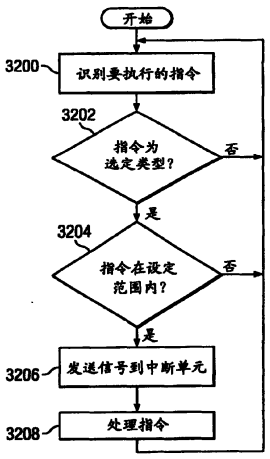
权利要求书 2 页 说明书 30 页 附图 12 页

[54] 发明名称

对特定指令类型的指令执行和数据访问计数的方法和系统

[57] 摘要

一种用于处理指令的方法、设备和计算机指令。响应数据处理系统的处理器内的指令高速缓存中接收到要执行的指令，判定指示符是否与该指令相关联以及该指令是否为指令范围内的特定类型。如果指示符与该指令相关联并且该指令为该指令范围内的特定类型，则产生中断。



1. 一种用于处理指令的数据处理系统中的方法，该方法包括:

响应处理器的指令高速缓存中接收到要执行的指令，判定是否存在一个与该指令相关联的指示符以及该指令是否为一个指令范围内的特定类型，其中该指示符标识该指令为要被监测的指令；以及

如果存在与该指令相关联的所述指示符并且该指令为该指令范围内的特定类型，则产生中断。

2. 如权利要求 1 所述的方法，其中所述产生步骤包括:

从指令高速缓存发送信号到处理器中的中断单元；以及

响应在中断单元接收到该信号，在中断单元中处理中断。

3. 如权利要求 2 所述的方法，其中所述处理步骤包括:

执行与该中断相关联的代码。

4. 如权利要求 3 所述的方法，其中所述代码记录试图访问高速缓存中的指令的功能单元的高速缓存失败。

5. 如权利要求 3 所述的方法，其中所述代码对特定类型的指令已被执行的次数和所述范围已被访问的次数进行计数。

6. 如权利要求 1 所述的方法，其中指示符位于映像存储器中。

7. 如权利要求 1 所述的方法，其中在指令包中接收指令，并且其中指示符包括指令包内的字段中的至少一个空闲位。

8. 如权利要求 1 所述的方法，其中指示符位于指令内的字段中。

9. 如权利要求 1 所述的方法，其中特定类型的指令为转移指令。

10. 如权利要求 1 所述的方法，其中所述指令范围覆盖子例程、库函数或软件模块之一。

11. 一种用于处理指令的数据处理系统，该数据处理系统包括:

判定装置，用于响应处理器的指令高速缓存中接收到要执行的指令，判定是否存在一个与该指令相关联的指示符以及该指令是否为指令范围内的特定类型，其中该指示符标识该指令为要被监测的指令；以及

产生装置，用于在存在与该指令相关联的所述指示符并且该指令为该指令范围内的特定类型的情况下，产生中断。

12. 如权利要求 11 所述的数据处理系统，其中所述产生装置包括:

发送装置，用于从指令高速缓存发送信号到处理器中的中断单元；以及

处理装置，用于响应在中断单元接收到该信号，而在中断单元中处理中断。

13. 如权利要求 12 所述的数据处理系统，其中所述处理装置包括:

执行装置，用于执行与该中断相关联的代码。

14. 如权利要求 13 所述的数据处理系统，其中所述代码记录试图访问高速缓存中的指令的功能单元的高速缓存失败。

15. 如权利要求 13 所述的数据处理系统，其中所述代码对特定类型的指令已被执行的次数和所述范围已被访问的次数进行计数。

16. 如权利要求 11 所述的数据处理系统，其中指示符位于映像存储器中。

对特定指令类型的指令执行和数据访问计数的方法和系统

技术领域

本发明一般涉及一种改进的数据处理系统。具体地说，本发明提供一种用于获得数据处理系统内的性能数据的方法和设备。更具体地说，本发明提供一种用于在获得数据处理系统内的性能数据时对软件工具提供硬件协助的方法和设备。

背景技术

在分析和增强数据处理系统和在该数据处理系统内执行的应用程序的性能时，知道数据处理系统内的哪些软件模块正在使用系统资源是有帮助的。数据处理系统的有效管理和增强需要知道如何和何时使用了各种系统资源。性能工具用来监测和检查数据处理系统以确定各种软件应用程序在数据处理系统内执行时的资源消耗。例如，性能工具可以识别数据处理系统中最频繁执行的模块和指令，或者可以识别分配最大量的存储器或执行最多 I/O 请求的那些模块。硬件性能工具可以内置在系统中，或者在以后的时间点添加。

一种公知的软件性能工具是跟踪工具。跟踪工具可以使用多种技术来提供表示执行程序的执行流的跟踪信息。一种技术通过随着特定事件的出现而对其进行记录来跟踪该特定指令序列，即所谓的基于事件的剖析(profiling)技术。例如，跟踪工具可以记录对模块、子例程、方法、函数或系统组件的每一次进入和每一次退出。或者，跟踪工具可以记录每一个存储器分配请求的请求者以及为其分配的存储器量。典型地，为每一个这样的事件产生带时间戳记录。还使用类似于进入－退出记录的对应记录对来跟踪开始和完成 I/O 或数据传输以及用于很多其它感兴趣事件的任意代码段的执行。

为了改善由不同计算机家族生成的代码的性能，经常有必要确定处理器在执行代码时在何处花费了时间，这样的工作在计算机处理领域内通常称作定位“热点”。在理想情况下，希望以指令和/或源代码行级别查出这样的热点，从而将注意力集中在通过代码改进而可能受益最大的区域。

另一种跟踪技术涉及周期性地对程序的执行流进行采样以识别该程序似乎花费大量时间的特定程序位置。该技术基于以规则间隔周期性地中断应用程序或数据处理系统执行的思想，即所谓的基于样本的剖析。每次中断时，对预定时间长度或者预定次数的感兴趣事件记录信息。例如，在此间隔期间可以记录当前执行线程的程序计数器，其是正被剖析的较大程序的可执行部分。可以在后处理时对照数据处理系统的装入图(load map)和符号表信息来解析这些值，并且可以通过这一分析获得正在何处花费时间的剖析信息(profile)。

创建诸如得到与特定情形或问题有关的答案的工具可能非常费力，并且可能非常难以校准，因为软件工具本身影响测试中的系统。本发明认识到对工具开发和问题分析的硬件协助可以极大地减轻开发软件性能工具所需的工作量。此外，随着处理器的密度增大，可以包括硬件协助来提供附加调试和分析特性。

因此，具有一种用于为用来分析数据处理系统的性能的性能工具提供硬件协助的改进方法、设备和计算机指令将是有利的。

发明内容

本发明提供了一种用于处理指令的方法包含：响应数据处理系统的处理器的指令高速缓存中接收到要执行的指令，判定是否存在一个与该指令相关联的指示符，以及该指令是否为指令范围内的特定类型，其中该指示符标识该指令为要被监测的指令。如果存在与该指令相关联的所述指示符并且该指令为该指令范围内的特定类型，则产生中断。

本发明也提供了一种用于处理指令的数据处理系统，该数据处理系统包括：判定装置，用于响应处理器的指令高速缓存中接收到要执行的指令，判定是否存在一个与该指令相关联的指示符以及该指令是否为指令范围内的特定类型，其中该指示符标识该指令为要被监测的指令；以及产生装置，用于在存在与该指令相关联的所述指示符并且该指令为该指令范围内的特定类型的情况下，产生中断。

附图说明

在所附权利要求中阐述了被认为是本发明特征的新颖特性。然而，通过

参考下面结合附图对示例性实施例的详细描述，本发明本身以及优选使用模式及其进一步目的和优点将会得到更好的理解，其中：

图 1 是可以实现本发明的数据处理系统的方框图；

图 2 是根据本发明优选实施例的用于处理信息的处理器系统的方框图；

图 3 是示出根据本发明优选实施例的用于处理与指示符相关联的指令的组件的图；

图 4 是示出根据优选实施例的一种用于将性能指示符与指令或存储单元(memory location)相关联的机制的图；

图 5 是示出根据本发明优选实施例的指令包(bundle)的图；

图 6A 和 6B 是根据本发明优选实施例的包含性能指示符的子例程的图；

图 7 是根据本发明优选实施例的用于处理包含性能指示符的指令的过程的流程图；

图 8 是根据本发明优选实施例的用于选择性地发送指令到中断单元的过程的流程图；

图 9 是根据本发明优选实施例的用于响应对与性能指示符相关联的存储单元的访问而产生中断的过程的流程图；

图 10 是根据本发明优选实施例的用于对事件进行计数的过程的流程图；

图 11 是根据本发明优选实施例的用于对指令进行选择性计数的过程的流程图；

图 12 是根据本发明优选实施例的用于对指令进行选择性计数的过程的流程图；

图 13 是根据本发明优选实施例的用于识别超过阈值的指令的过程的流程图；

图 14 是根据本发明优选实施例的用于访问存储单元的过程的流程图；

图 15 是示出根据本发明优选实施例的用于生成元数据如性能指示符的组件的方框图；

图 16 是示出根据本发明优选实施例的元数据的图；

图 17 是示出根据本发明优选实施例的装入和维护性能检测映像高速缓存(performance instrumentation shadow cache)时所涉及的组件的图；

图 18 是根据本发明优选实施例的用于生成指令元数据的过程的流程图；

图 19 是根据本发明优选实施例的用于生成存储单元元数据的过程的流

程图;

图 20 是根据本发明优选实施例的用于对特定指令的执行进行计数的过程的流程图;

图 21 是根据本发明优选实施例的用于对特定存储单元的访问进行计数的过程的流程图;

图 22 是示出根据本发明优选实施例的用于访问关于对指令的执行或对存储单元的访问所收集的信息的组件的图;

图 23 是根据本发明优选实施例的用于自主修改程序中的代码以允许对部分代码进行选择性计数或剖析的组件的方框图;

图 24 是根据本发明优选实施例的用于将性能指示符动态添加到指令中或者使其与指令相关联的过程的流程图;

图 25 是示出根据本发明优选实施例的用来通过将性能指示符与页内的指令相关联来扫描页的组件的图;

图 26 是根据本发明优选实施例的用于将指示符关联到页内的指令的过程的流程图;

图 27 是示出根据本发明优选实施例的包含堆栈帧的调用堆栈的图;

图 28 是根据本发明优选实施例的用于识别与调用和返回从性能监测器单元收集数据的指令相关联的事件的过程的流程图;

图 29 是根据本发明优选实施例的用于识别已被执行多于选定次数的指令的过程的流程图;

图 30 是根据本发明优选实施例的用于在特定指令被执行多于某选定次数时检查调用堆栈并识别例程调用者的过程的流程图;

图 31 是示出根据本发明优选实施例的为进行监视而选择的指令和数据范围的图; 以及

图 32 是根据本发明优选实施例的用于对设定范围的访问次数以及在设定范围内执行的指令数进行计数的过程的流程图。

具体实施方式

现在参照图 1, 其中示出了可以实现本发明的数据处理系统的方框图。客户机 100 是计算机的例子, 实现本发明的过程的代码或指令可以位于其中。客户机 100 采用外围组件互连(PCI)局部总线架构。虽然所示例子采用 PCI 总

线，但是也可以使用其它总线架构如加速图形端口(AGP)和工业标准架构(ISA)。处理器 102 和主存储器 104 通过 PCI 桥 108 连接到 PCI 局部总线 106。PCI 桥 108 还可以包括用于处理器 102 的集成存储器控制器和高速缓冲存储器。与 PCI 局部总线 106 的其它连接可以通过直接组件互连或者通过内插板来实现。在所示例子中，局域网(LAN)适配器 110、小型计算机系统接口 SCSI 主机总线适配器 112 和扩展总线接口 114 通过直接组件连接而连接到 PCI 局部总线 106。与此相反，音频适配器 116、图形适配器 118 和音频/视频适配器 119 通过插入到扩展槽中的内插板而连接到 PCI 局部总线 106。扩展总线接口 114 为键盘和鼠标适配器 120、调制解调器 122 和附加存储器 124 提供连接。SCSI 主机总线适配器 112 为硬盘驱动器 126、磁带驱动器 128 和 CD-ROM 驱动器 130 提供连接。典型的 PCI 局部总线实现将支持三个或四个 PCI 扩展槽或内插连接器。

操作系统运行在处理器 102 上，并且用来协调和提供对图 1 的数据处理系统 100 内的各个组件的控制。操作系统可以是市场上可买到的操作系统，如可从微软公司获得的 Windows XP。面向对象的编程系统如 Java 可以结合操作系统运行，并且提供从在客户机 100 上执行的 Java 程序或应用程序对操作系统的调用。“Java” 是太阳微系统公司(Sun Microsystems, Inc.)的商标。操作系统、面向对象的编程系统、以及应用程序或程序的指令位于诸如硬盘驱动器 126 的存储装置上，并且可以装入到主存储器 104 中以由处理器 102 执行。

本领域的普通技术人员应当理解，图 1 中的硬件可以根据具体实现而不同。作为对图 1 所示的硬件的补充或替代，可以使用其它内部硬件或外围装置，如快闪只读存储器(ROM)、等效非易失性存储器或光盘驱动器等。另外，本发明的过程也可以应用于多处理器数据处理系统。

例如，客户机 100，如果可选地配置为网络计算机，则可以不包括 SCSI 主机总线适配器 112、硬盘驱动器 126、磁带驱动器 128 和 CD-ROM 130。在这种情况下，该计算机可以适当地称作客户计算机，并包括某种网络通信接口如 LAN 适配器 110、调制解调器 122 等。作为另一个例子，客户机 100 可以是配置成在不依赖于某种网络通信接口的情况下可启动的独立系统，而不管客户机 100 是否包括某种网络通信接口。作为另外一个例子，客户机 100 可以是个人数字助理(PDA)，其配置有 ROM 和/或快闪 ROM 来提供用于存储

操作系统文件和/或用户所产生数据的非易失性存储器。图 1 所示的例子和上述例子不意味着隐含架构限制。

本发明的过程使用可以位于例如主存储器 104、存储器 124 或一个或多个外围装置 126-130 的存储器中的计算机实现指令由处理器 102 执行。

接下来参照图 2，其示出根据本发明优选实施例的用于处理信息的处理器系统的方框图。处理器 210 可以作为图 1 中的处理器 102 实现。

在优选实施例中，处理器 210 是单个集成电路超标量微处理器。从而，如下面进一步所述，处理器 210 包括各种单元、寄存器、缓冲器、存储器和其它部分，所有这些都由集成电路形成。另外，在优选实施例中，处理器 210 根据简化指令集计算机(“RISC”)技术运行。如图 2 所示，系统总线 211 连接到处理器 210 的总线接口单元(“BIU”)212。BIU 212 控制处理器 210 与系统总线 211 之间的信息传输。

BIU 212 连接到处理器 210 的指令高速缓存 214 和数据高速缓存 216。指令高速缓存 214 将指令输出到定序器单元 218。响应来自指令高速缓存 214 的这些指令，定序器单元 218 将指令选择性地输出到处理器 210 的其它执行电路。

除了定序器单元 218 之外，在本优选实施例中，处理器 210 的执行电路还包括多个执行单元，即转移(branch)单元 220、定点单元 A(“FXUA”)222、定点单元 B(“FXUB”)224、复合定点单元(“CFXU”)226、装入/存储单元(“LSU”)228 以及浮点单元(“FPU”)230。FXUA 222、FXUB 224、CFXU 226 和 LSU 228 从多个通用架构寄存器(“GPR”)232 和多个定点重命名(rename)缓冲器 234 输入其源操作数信息。而且，FXUA 222 和 FXUB 224 从进位(“CA”)寄存器 239 输入“进位”。FXUA 222、FXUB 224、CFXU 226 和 LSU 228 输出其运算结果(目的操作数信息)以存储在定点重命名缓冲器 234 内的选定条目(entry)上。另外，CFXU 226 从专用寄存器处理单元(“SPR 单元”)237 输入并向其输出源操作数信息和目的操作数信息。

FPU 230 从多个浮点架构寄存器(“FPR”)236 和多个浮点重命名缓冲器 238 输入其源操作数信息。FPU 230 输出其运算结果(目的操作数信息)以存储在浮点重命名缓冲器 238 内的选定条目上。

响应装入指令，LSU 228 从数据高速缓存 216 输入信息，并且将该信息拷贝到重命名缓冲器 234 和 238 中选定的一个。如果该信息未存储在数据高

速缓存 216 中，则数据高速缓存 216(通过 BIU 212 和系统总线 211)从连接到系统总线 211 的系统存储器 239 输入该信息。而且，数据高速缓存 216 能够(通过 BIU 212 和系统总线 211)将信息从数据高速缓存 216 输出到连接到系统总线 211 的系统存储器 239。响应存储指令，LSU 228 从 GPR 232 和 FPR 236 中选定的一个输入信息，并且将该信息拷贝到数据高速缓存 216。

定序器单元 218 从 GPR 232 和 FPR 236 输入信息，并且向其输出信息。转移单元 220 从定序器单元 218 输入指令和表示处理器 210 的当前状态的信号。响应该指令和信号，转移单元 220(向定序器单元 218)输出表示存储要由处理器 210 执行的指令序列的合适的存储器地址的信号。响应来自转移单元 220 的该信号，定序器单元 218 从指令高速缓存 214 输入所指示的指令序列。如果一条或多条指令序列没有存储在指令高速缓存 214 中，则指令高速缓存 214(通过 BIU 212 和系统总线 211)从连接到系统总线 211 的系统存储器 239 输入这些指令。

响应从指令高速缓存 214 输入的指令，定序器单元 218 选择性地将指令调度到执行单元 220、222、224、226、228 和 230 中选定的一个。每个执行单元执行一条或多条特定指令类的指令。例如，FXUA 222 和 FXUB 224 对源操作数执行第一类定点数学运算，如加法、减法、与运算、或运算以及异或运算。CFXU 226 对源操作数执行第二类定点运算，如定点乘法和除法。FPU 230 对源操作数执行浮点运算，如浮点乘法和除法。

当信息存储在选定的一个重命名缓冲器 234 上时，该信息与由为其分配了选定重命名缓冲器的指令指定的存储位置(例如，GPR 232 或进位(CA)寄存器 242 之一)相关联。响应来自定序器单元 218 的信号，将选定的一个重命名缓冲器 234 上存储的信息拷贝到与其关联的一个 GPR 232(或 CA 寄存器 242)中。定序器单元 218 响应“完成”产生了该信息的指令，引导拷贝在选定的一个重命名缓冲器 234 上存储的信息。该拷贝称作“写回”

当信息存储在选定的一个重命名缓冲器 238 上时，该信息与一个 FPR 236 相关联。响应来自定序器单元 218 的信号，将选定的一个重命名缓冲器 238 上存储的信息拷贝到与其关联的一个 FPR 236 中。定序器单元 218 响应“完成”产生该信息的指令，引导拷贝在选定的一个重命名缓冲器 238 上存储的信息。

处理器 210 通过在执行单元 220、222、224、226、228 和 230 中的各个

单元上同时处理多条指令来实现高性能。因此，每条指令作为一系列级处理，其中每个级与其它指令的级可并行执行。该技术称作“流水线技术”。在该示例性实施例的一个重要方面，指令通常以六个级处理，即提取、译码、调度、执行、完成和写回。

在提取级，定序器单元 218 选择性地(从指令高速缓存 214)从上面有关转移单元 220 和定序器单元 218 而进一步讨论的存储指令序列的一个或多个存储器地址输入一条或多条指令。

在译码级，定序器单元 218 对多达四条所提取的指令进行译码。

在调度级，定序器单元 218 在为所调度指令结果(目的操作数信息)保留重命名缓冲器条目之后，将多达四条已译码的指令选择性地调度到执行单元 220、222、224、226、228 和 230 中(响应译码级中的译码)选定的一个。在调度级，将操作数信息提供给所调度指令的选定执行单元。处理器 210 以指令编程序列的次序调度指令。

在执行级，如上所述，执行单元执行其调度指令，并且输出其操作结果(目的操作数信息)以存储在重命名缓冲器 234 和重命名缓冲器 238 内的选定条目上。以这种方式，处理器 210 能够相对于指令编程序列无序地执行指令。

在完成级，定序器单元 218 指示指令“完成”。处理器 210 以指令编程序列的次序“完成”指令。

在写回级，定序器 218 引导将信息分别从重命名缓冲器 234 和 238 拷贝到 GPR 232 和 FPR 236。定序器单元 218 引导拷贝选定重命名缓冲器上存储的信息。同样，在特定指令的写回级，处理器 210 响应该特定指令更新其架构状态。处理器 210 以指令编程序列的次序处理指令的各“写回”级。处理器 210 在特定情形下有利地合并指令的完成级和写回级。

在该示例性实施例中，每条指令需要一个机器周期来完成指令处理的每一个级。但是，一些指令(例如由 CFXU 226 执行的复合定点指令)可能需要多个周期。因此，响应完成先前指令所需的时间的变化，在特定指令的执行和完成级之间可能发生可变延迟。

完成缓冲器 248 安设在定序器 218 内以跟踪正在执行单元内执行的多条指令的完成。一旦出现以应用程序指定顺序成功地完成了一条指令或一组指令的指示时，可以利用完成缓冲器 248 来发起将这些已完成指令的结果传输到关联通用寄存器。

另外，处理器 210 还包括连接到指令高速缓存 214 以及处理器 210 中其它单元的性能监测器单元 240。可以利用性能监测器单元 240 来监测处理器 210 的操作，在本示例性实施例中，性能监测器单元 240 是能够提供描述指令执行资源利用和存储控制的详细信息的软件可访问机制。虽然图 2 中未示出，性能监测器单元 240 耦接到处理器 210 的每个功能单元以允许监测处理器 210 的所有方面的操作，包括例如重建事件之间的关系、识别虚假触发、识别性能瓶颈、监测流水线停顿(pipeline stall)、监测空闲处理器周期、确定调度效率、确定转移效率、确定未对准(misaligned)数据访问的性能损失、识别串行化指令的执行频率、识别被禁止中断以及确定性能效率。感兴趣事件还可以包括例如指令译码时间、指令执行、转移事件、高速缓存失败(miss)和高速缓存成功(hit)。

性能监测器单元 240 包括其数目依赖于具体实现的(例如，2-8 个)计数器 241-242，标记为 PMC1 和 PMC2，用来对选定事件的出现进行计数。性能监测器单元 240 还包括至少一个监测器模式控制寄存器(MMCR)。在本例中，有两个指定计数器 241-242 功能的控制寄存器 MMCR 243 和 244。计数器 241-242 和 MMCR 243-244 最好作为通过可由 CFXU 226 执行的 MFSPR(从 SPR 传送)和 MTSPR(传送至 SPR)指令可访问以进行读或写的 SPR 实现。然而，在一个可替换实施例中，计数器 241-242 和 MMCR 243-244 可以简单地作为 I/O 空间中的地址实现。在另一个替换实施例中，控制寄存器和计数器可以通过变址寄存器来间接访问。该实施例在来自英特尔公司(Intel Corporation)的处理器内的 IA-64 架构中实现。

另外，处理器 210 还包括连接到指令高速缓存 214 的中断单元 250。另外，虽然图 2 中未示出，中断单元 250 连接到处理器 210 内的其它功能单元。中断单元 250 可以从其它功能单元接收信号，并且发起诸如开始错误处理或捕获过程的操作。在这些例子中，使用中断单元 250 来产生在程序执行期间可能发生的中断和异常。

本发明提供了在程序执行期间监视对特定指令的执行以及对特定存储单元的访问的能力。具体地说，可以使用空闲字段来保存用于标识指令或存储单元为要由性能监测器单元或处理器中的某个其它单元监测的指令或存储单元的指示符。或者，指示符可以存储在与指令或存储单元相关联的另一个位置中。在指示符置于指令中的情况下，典型地使用空闲字段，但是在一些情

况下可以扩展指令以包括指示符所需的空间。在这种情况下，处理器的架构可能需要改变。例如，64 位架构可以改成 65 位架构以容纳指示符。对于数据访问，指示符可以与数据或该数据所在的存储单元相关联。

现在参照图 3，其示出根据本发明优选实施例的用于处理与指示符相关联的指令的组件的图。指令高速缓存 300 接收指令包(bundle)302。指令高速缓存 300 是图 2 中的指令高速缓存 214 的示例。指令包是一种指令编组。这种指令编组典型地出现于可从英特尔公司获得的 IA-64 处理器中。指令高速缓存 300 处理所要执行的指令。

作为该指令处理的一部分，指令高速缓存 300 确定哪些指令与指示符相关联。在这些例子中，这些指示符也称作“性能指示符”。信号 304 已与性能指示符相关联。结果，将指令的信号 304 发送到性能监测器单元 306。性能监测器单元 306 是图 2 中的性能监测器单元 240 的示例。

当指令高速缓存 300 确定存在与指示符相关联的指令时，发送信号以表示正在执行被标记(marked)的指令。在这些例子中，被标记的指令是与性能指示符相关联的指令。或者，性能指示符可以指示指令包中的所有项目或指令均被标记以被计数。另外，这些指令的信号由指令高速缓存 300 发送到适当的功能单元。根据具体实现，不同于性能监测器单元 306 的功能单元可以对指令执行进行计数。在性能指示符位于指令或指令包中的情况下，高速缓存单元即指令高速缓存 300 探测指示符，并且向性能监测器单元 306 发送信号。

当性能监测器单元 306 接收到这些指令的信号时，性能监测器单元 306 对与指令 304 执行相关联的事件进行计数。如图所示，性能监测器单元 306 编程为仅对与性能指示符相关联的指令的事件进行计数。换句话说，使用与指令或存储单元相关联的指示符来使得能够由性能监测器单元 306 对与指令或存储单元相关联的事件进行计数。如果指令高速缓存 300 接收到没有性能指示符的指令，则不对与该指令相关联的事件进行计数。总而言之，性能指示符使得能够在处理器中逐个指令地或逐个存储单元地进行计数。

如果性能监测器单元 306 设成对这些类型的被标记指令所允许的规格(metrics)进行计数的模式，则性能监测器单元 306 对与性能指示符相关联的指令的事件进行计数。在某些情况下，性能监测器单元 306 可以设成执行作为当前可用功能的某种其它类型的计数，例如对所有指令的执行进行计数。

对于访问存储单元中的数据，由数据高速缓存如图 2 中的数据高速缓存

216 而不是由指令高速缓存来处理数据和指示符。数据高速缓存将表示正在访问被标记存储单元的信号发送到性能监测器单元 306。被标记存储单元类似于被标记指令。这些类型的存储单元是与性能指示符相关联的存储单元。

现在参照图 4，其示出根据本发明优选实施例的一种用于将性能指示符与指令或存储单元相关联的机制的图。处理器 400 从高速缓存 402 接收指令。在本例中，指示符不与指令一起存储，也不存储于找到数据的存储单元中。相反，指示符存储在单独的存储区域，即性能检测映像高速缓存(shadow cache)404 内。该存储装置可以是任何存储装置，例如系统存储器、闪存、高速缓存或盘。

当处理器 400 从高速缓存 402 接收指令时，处理器 400 检查性能检测映像高速缓存 404 以查看性能指示符是否与指令相关联。对包含数据的存储单元的访问进行类似的检查。在一个实施例中，为每个不影响实际数据段的对应的字提供完全映像字。换句话说，处理器 400 允许高速缓存 402 的架构或配置保持不变。在这些例子中，所述映射是逐字的。然而，也可以使用某种其它类型的映射，例如每数据字一个映像位，其中性能检测映像高速缓存 404 中的一位对应于数据的一个字。

对于这种类型的架构，利用该特性，编译器以类似于调试符号的方式在与数据区本身分离的工作区内创建调试信息。当装入模块时，由装入器准备额外的信息即性能指示符，从而当将指令装入到高速缓存 402 中时使其可用于并入性能检测映像高速缓存 404 中。这些高速缓存区域可以是混合的并且要么如此标记要么通过操作模式理解。处理器 400 使用性能指示符来确定如何对相关数据访问和指令执行进行计数，或者如何避免(take exception)相关数据访问和指令执行。在这些例子中，通过调试器或性能分析程序将该过程编程为了解在执行指令时是否使用映像信息。

现在参照图 5，其示出根据本发明优选实施例的指令包的图。指令包 500 包含指令槽 502、指令槽 504、指令槽 506 和模板 508。如图所示，指令包 500 包含 128 位。每个指令槽包含 41 位，而模板 508 包含 5 位。模板 508 用来标识当前指令包内的终止，并且将槽内的指令映射到不同类型的执行单元中。

指令包 500 内的空闲位用来保存本发明的指示符。例如，指示符 510、512 和 514 分别位于指令槽 502、504 和 506 内。这些指示符可以根据具体实现而采取各种形式和各种大小。指示符可以使用单个位或者可以使用多个位。

单个位可以用来指示响应该指令的执行要对事件进行计数。多个位可以用来标识阈值，例如可以在对事件进行计数之前传递的指令执行的处理器或时钟周期的数量。此外，这些位甚至可以用作针对特定指令的计数器。类似的字段使用可以用于标记数据或存储单元的指示符。

或者，模板 508 可以用来包含相关指示符的指令包，从而使用一位来标识指令包中的所有指令。另外，指令包本身可以扩展成 256 位或某一其它位数，以包含性能指示符的额外信息。

接下来参照图 6A 和 6B，示出根据本发明优选实施例的包含性能指示符的子例程和包含性能指示符的数据的图。在本例中，图 6A 中的子例程 600 包括多条指令，其中指令 602、604 和 606 与性能指示符相关联。这些指令也称作被标记指令。当执行这些指令时，对与这些指令相关联的事件进行计数，从而为软件工具获得数据以分析执行子例程 600 的数据处理系统的性能。

数据或包含数据的存储单元可以采用类似方式以指示符标记。在这些例子中，这些指示符用于对数据或存储单元访问进行计数。在图 6B 中，数据 610 包括与性能指示符相关联的数据。数据 612 和数据 614 是与性能指示符相关联的数据 610 的部分。与性能指示符相关联的这些数据部分也称作被标记数据。

现在参照图 7，其示出了根据本发明优选实施例的用于处理包含性能指示符的指令的过程的流程图。图 7 所示的过程可以在指令高速缓存如图 2 中的指令高速缓存 214 中实现。

该过程以接收指令包(步骤 700)开始。在这些例子中，每个指令包具有类似于图 5 中的指令包 500 的格式。识别指令包中的指令(步骤 702)。判定是否存在与指令相关联的性能指示符(步骤 704)。该判定可以通过检查指令或指令包中的适当字段来进行。或者，可以检查性能检测映像高速缓存如图 4 中的性能检测映像高速缓存 404，以查看性能指示符是否与指令相关联。

如果存在性能指示符，则发送信号到性能监测器单元(步骤 706)。当接收到该信号时，性能监测器单元将对与指令执行相关联的事件进行计数。另外，对指令进行处理(步骤 708)。指令处理包括例如将指令发送到适当的功能单元以便执行。

然后，判定在指令包中是否存在另外的未处理指令(步骤 710)。如果在指令包中存在另外的未处理指令，则该过程返回到如上所述的步骤 702。否则，

该过程终止。回到步骤 704，如果不存在性能指示符，则该过程直接进入步骤 708。

现在参照图 8，其示出了根据本发明优选实施例的用于选择性地发送信号到中断单元的过程的流程图。图 8 所示的过程可以在指令高速缓存如图 2 的指令高速缓存 242 中实现。在使用性能监测器单元监测事件可能错过特定事件的情况下采用该过程。例如，性能监测器单元对事件进行计数。当发生高速缓存失败时，发送信号到性能监测器单元。当将对应高速缓存线的元数据装入到高速缓存中时，也引发(raise)一个或多个适当信号。如果元数据表示要引发异常，则发送信号到中断单元，其中该信号表示要引发异常。

该过程以接收指令包(步骤 800)开始。识别指令包中的指令(步骤 802)。判定是否存在与该指令相关联的性能指示符(步骤 804)。发送到中断单元以表示要引发异常的信号不同于发送到性能监测器单元的信号。例如，指令可以与具有导致发送信号到中断单元的第一值的特定性能指示符相关联。性能指示符的第二值可以用来发送不同信号到性能监测器单元。如果存在具有第一值的性能指示符，则发送信号到中断单元(步骤 806)。当接收到该信号时，中断单元发起适当的调用流支持以处理该中断。调用流支持可以例如记录试图访问高速缓存中的指令或数据的功能单元可能发生的高速缓存失败。

另外，对指令进行处理(步骤 808)。指令的处理包括例如发送指令到适当的功能单元以便执行。

然后，判定指令包中是否存在另外的未处理指令(步骤 810)。如果指令包中存在另外的未处理指令，则该过程返回到如上所述的步骤 802。否则，该过程终止。回到步骤 804，如果不存在性能指示符，则该过程直接进入步骤 808。

现在参照图 9，其示出根据本发明优选实施例的用于响应对与性能指示符相关联的存储单元的访问而产生中断的过程的流程图。图 9 所示的过程可以在数据高速缓存如图 2 的数据高速缓存 246 中实现。

该过程以识别访问存储单元的请求(步骤 900)开始。响应于识别出该请求，判定性能指示符是否与存储单元相关联(步骤 902)。如果性能指示符与存储单元相关联，则通过发送信号到中断单元来产生中断(步骤 904)。然后，处理对存储单元的访问(步骤 906)，然后该过程终止。

在图 10 中，示出了根据本发明优选实施例的用于对事件进行计数的过程

的流程图。图 10 所示的过程可以在性能监测器单元如图 2 的性能监测器单元 240 中实现。

该过程以从指令高速缓存接收表示正在处理带有性能指示符的指令的信号(步骤 1000)开始。下一步，对与正被处理的指令相关联的事件进行计数(步骤 1002)，然后该过程终止。事件计数可以存储在计数器如图 2 的计数器 241 中。

接下来参照图 11，其示出了根据本发明优选实施例的用于对指令进行选择性计数的过程的流程图。图 11 所示的过程可以在指令高速缓存如图 2 的指令高速缓存 214 中实现。

该过程以判定是否接收到与性能指示符相关联的指令(步骤 1100)开始。在本例中，该指示符导致对由处理器执行的这一指令和所有后续指令的事件进行计数。或者，该指示符可以是指示要开始新计数模式的指令本身。如果接收到带有指示符的指令，则设置(set)标志以开始对指令的事件进行计数(步骤 1102)。该标志表示应开始对指令的事件进行计数。

下一步，判定是否接收到带有指示符的指令(步骤 1104)。或者，指示符可以是指示要停止新计数模式的指令本身。如果接收到带有指示符的指令，则清除(unset)该标志以停止对事件的计数(步骤 1106)，然后该过程终止。

步骤 1100 和步骤 1104 中的指示符可以是相同的指示符，其中该指示符切换标志的设置和清除。在另一种实现中，可以使用两个不同的指示符，其中第一指示符仅设置标志。第二指示符用来清除标志。可以通过在要进行计数时采用高信号而在不再启动计数时采用低信号来简单地实现高速缓存单元如指令高速缓存或数据高速缓存与性能监测器单元之间为表示计数模式而进行的通信。

接下来参照图 12，其示出了根据本发明优选实施例的用于对指令进行选择性计数的过程的流程图。图 12 所示的过程可以在指令高速缓存如图 2 的指令高速缓存 214 中实现。

该过程以检查标志(步骤 1200)开始。判定是否设置了标志(步骤 1202)。如果设置了标志，则发送信号到性能监测器单元以启动该单元对事件进行计数(步骤 1204)，然后该过程终止。否则，发送信号到性能监测器单元以禁止对事件计数(步骤 1206)，然后该过程终止。

图 11 和 12 所示的过程在指令与性能指示符相关联之后对所有指令的事

件进行计数。通过这种方式，可以使用较少的位来触发对事件的计数。此外，在对所有指令计数的情况下，可以对与外部子例程调用相关联的事件进行计数。

现在参照图 13，其示出根据本发明优选实施例的用于识别超过阈值的指令的过程的流程图。图 13 所示的过程可以在指令高速缓存如图 2 的指令高速缓存 214 中实现。

该过程以接收与性能指示符相关联的指令(步骤 1300)开始。为指令识别阈值(步骤 1302)。在这些例子中，阈值与完成指令所需的处理器或时钟周期的数量相关。如果访问高速缓存所需的高速缓存延迟或时间量超过该阈值，则对该事件进行计数。在这些例子中，阈值设置在指示符内。

例如，可以使用三位来设置八个不同的阈值。例如，“xx1” = 10 周期，“x1x” = 50 周期，以及“1xx” = 100 周期。这三位的某组合可以用来设置阈值。根据具体实现，可以使用更多或更少的位，并且可以将不同值分配给这些位。这些位的含义也可以通过接口来控制，例如可以用来设置每个位的含义的一组寄存器。这些寄存器是为此特定目的而添加到处理器架构的寄存器。

监测用于执行该指令的周期(步骤 1304)。判定对该指令是否超过了阈值(步骤 1306)。如果超过了阈值，则执行选定操作(步骤 1308)。该选定操作可以根据具体实现而采取不同的形式。例如，每次超过阈值时，可以递增计数器。或者，可以产生中断。中断可以将控制传递给另一个过程以收集数据。例如，该数据可以包括调用堆栈和有关该调用堆栈的信息。堆栈是保留存储器区域，其中一个或多个程序存储状态数据，如过程和函数调用地址、所传递的参数、性能监测器计数器值以及有时还有局部变量。

判定监测是否结束(步骤 1310)。步骤 1310 可以一次一条指令地实现。当执行了指令或者超过了阈值时，发送信号。在本例中，单条指令的执行导致发送一个信号。在可以同时执行多条指令的情况下，可能需要多个信号来表示每条指令的执行。在一些实施例中，可以支持采样方案，其中一次仅对一条指令支持阈值。这可以通过仅支持处理器指令队列中的特定位置中的那些指令的阈值来实现。在其它实施例中，如果至少一条被标记指令超过阈值，则可以发送一个信号。对于超过阈值的每条指令，为该指令引发或产生单独的信号。

如果监测结束，则将所收集的信息发送到监测程序(步骤 1312)，然后，该过程终止。否则，该过程返回到如上所述的步骤 1304。在步骤 1306，如果未超过该指令的阈值，则该过程直接进入步骤 1310。

可以在数据高速缓存如图 2 的数据高速缓存 216 中实现类似的过程，以监测对存储单元的访问。图 13 所示的过程可以修改成识别访问存储单元中的数据所需的周期。如同指令执行一样，当访问存储单元中的数据所需的时间量超过指定阈值时进行计数或产生中断。

如同其它例子一样，可以作为指令的一部分或者与存储单元中的数据一起包括这些指示符。或者，这些指示符可以与指令或数据相关联地在性能检测映像高速缓存或存储器中找到。

参照图 14，其示出根据本发明优选实施例的用于监测对存储单元的访问的过程的流程图。图 14 所示的过程可以在数据高速缓存如图 2 的数据高速缓存 216 中实现。该过程用来对存储单元中的数据访问进行计数。

该过程以接收与性能指示符相关联的数据(步骤 1400)开始。判定是否访问了该数据的存储单元(步骤 1402)。如果访问了该存储单元，则递增计数器(步骤 1404)。判定监测是否结束(步骤 1406)。如果对存储单元的监测结束，则该过程终止。否则，该过程返回到步骤 1402。在步骤 1402，如果没有访问存储单元，则该过程进入步骤 1406。

参照图 15，其示出根据本发明优选实施例的用于生成元数据如性能指示符的组件的方框图。编译器支持嵌入在指示要生成的元数据的源代码中的命令(directive)。编译器 1500 可以生成用于执行的指令 1502 和用于监测的元数据。在这些例子中，随着指令或数据高速缓存页被装入到存储器中，操作系统程序装入器/链接器和/或性能监测程序读取由编译器 1500 生成的元数据，并且将元数据装入到存储器如性能监测器部分 1506 中。该部分本身被标记为元数据 1504。处理器可以接受性能监测器部分 1506 中具有编译器所生成部分数据的格式的元数据 1504，并且向处理器的内部性能检测映像高速缓存填充该数据。下面参照图 17 描述面向块的方案。

在一个实施例中，该格式对于其块或扇区(sector)引用中的每一个都简单地具有性能检测映像高速缓存条目，并且将元数据 1504 传送至其对应的一个或多个映像条目。代替具有性能检测映像高速缓存，可以修改高速缓存本身的内部格式来包含元数据 1504。在修改指令流本身以包含元数据的实施例中，

装入器更新指令流以包含适当的指示符和工作区，或者编译器 1500 生成了代码来包含元数据 1504。在任何情况下，在装入了代码之后，处理器接收元数据 1504。

另外，元数据 1504 可以与指令 1502 相关联地置于性能检测映像存储器 1505 中。编译器 1500 在表或调试数据部分中产生信息。性能监测程序将该信息装入到性能检测映像存储器 1505 内的映像数据区中。或者，调试区可以由一起工作的操作系统和处理器自动填充。

然后，可以由处理器 1508 执行指令 1502。编译器 1500 可以设置处理器 1508 中的寄存器如模式寄存器 1510。当设置了该寄存器时，处理器 1508 在执行指令 1502 时查看性能检测映像存储器 1505 中的元数据 1504，以判定元数据 1504 中的性能指示符是否与指令 1502 中正被执行的指令相关联。使用例如上面参照图 2-14 所述的过程处理这些性能指示符。如果没有设置模式寄存器 1510，则在执行指令 1502 时忽略元数据 1504。

可以对存储单元 1512 中的数据执行类似的过程。根据具体实现，元数据 1504 可以置于指令内或数据内，而不是置于性能检测映像存储器 1505 中。然而，通过将元数据 1504 置于性能检测映像存储器 1505 中，当元数据 1504 置于性能检测映像存储器 1505 中时可以动态执行元数据 1504 的生成。

该特性允许在不必修改程序的情况下进行对指令的选择和监测。换句话说，编译器 1500 可以在编译了指令 1502 以便由处理器 1508 执行之后生成元数据 1504。设置模式寄存器 1510 使处理器 1508 在性能检测映像存储器 1505 中查找元数据 1504 而不必修改指令 1502。在这些例子中，元数据 1504 采取告诉处理器 1508 如何处理指令 1502 的执行和/或对存储单元 1512 的数据访问的性能指示符的形式。

接下来参照图 16，其示出根据本发明优选实施例的元数据的图。元数据 1600 是图 15 中的元数据 1504 的例子。该元数据由编译器如编译器 1500 生成。

在本例中，元数据 1600 包括 5 个条目，即条目 1602、1604、1606、1608 和 1610，如元数据 1600 中的行 1612 所示。在本例中，这些条目中的每一个都包括偏移、长度和用于描述代码的检测(instrumentation)的标志。

条目 1602 的偏移为 0，而其条目长度为 120 字节。标志 1614 表示需要对由条目长度 1616 表示的范围内的所有指令进行计数。在这些例子中，每条

指令的长度为 4 字节。条目 1604 的条目长度为 4 字节，这与指令相对应。标志 1618 表示当执行该指令时应当产生异常。

在条目 1606 中，以 160 字节的偏移开始的指令与标志 1620 相关联。该标志表示如果超过阈值即 100 个周期则应当对指令进行计数。

条目 1608 中的标志 1622 表示应当在偏移为 256 字节的指令处开始跟踪。如条目 1610 中的标志 1624 所指示而停止跟踪，其中条目 1610 具有用于偏移为 512 字节的指令的标志。

这些标志用来生成与这些指令相关联的性能指示符。操作系统将由编译器生成的该元数据传送到性能检测映像存储器如图 15 的性能检测映像存储器 1506 中，并且处理该元数据。或者，根据具体实现，该元数据可以置于指令内的字段中。

现在参照图 17，示出根据本发明优选实施例的在装入和维护性能检测映像高速缓存时所涉及的组件的图。在本例中，现有高速缓存 1700 包含主段 1702。主段 1702 包括块 1704、1706、1708、1710、1712、1714、1716、1718、1720、1722 和 1724。转换表 1726 用来提供对主段 1702 中的块 1704-1724 到性能检测(perfinst)段 1728 中的块的映射。该段中的数据置于新性能检测映像高速缓存 1730 中。

在程序编译的时候，编译器生成如前所述的新性能检测数据部分。在程序装入时候，装入器向处理器查询以确定高速缓存线大小。装入器以处理器所要求的格式，为装入器所装入的任何文本或数据段解析性能检测段 1728 并且构造映像段。该映像段置于新性能检测映像高速缓存 1730 中。

映像段中的每一块包含对应主高速缓存块中的指令或数据的元数据。该元数据例如包括主段 1702 的块内每个带标签(tagged)项目的标志、标签字段、阈值和计数字段。该元数据还可以包括表示块中的所有指令或数据的标志。

装入器构造将主段 1702 中的每一块映射到性能检测段 1728 中对应的性能检测块如块 1732、1734、1736、1738、1740、1742、1744、1746、1748、1750 和 1752 的表，即转换表 1726。此外，装入器还向处理器登记该表即转换表 1726 的头以及主段 1702 的位置和大小。

在页更替的时候，页面调度软件提供新接口来使性能检测段 1728 与对应主段即主段 1702 相关联。当主段 1702 页面调入或调出时，性能检测段 1728 也页面调入或调出。

在高速缓存线更替的时候，处理器包含新性能检测映像高速缓存 1730，其中的高速缓存帧与现有数据和指令高速缓存如现有高速缓存 1700 中的帧直接相关联。当处理器的指令或数据高速缓存装入新线时，高速缓存也必须将对应性能检测块装入到性能检测映像高速缓存即新性能检测映像高速缓存 1730 中。处理器(从程序装入的时候由装入器提供的登记数据)知道处理器正在将块带入其具有关联性能检测段即性能检测段 1728 的高速缓存。处理器在与该段相关联的转换表 1726 中查看，得到对与将要装入的块相对应的性能检测块的引用，并且将该性能检测块装入到新性能检测映像高速缓存 1730 中。在这些例子中，与元数据相关联的高速缓存失败不被用信号通知，或者以不同于与主高速缓存块如主段 1702 中的数据相关联的高速缓存失败的方式进行处理。

现在参照图 18，其示出根据本发明优选实施例的用于生成指令的元数据的过程的流程图。图 18 所示的过程可以由性能监测程序实现。

该过程以识别要剖析的指令(步骤 1800)开始。该指令可以是例如已被执行多于选定次数的指令。为所识别的指令生成元数据(步骤 1802)。该元数据采取性能指示符的形式。性能指示符可以，例如，每当执行该指令时递增计数器，在执行指令所需的周期数超过阈值的情况下递增计数器，在该指令之后对所有事件、所有指令触发对事件的计数，或者对响应执行指令而发生的事件进行计数。在优选实施例中，计数器位于关联的性能检测映像高速缓存中，并且采取若干位来允许高速缓存中的数据或指令与被保留用于计数的位之间的一一对应关系。

然后，将元数据与指令相关联(步骤 1804)。接下来，判定是否存在更多指令要处理(步骤 1806)。如果存在另外的指令，则该过程返回到步骤 1800。否则，该过程终止。可以使用类似的过程来动态生成存储单元中的数据的元数据。

现在参照图 19，其示出根据本发明优选实施例的用于生成存储单元的元数据的过程的流程图。图 19 所示的过程可以在编译器如图 15 的编译器 1500 中实现。

该过程以识别要剖析的存储单元(步骤 1900)开始。通过探测对被标记位置的访问而发生步骤 1900。为所识别的存储单元生成元数据(步骤 1902)。该元数据采取性能指示符的形式。性能指示符可以，例如，每当访问存储单元

时递增计数器，在访问存储单元所需的周期数超过阈值的情况下递增计数器，或者触发对存储单元的所有访问的计数。然后，将元数据与存储单元相关联(步骤 1904)。接下来，判定是否存在更多存储单元要处理(步骤 1906)。如果存在另外的存储单元，则该过程返回到步骤 1900。否则，该过程终止。

现在参照图 20，其示出根据本发明优选实施例的用于对特定指令的执行进行计数的过程的流程图。图 20 所示的过程可以在指令高速缓存如图 2 的指令高速缓存 214 中实现。

该过程以执行指令(步骤 2000)开始。判定计数器是否与指令相关联(步骤 2002)。计数器可以包括在指令内的字段中，或者可以位于性能检测映像存储器中。如果计数器与指令相关联，则递增计数器(步骤 2004)，然后该过程终止。否则，该过程终止而不递增计数器。如果计数器超过阈值，则可以将计数器清零。

当计数器作为指令的一部分实现时，该计数器可能为有限大小。在这种情况下，计数器的阈值可以设为表示计数器何时处于上溢的危险中。然后，在读到该值之后，则可以将计数器清零。该值可以由性能监测器单元或用来分析数据的程序读取。可以实现 API 来访问该数据。

现在参照图 21，其示出根据本发明优选实施例的用于对特定存储单元的访问进行计数的过程的流程图。图 21 所示的过程可以在数据高速缓存如图 2 的数据高速缓存 216 和指令高速缓存 214 中实现。

该过程以探测对存储单元的访问(步骤 2100)开始。判定计数器是否与存储单元相关联(步骤 2102)。计数器可以包括在存储单元内，或者可以位于性能检测映像存储器中。如果计数器与存储单元相关联，则递增计数器(步骤 2104)，然后该过程终止。否则，该过程终止而不递增计数器。

接下来参照图 22，其是根据本发明优选实施例的用于访问关于指令执行或存储单元访问而收集的信息的组件的图。在本例中，指令单元 2200 执行指令 2202，并且递增计数器 2204。每次执行指令 2202 时，都递增该计数器。在本例中，指令单元 2200 可以实现为图 2 中的指令高速缓存 214。

当指令或数据高速缓存页被装入到存储器中时，操作系统程序装入器/链接器和/或性能监测程序读取由编译器生成的元数据，并且确定计数与指令或数据访问相关联，然后装入过程分配数据区来维护计数器作为其性能检测段的一部分。计数器的大小和数据访问的粒度决定所要分配的工作区的数量。

在简单的情况下，数据或指令访问的粒度可以是字大小(从而对字中的任何字节的访问都被认为是一次访问)并且计数也可以是字大小。在这种情况下，在主段和性能检测段之间存在一对多映射(不需要全字来包含计数或阈值)。装入过程分配一个或多个映像页，并且告诉处理器使用所述一个或多个映像页来包含计数。该映射的详细信息在上面参照图 17 作过描述。处理器中的高速缓存单元维护映像块条目以指示对应页包含计数信息。可以提供不同映射和不同级别的支持。

在另一实施例中，编译器分配工作区来维护计数，并且指示将这些工作区置于其生成的数据区中。元数据中的条目可以表示数据的开始、数据的字节数、数据的粒度、计数区的开始和每个计数单元的粒度。在任何情况下，将元数据装入到处理器中，并且处理器向其内部(映像)高速缓存填充元数据。在修改指令流本身来包含元数据的示例性实施例中，装入器更新指令流以包含适当的指示符和工作区，或者编译器生成了代码来包含元数据。在任一种情况下，在装入了代码之后，处理器接收元数据。

数据单元 2206 可以作为图 2 中的数据高速缓存 206 实现。在本例中，每当访问数据 2208 时，都递增计数器 2210。数据 2208 和计数器 2210 均处于特定存储单元中。在这些例子中，可以采用新指令，其中，该指令称作 ReadDataAccessCount(RDAC)，其获得(take)数据地址和寄存器，并且将与该数据地址相关联的计数置于该寄存器中。

指令执行和数据访问这些事件中的每一个都导致计数器的递增。本发明的机制提供一个接口即硬件接口 2212 来访问所收集的这一数据。在这些例子中，硬件接口 2212 采取用于操作系统 2214 的应用程序编程接口(API)的形式。这样，分析工具 2216 可以从计数器 2204 和计数器 2210 获得数据。分析工具 2216 可以采取多种形式，例如 Oprofile，其是 Linux 系统的公知的全系统剖析器。虽然图 22 中的例子示出向指令单元和数据单元提供接口，但是也可以实现硬件接口 2212 来提供对来自处理器中的其它单元的信息的访问。例如，可以为允许访问位于性能监测器单元的计数器如图 2 的性能监测器单元 240 的计数器 241 和 242 中的信息的硬件接口 2212 创建 API。

在图 23 中，示出根据本发明优选实施例的用于自主修改程序代码以允许对代码部分进行选择性计数或剖析中的组件的方框图。在本例中，剖析器 2300 是可以用来识别程序如程序 2302 中具有高使用率的例程的程序，如 tprof。在

这些例子中，“tprof”是定时器剖析器，其捆绑(ship)在来自国际商业机器(IBM)公司的高级交互性执行体(AIX)操作系统上。该程序采集由定时器发起的样本。当定时器结束时，tprof识别所执行的指令。tprof是可以用于系统性能分析的CPU剖析工具。该工具是分析工具的例子，并且基于包括以下步骤的采样技术：通过时间或性能监测器计数器周期性地中断系统；随同进程 id(pid)和线程 id(tid)一起确定被中断代码的地址；记录软件跟踪缓冲器中的 TPROF挂钩(hook)；并且返回到被中断代码。

或者，可以使用性能监测器计数器的固定数目的计数来代替定时器。该程序剖析用来指示在程序内何处花费了时间的子例程。使用率超过特定阈值的程序也称作“热点(hot)”。通过使用来自剖析器 2300 的信息，可以识别感兴趣的例程如程序 2302 中的子例程 2304。

采用该信息，可以由分析工具 2306 自主修改子例程 2304 中的指令，以允许对子例程 2304 的执行进行计数。可以识别另外的例程以由分析工具 2306进行修改。例如，还可以识别子例程 2304 为感兴趣的例程，并修改该例程的指令以允许对子例程 2304 的执行进行计数。对这些例程中的代码的修改包括将性能指示符与这些子例程中每一个内的一条或多条指令相关联。

在由分析工具 2306 修改了这些例程中的指令之后，由处理器 2308 执行程序 2302。处理器 2308 执行程序 2302 并且为这些例程提供计数。例如，对所执行的指令和执行例程时所用周期数的计数可以使用上述机制由处理器 2308 执行。

参照图 24，其示出根据本发明优选实施例的用于动态地将性能指示符添加到指令或使其与指令关联的过程的流程图。图 24 所示的过程可以在诸如图 23 的分析工具 2306 的程序中实现。分析工具是用来获得有关程序执行的规格的程序。这些规格可以是任何可测量参数，如执行时间、所执行的例程、所执行的特定指令和所访问的存储单元。

该过程以使用来自剖析器的数据识别感兴趣的指令(步骤 2400)开始。该剖析器可以例如是 AIX 中见到的定时器剖析器。从识别出的指令中选择一条指令以作修改(步骤 2402)。然后，将性能指示符动态添加到所选指令(步骤 2404)。

在步骤 2404，可以以无需为执行而修改指令的方式添加指令。可以采用性能检测映像存储器如图 15 中的性能检测映像存储器 1506 来保存性能指示

符。在这种情形下，设置处理器中的寄存器以指示当执行指令时应检查性能检测映像存储器以获得性能指示符。

然后，判定是否存在另外的所识别指令要修改(步骤 2406)。如果存在另外的指令要修改，则该过程返回到步骤 2402。否则，该过程终止。

接下来参照图 25，其示出根据本发明优选实施例的用来通过将性能指示符与页内的指令相关联而扫描页的组件的图。本发明的机制使用性能指示符来允许每次一页地检测(instrument)或修改程序中的指令。

在本例中，程序 2500 包含三页，即页 2502、页 2504 和页 2506。扫描守护进程(daemon)2508 每次一页或多页地将性能指示符与程序 2500 中的指令相关联。例如，页 2502 中的指令可以通过扫描守护进程 2508 与性能指示符相关联。然后，由处理器 2510 执行程序 2500。然后可以收集来自对程序 2500 的执行的数据。该数据包括例如对响应页 2502 中的指令而发生的事件的计数，从而对执行页 2502 中的每条指令的次数进行计数和/或识别对页 2502 的访问次数。

下一步，扫描守护进程可以从页 2502 内的指令中去除性能指示符，并且将性能指示符与页 2504 中的指令相关联。然后，由处理器 2510 再次执行程序 2500，并且收集来自对该程序的执行的数据。然后，可以在所执行的程序 2500 中修改页 2506 中的指令以收集有关该页的数据。

以这种方式，可以识别诸如定时器剖析器的程序通常不记录的对例程的使用。由于中断可能被禁止，或者样本的定时可能产生同步非随机行为，所以定时器剖析器可能不记录对例程的某些使用。通过修改程序 2500 中的指令，可以获得对例程或其它模块的计数，其中，计数是无偏的，并且系统是不受干扰的。以这种方式，避免了中断驱动的计数。此外，虽然对代码的检测是每次一页的，但是在扫描程序时也可以使用指令的其它编组，例如形成程序的模块。例如，编组可以是单个可执行程序、库、一组选定函数和一组选定页。

接下来参照图 26，其示出根据本发明优选实施例的用于将指示符添加到页内指令的过程的流程图。图 26 所示的过程可以在诸如图 25 的扫描守护进程 2508 的程序中实现。

首先，识别页的选择范围(selection)(步骤 2600)。在本例中，这些页是程序中所要扫描或检测的那些页。接下来，选择页的选择范围中的一页以作修

改(步骤 2602)。然后，使指示符与选定页内的所有指令相关联(步骤 2604)。然后执行程序(步骤 2606)。接下来，判定是否扫描了选择范围内的所有页(步骤 2608)。如果扫描了所有页，则该过程随后终止。然而，如果并非所有的页都已被扫描，则选择所要扫描的下一页(步骤 2610)，而该过程返回到如上所述的步骤 2604。

图 26 所示的过程示出所扫描的作为页的指令编组。根据具体实现，可以以这种方式扫描或检测其它类型的指令分组，例如形成程序的模块。

采用程序来根据调用堆栈中找到的信息从例程中识别调用者。该程序通过识别已进行的函数调用，允许识别例程中发生了什么，并且提供对程序中发生了什么的总结。然而，该程序需要将指令插入代码中以获得这一信息。

本发明的机制允许识别调用和返回，而不必执行特别的代码检测。具体地说，可以使用对特定指令集产生中断的函数来收集有关系统和应用程序的信息。在这些例子中，调用和返回的指令与产生中断的性能指示符相关联。

通过向上回巡(walk back)调用堆栈，可以获得完整的调用堆栈以作分析。“堆栈巡视(stack walk)”也可以描述为“堆栈展开(stack unwind)”，而“巡视堆栈”的过程也可以描述为“展开堆栈”。这些术语中的每一个阐明了对该过程的不同比喻。当该过程必须逐步或逐帧地获得和处理堆栈帧时，该过程可以描述为“巡视”。当该过程必须获得和处理指向彼此的堆栈帧时，该过程也可以描述为“展开”，而这些指针及其信息必须通过很多指针解除参考(dereference)来“展开”。

堆栈展开遵循中断时的函数/方法调用顺序，并且响应与性能指示符相关联的指令的执行而生成。调用堆栈是例程加上在程序执行期间进入的例程(即模块、函数、方法等)内偏移的有序列表。例如，如果例程 A 调用例程 B，然后例程 B 调用例程 C，而处理器正在执行例程 C 中的指令，则调用堆栈为 ABC。当把控制从例程 C 返回到例程 B 时，调用堆栈为 AB。为了在所生成的报告内表达更简洁和易于解释起见，提供例程的名称而没有任何偏移信息。偏移可以用于对程序执行更详细的分析，然而，这里不进一步考虑偏移。

因此，在通过执行与特定性能指示符关联的指令而发起的中断处理期间或后处理时，所生成的基于样本的剖析信息反映调用堆栈的采样，而不是如同某些程序计数器采样技术中一样仅仅为可能调用堆栈的叶子(leaf)。叶子是分枝末端的节点，即没有子代的节点。子代是父节点之子，而叶子是无子节

点。

现在参照图 27，其是示出根据本发明优选实施例的包含多个堆栈帧的调用堆栈的图。“堆栈”是保留存储器区域，其中一个或多个程序存储状态数据，如过程和函数调用地址、所传递参数，并且有时还有局部变量。“堆栈帧”是线程堆栈的一部分，其表示单个函数调用的局部存储(参数、返回地址、返回值和局部变量)。每一个活动的执行线程具有为其堆栈空间分配的一部分系统存储器。线程堆栈由堆栈帧序列组成。线程堆栈上的帧集在任何时候都表示该线程的执行状态。由于堆栈帧典型地是相互链接的(例如，每个堆栈帧指向前一堆栈帧)，因此经常有可能向上往回跟踪堆栈帧序列，并且形成“调用堆栈”。调用堆栈表示所有尚未完成的函数调用——换句话说，它反映任何时间点的函数调用序列。

调用堆栈 2700 包括标识正在运行的例程、调用其的例程等等一直到主程序的信息。调用堆栈 2700 包括多个堆栈帧 2702、2704、2706 和 2708。在所示例子中，堆栈帧 2702 位于调用堆栈 2700 的顶部，而堆栈帧 2708 位于调用堆栈 2700 的底部。调用堆栈的顶部也称作“根”。修改(大多数操作系统中所见)的中断，以获得被中断线程的程序计数器值(pcv)以及指向该线程的当前活动堆栈帧的指针。在英特尔架构中，这典型地由寄存器：EIP(程序计数器)和 EBP(指向堆栈帧的指针)的内容表示。

通过访问当前活动的堆栈帧，有可能利用(典型的)堆栈帧链接约定，以便将所有帧链在一起。标准链接约定的一部分还规定函数返回地址正好放在被调用函数的堆栈帧之上；这可以用来确定被调用函数的地址。虽然本讨论采用基于英特尔的架构，但是本例不是限制。大多数架构采用可以类似地由修改的剖析中断处理程序导航的链接约定。

当发生中断时，所获取的第一参数是程序计数器值。下一个值是指向被中断线程的当前堆栈帧的顶部的指针。在所示例子中，该值将指向堆栈帧 2708 中的 EBP 2708a。EBP 2708 又指向堆栈帧 2706 中的 EBP 2706a，而 EBP 2706a 又指向堆栈帧 2704 中的 EBP 2704a。该 EBP 又指向堆栈帧 2702 中的 EBP 2702a。标识调用例程的返回地址的 EIP 2702b-2708b 位于堆栈帧 2702-2708 内。可以根据这些地址来识别这些例程。因此，通过向上或向后巡视堆栈而收集所有返回地址来定义例程。

在某些情况下获得完整的调用堆栈可能是困难的，因为例如当具有一个

调用堆栈的应用程序对具有不同调用堆栈的内核进行调用时，环境可能使跟踪困难。由本发明的机制提供的硬件支持避免了这些问题中的某些问题。

接下来参照图 28，其示出根据本发明优选实施例的用于识别与调用和返回指令相关联的事件的过程的流程图，其中从性能监测器单元收集数据。图 28 所示的过程也可以在分析工具如图 22 的分析工具 2216 中实现。

该过程以识别调用和返回指令(步骤 2800)开始。调用和返回指令是用于确定何时调用了例程和何时例程完成的感兴趣的指令。这可以为中断、中断返回、系统调用和从系统调用返回而实现。

接下来，使性能指示符与所识别的调用和返回指令相关联(步骤 2802)。然后执行程序(步骤 2804)，并且从性能监测器单元收集数据(步骤 2806)，然后该过程终止。该信息可以通过接口如图 22 所示的硬件接口 2212 来收集，其中，采用 API 来获得由处理器中的不同功能单元收集的数据。

利用该数据，可以识别例程的调用者。该信息可以用来产生诸如树的数据结构来跟踪和呈现有关程序执行的信息。数据结构的这一生成可以使用类似于在分析工具中提供的过程来实现。

接下来参照图 29，其示出根据本发明优选实施例的用于识别已被执行多于选定次数的例程的过程的流程图。图 29 所示的过程可以在处理器内的功能单元如图 2 的指令高速缓存 214 中实现。该过程用来识别对被执行指令的计数，并且当这些指令出现次数多于某选定次数时产生中断。

首先，判定是否探测到对选定指令的执行(步骤 2900)。通过检查每条被执行的指令来判定，以查看性能指示符是否与该指令相关联。这些性能指示符可以通过不同工具如图 15 中的编译器 1500 或图 22 中的分析工具 2216 与该指令相关联。

如果没有识别出对包含性能指示符的指令的执行，则该过程返回到步骤 2900，直到探测出选定指令。如果选定指令被识别为正在执行，则为该选定指令递增具有设定阈值的计数器，以对执行该特定指令的频度计数(步骤 2902)。在这些例子中，为被标识以进行监测的每条指令分配计数器。

接下来，判定是否达到设定阈值(步骤 2904)。对于每一个高速缓存级，起初通过使用文档化高速缓存失败次数来确定阈值。然而，增加次数用来确定因高速缓存干扰(来自其它处理器的访问)引起的问题。可以以不同值重复运行，以识别具有最差性能的区域。

在这些例子中，指令可以与包括要监测对指令的执行并提供计数器的指示的指示符相关联。此外，可以包括计数标准来识别何时要产生中断。例如，当指令被执行多于十三次时，可以产生中断。

如果尚未达到阈值，则该过程返回到如上所述的步骤2900。如果达到了设定阈值，则将中断发送到监测程序(步骤 2906)，然后该过程终止。该中断可以被发送到中断单元如图 2 中的中断单元 250，这将控制传到适当的程序或进程以处理中断。

该过程可能对具有很多转移的例程尤其有用。在这种情况下，将标志所有转移指令以进行计数。通过这种计数而获得的信息可能对通过使转移最少化或调整所用处理器的指令架构中支持的提示(hint)标志来识别对编译器和运行时编译执行的(JIT，just-in-time)代码生成的改进有用。

接下来参照图 30，其示出根据本发明优选实施例的用于当特定指令被执行的次数多于某选定次数时检查调用堆栈并识别例程的调用者的过程的流程图。图 7 所示的过程可以由中断单元如图 2 中的中断单元 250 发起。该过程用来识别例程中的调用，并且可以用来递归获得调用者的信息。

首先，检查调用堆栈，并且识别例程的调用者(步骤 3000)。接下来，从指令高速缓存捕获对被执行指令的计数(步骤 3002)。该计数是针对图 29 的步骤 2902 中所用的计数器的。然后将计数器清零(步骤 3004)，随后从中断返回控制(步骤 3006)。可以使用在图 30 的过程中获得的信息来识别另外的要监测的例程以递归识别例程的调用者。

接下来参照图 31，其是示出根据本发明优选实施例的为进行监测而选择的指令和数据的范围的图。在本例中，程序 3100 包括指令范围 3102 和 3104。这些范围中的每一个均被识别为所要监测的感兴趣范围。这些范围中的每一个均设在诸如图 2 的指令高速缓存 214 的指令单元内。每个范围都用来在程序 3100 的执行期间告诉处理器在一范围内执行的指令数以及进入一范围的次数。此外，可以使用范围寄存器 3108 来判定是否识别了要执行的选定类型的指令。

使用来自范围寄存器 3108 的附加或不同机制来识别一种或多种指令类型。一种机制涉及使用具有指令掩码的指令匹配寄存器(IMR)。该掩码用来识别指令中必须匹配特定值的位，或者用来识别其值无关紧要的位。这些位也称作无关位。另一种方案使用操作码或指令类型列表。在该实现中，指示符

可以用来识别哪些指令要结合范围寄存器 3108 使用。

指令高速缓存 3106 使用范围寄存器 3108 来定义指令范围。这些寄存器可以是现有寄存器，或者可以修改处理器来包括定义指令范围的寄存器。这些范围可以基于指令地址。另外，范围寄存器 3108 可以由各种调试器程序和性能工具更新。

如果在诸如指令范围 3102 或指令范围 3104 的范围内执行指令，则在指令高速缓存 3106 中递增计数器。或者，可以发送信号到性能监测器单元如图 2 中的性能监测器单元 240。在这些例子中，性能监测器单元跟踪对该范围内执行的指令数和进入该指令范围的次数的计数。此外，识别由范围寄存器 3108 设置的范围内选定类型的指令可以导致指令高速缓存 3106 发送信号到中断单元。在本例中，中断单元执行代码，以执行可能涉及对指令进行计数、识别和分析指令的调用堆栈或者根据需要执行某种其它分析的过程。

数据访问可以以类似方式监测。例如，数据 3112 包括数据范围 3114。可以以与指令范围 3102 或指令范围 3104 内的指令执行类似的方式对数据范围 3114 的数据访问进行计数。这些范围可以定义在诸如图 2 的数据高速缓存 216 的数据单元内的寄存器中。这些数据范围可以作为数据的存储单元范围定义在寄存器中。

接下来参照图 32，其示出根据本发明优选实施例的用于对设定范围的访问次数以及在设定范围内执行的指令数进行计数的过程的流程图。图 32 所示的过程可以在诸如图 2 的指令高速缓存 214 的指令单元中实现。该过程对识别指令地址范围内特定或选定类型的指令的执行尤其有用。例如，该过程可以用于处理在一个指令范围内发生的转移指令的执行。

首先，识别所要执行的指令(步骤 3200)。接下来，判定指令是否为选定类型(步骤 3202)。可以通过使用指示符或使用与范围寄存器分开的寄存器集中的掩码集来识别指令类型。掩码或指示符用来判定指令是否为选定类型。该选定类型可以是例如转移指令。在这些例子中，由于所采取的转移识别路径流，因此它们尤其令人感兴趣。调用和返回也可以是被识别以进行处理的指令类型，因为这些类型的指令在识别函数和/或子例程方面很有用。可以被识别为选定类型的指令的其它类型的指令是与存储器访问相关的指令，特别是重复的指令，如串传送。

然后，判定指令是否在设定的指令范围内(步骤 3204)。该范围可以通过

检查定义一个或多个指令范围的寄存器来识别。在这些例子中，设定的指令范围可以对应于某代码编组，例如子例程、库或软件模块。如果指令不在设定的指令范围内，则该过程返回到如上所述的步骤 3200。如果该指令在设定的指令范围内，则发送信号到中断单元(步骤 3206)。

在步骤 3206，本发明提供了一种用于处理指令的方法、设备和计算机指令。响应数据处理系统的处理器内的指令高速缓存中接收到要执行的指令，判定指示符是否与指令相关联，以及指令是否为指令范围内的特定类型。如果指示符与指令相关联并且指令为该指令范围内的特定类型，则产生中断。

把控制传到中断单元，以执行用于处理以信号通知的中断的代码。该代码可以简单地对指令的执行进行计数，或者可以获得和分析指令的调用堆栈信息。然后，当中断单元完成处理中断时，处理指令(步骤 3208)，然后，该过程返回到如上所述的步骤 3200。

回到步骤 3204，如果指令不在设定范围内，则该过程返回到步骤 3200。在步骤 3202，如果指令不为选定类型，则该过程也返回到步骤 3200。

可以对数据执行类似的过程。在这种实现中，判定选定类型的数据，例如对未对准(unaligned)数据字的访问。

因此，本发明提供了一种用于在监测对程序的执行时提供协助的改进方法、设备和计算机指令。本发明的机制包括采用由处理器识别以启动对与指示符关联的指令的执行进行计数的指示符。通过该机制启动如上所述的各种计数。此外，采用通过利用指示符与特定指令的关联而提供的信息，本发明的机制还提供在监测和分析程序性能中对程序的各种调整。此外，如上所述，可以自动调整程序，以允许监测选定指令乃至例程和模块而不必修改程序。

值得注意的是，虽然本发明是在完全功能数据处理系统的上下文中描述的，但是本领域的普通技术人员应当理解本发明的过程能够以指令的计算机可读介质的形式和各种形式来分发，并且本发明与实际上用来执行分发的信号承载介质的具体类型无关地同等适用。计算机可读介质的例子包括诸如软盘、硬盘驱动器、RAM、CD-ROM、DVD-ROM 的可记录型介质和诸如数字和模拟通信链路、采用各种传输形式例如射频和光波传输的有线或无线通信链路的传输型介质。计算机可读介质可以采取为在特定数据处理系统中实际使用而译码的编码格式的形式。

本发明的描述是为了示例和描述的目的而提供的，不旨在穷举或者将本

发明限定于所公开的形式。对于本领域的普通技术人员而言，很多修改和变动将是显然的。例如，代替使用指令中或指令包中的字段，可以使用新指令或操作码来指示后续的指令或后续的指令集是被标记指令。此外，在希望在指令的字段内包括性能指示符的情况下，如果用于性能指示符的空闲字段不可用，则可以改变处理器的架构来包括附加位。此外，虽然给出了诸如指令执行的事件、执行指令所需时间如时钟或处理器周期、访问数据及进入到代码部分中的时间的例子，但是这些例子并不旨在将本发明限定于可以计数的事件的类型。可以使用本发明的机制对任何与对指令的执行或对存储单元的访问有关的事件进行计数。

选择和描述这些所示实施例是为了最佳地说明本发明的原理、实际应用，并使得本领域的其他普通技术人员能够理解本发明能有计划地以具有各种修改的各种实施例来适用于具体应用。

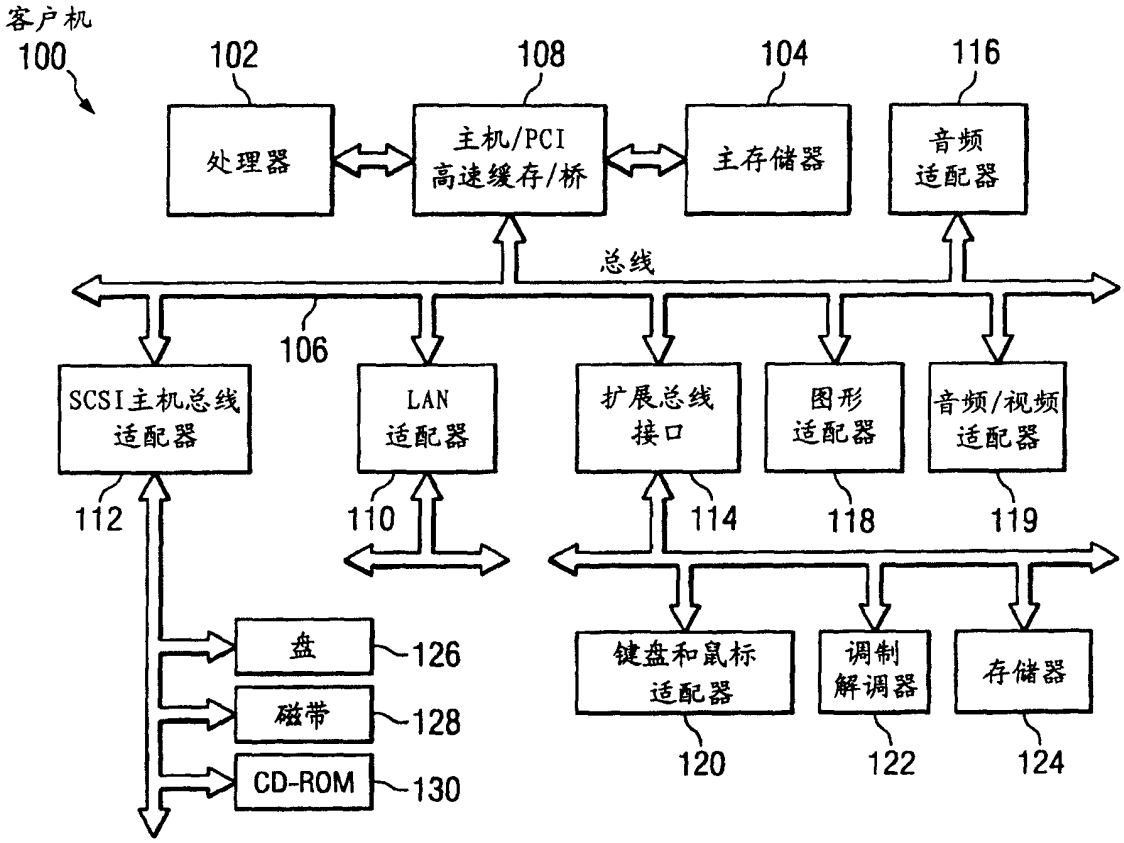

客户机
100
102
108
104
116
处理器
主机/PCI
高速缓存/桥
主存储器
音频
适配器
总线
106
SCSI主机总线
适配器
LAN
适配器
扩展总线
接口
图形
适配器
音频/视频
适配器
112
110
114
118
119
盘
126
磁带
128
CD-ROM
130
键盘和鼠标
适配器
调制
解调器
存储器
120
122
124

图 1

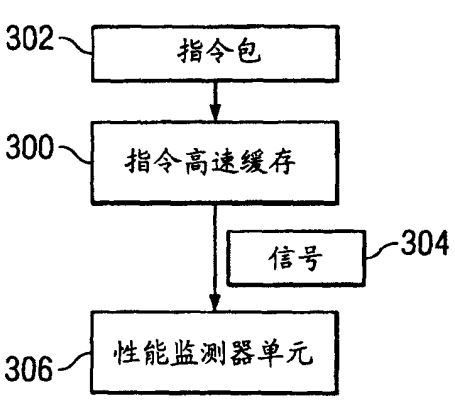

302
指令包
300
指令高速缓存
信号
304
306
性能监测器单元

图 3

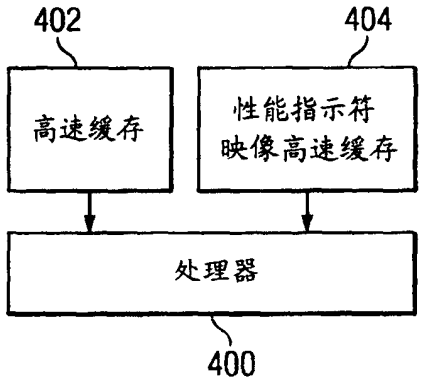

402
404
高速缓存
性能指示符
映像高速缓存
处理器
400

图 4

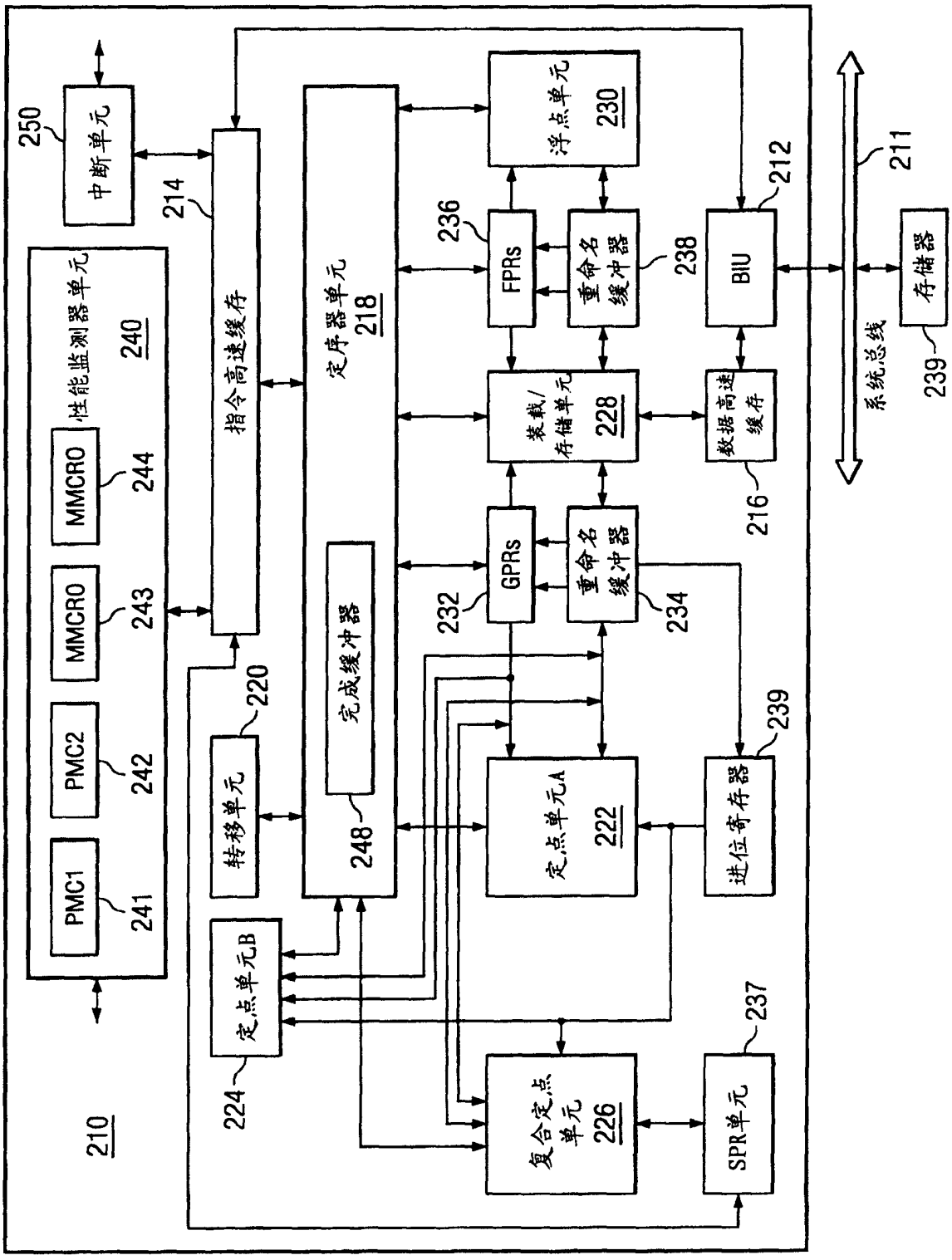

210
250
中断单元
214
指令高速缓存
定序器单元
218
浮点单元
230
236
FPRs
重命名
缓冲器
238
BIU
212
211
系统总线
存储器
239
性能监测器单元
240
PMC1
241
PMC2
242
MMCR0
243
MMCR0
244
装载/
存储单元
228
数据高速
缓存
216
完成缓冲器
248
GPRs
232
重命名
缓冲器
234
转移单元
220
定点单元A
222
进位寄存器
239
定点单元B
224
复合定点
单元
226
SPR单元
237

图 2

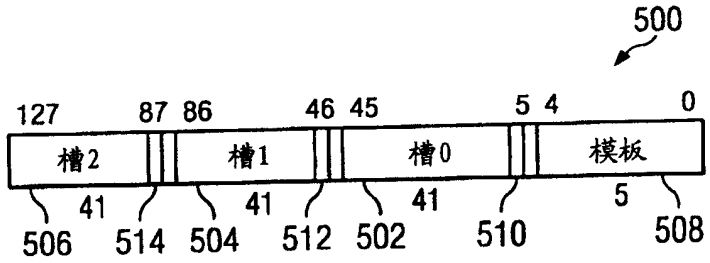

500
127
87 86
46 45
5 4
0
槽2
槽1
槽0
模板
41
41
41
5
506
514
504
512
502
510
508

图 5

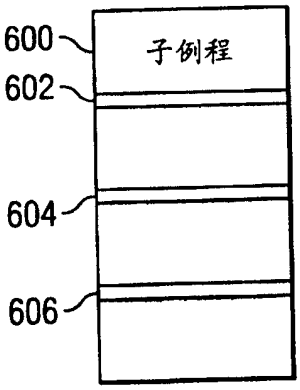

600
602
604
606
子例程

图 6A

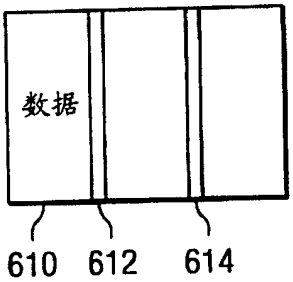

数据
610
612
614

图 6B

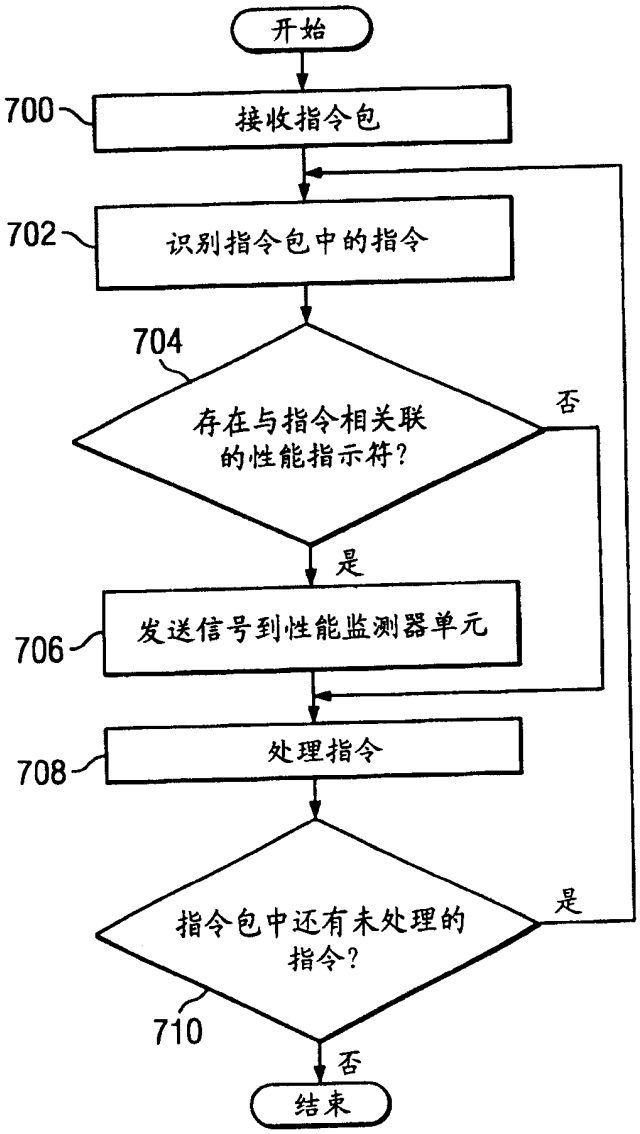

开始
700
接收指令包
702
识别指令包中的指令
704
存在与指令相关联的性能指示符?
否
是
706
发送信号到性能监测器单元
708
处理指令
指令包中还有未处理的指令?
是
710
否
结束

图 7

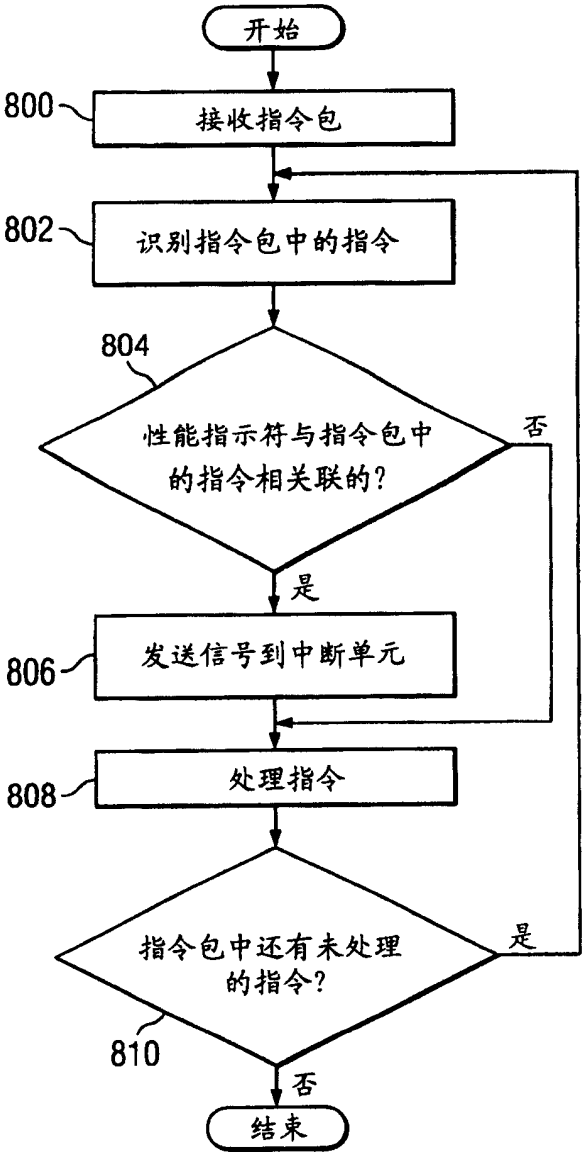

开始
800
接收指令包
802
识别指令包中的指令
804
性能指示符与指令包中的指令相关联的?
否
是
806
发送信号到中断单元
808
处理指令
指令包中还有未处理的指令?
是
810
否
结束

图 8

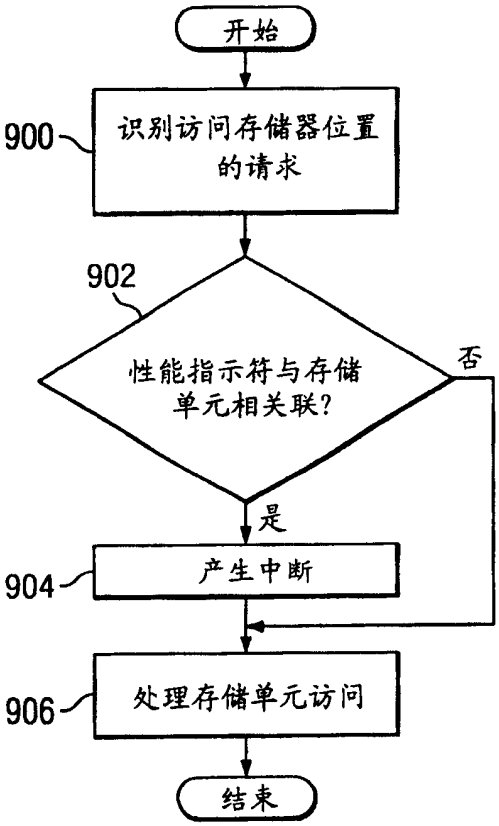

开始
900
识别访问存储器位置的请求
902
性能指示符与存储单元相关联?
否
是
904
产生中断
906
处理存储单元访问
结束

图 9

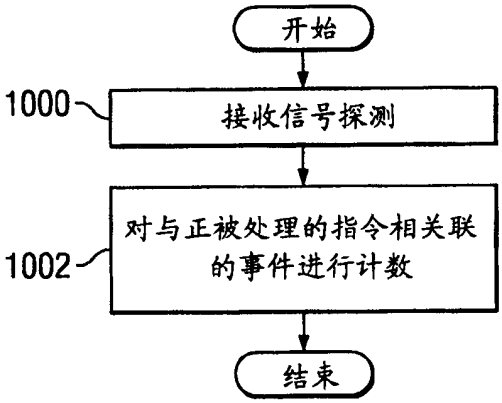

开始
1000
接收信号探测
1002
对与正被处理的指令相关联的事件进行计数
结束

图 10

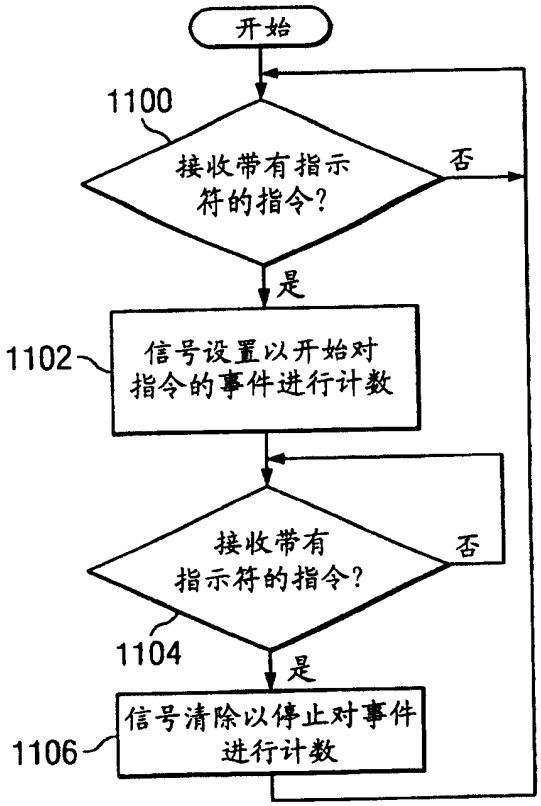

开始
1100
接收带有指示符的指令?
否
是
1102
信号设置以开始对指令的事件进行计数
接收带有指示符的指令?
否
1104
是
1106
信号清除以停止对事件进行计数

图 11

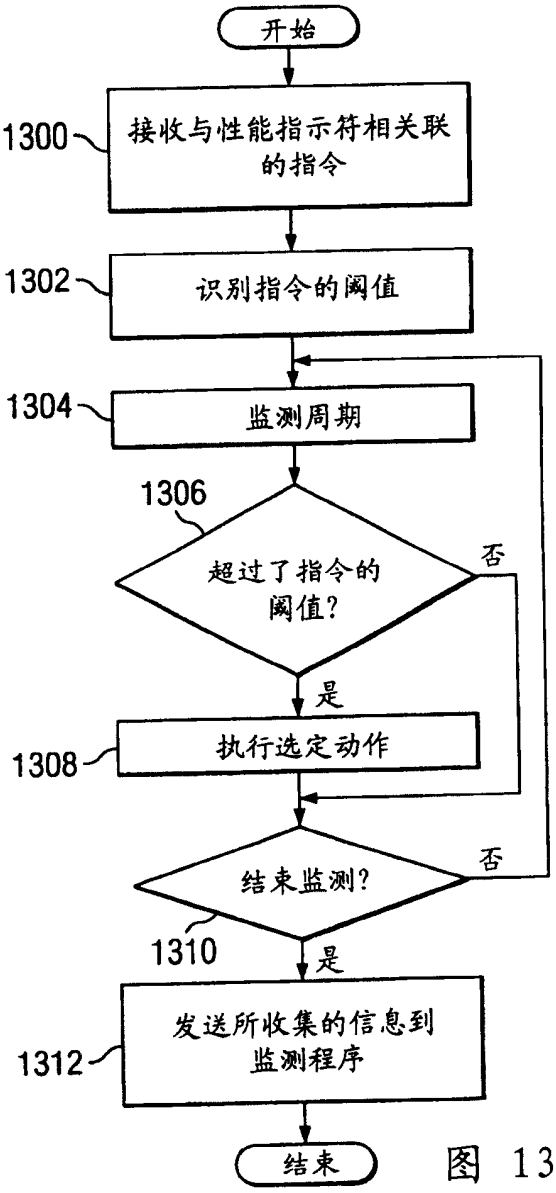

开始
1300
接收与性能指示符相关联的指令
1302
识别指令的阈值
1304
监测周期
1306
超过了指令的阈值?
否
是
1308
执行选定动作
结束监测?
否
1310
是
1312
发送所收集的信息到监测程序
结束

图 13

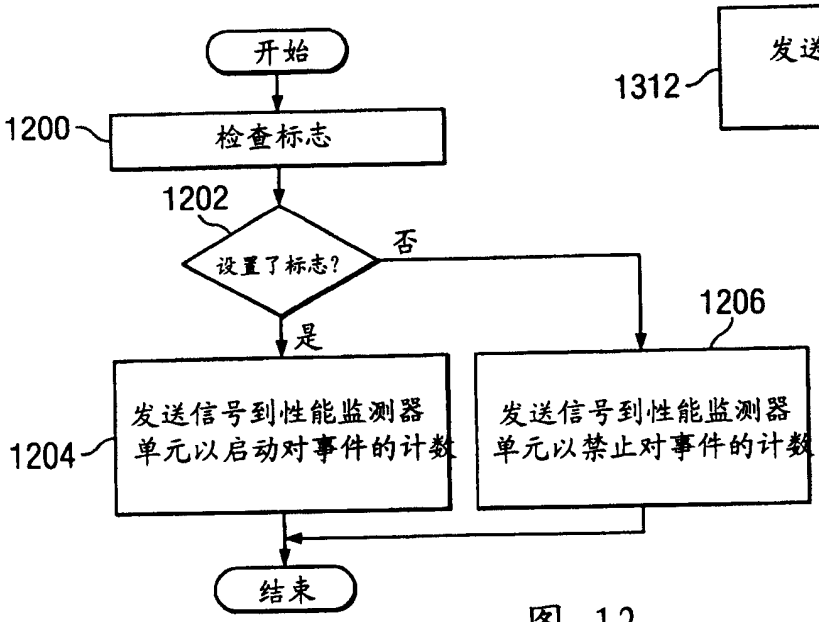

开始
1200
检查标志
1202
设置了标志?
否
是
1206
1204
发送信号到性能监测器单元以启动对事件的计数
发送信号到性能监测器单元以禁止对事件的计数
结束

图 12

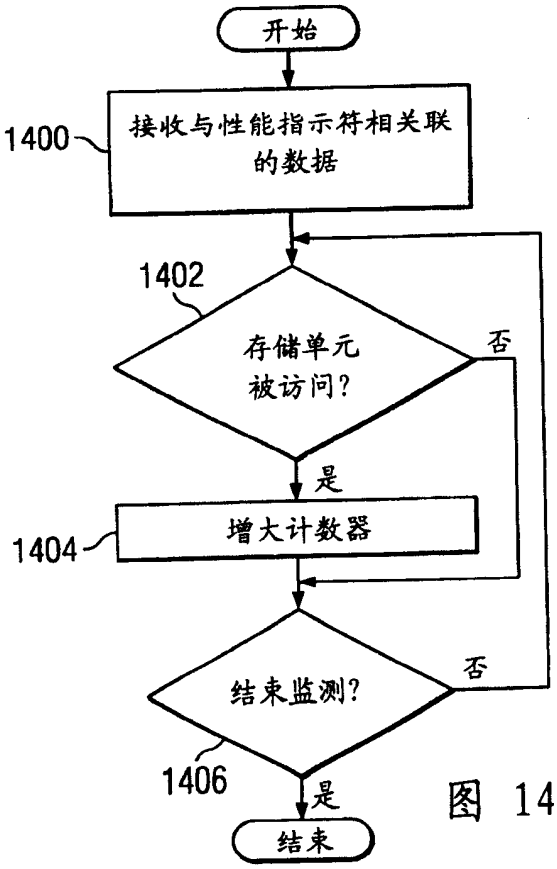

开始
1400
接收与性能指示符相关联的数据
1402
存储单元被访问?
否
是
1404
增大计数器
结束监测?
否
1406
是
结束

图 14

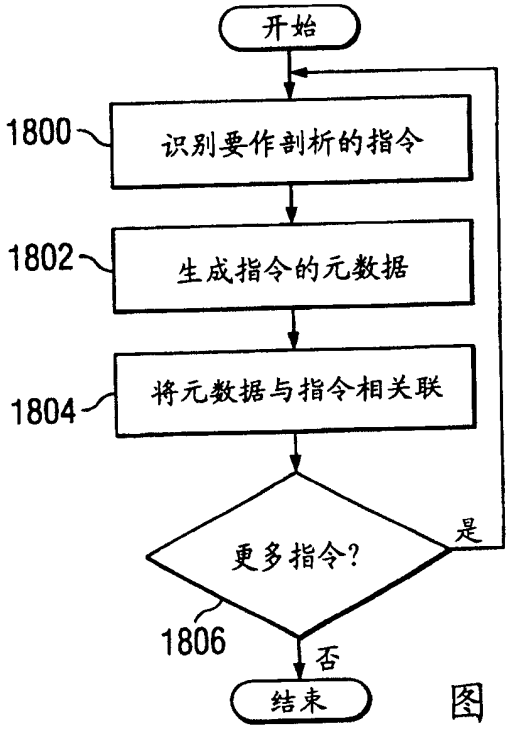

开始
1800
识别要作剖析的指令
1802
生成指令的元数据
1804
将元数据与指令相关联
更多指令?
是
1806
否
结束

图 18

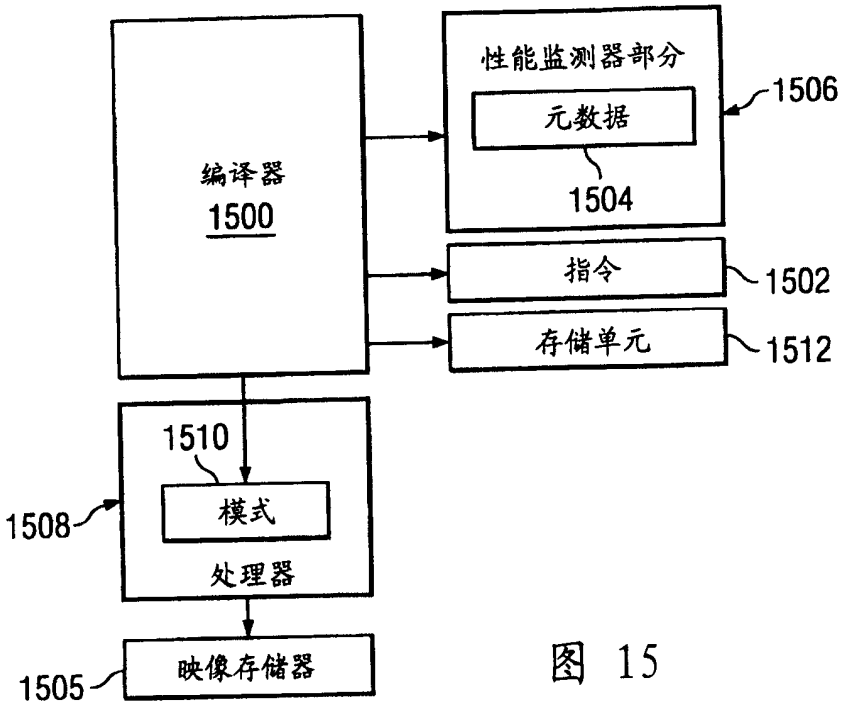

编译器
1500
性能监测器部分
元数据
1504
1506
指令
1502
存储单元
1512
1510
模式
1508
处理器
映像存储器
1505

图 15

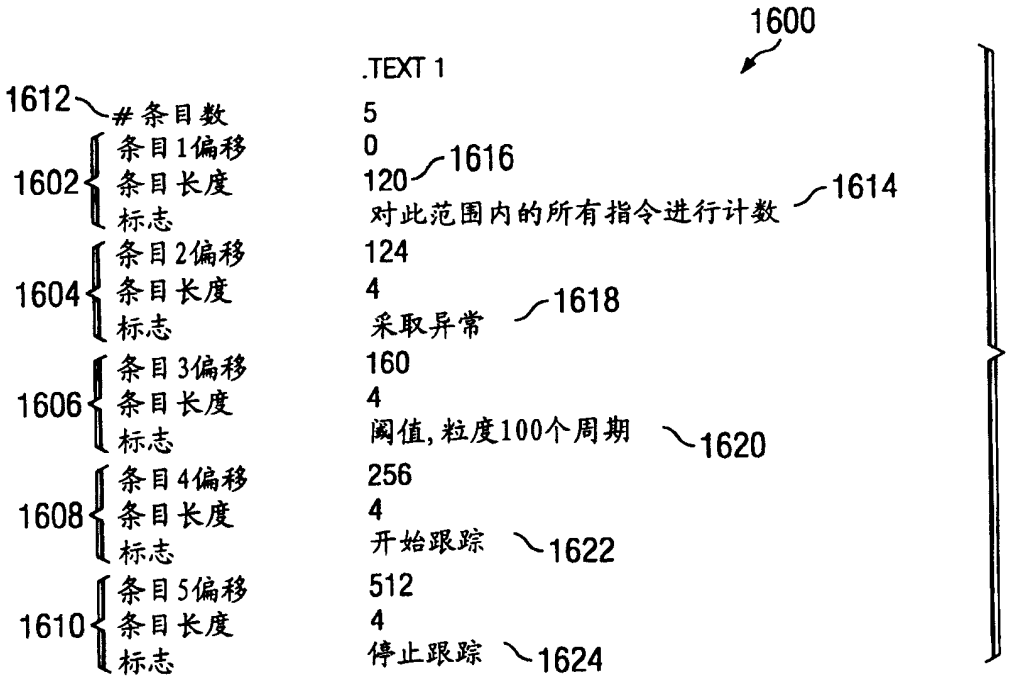

1600
.TEXT 1
1612
条目数
5
1602
条目1偏移
条目长度
标志
0
120
1616
对此范围内的所有指令进行计数
1614
1604
条目2偏移
条目长度
标志
124
4
采取异常
1618
1606
条目3偏移
条目长度
标志
160
4
阈值,粒度100个周期
1620
1608
条目4偏移
条目长度
标志
256
4
开始跟踪
1622
1610
条目5偏移
条目长度
标志
512
4
停止跟踪
1624

图 16

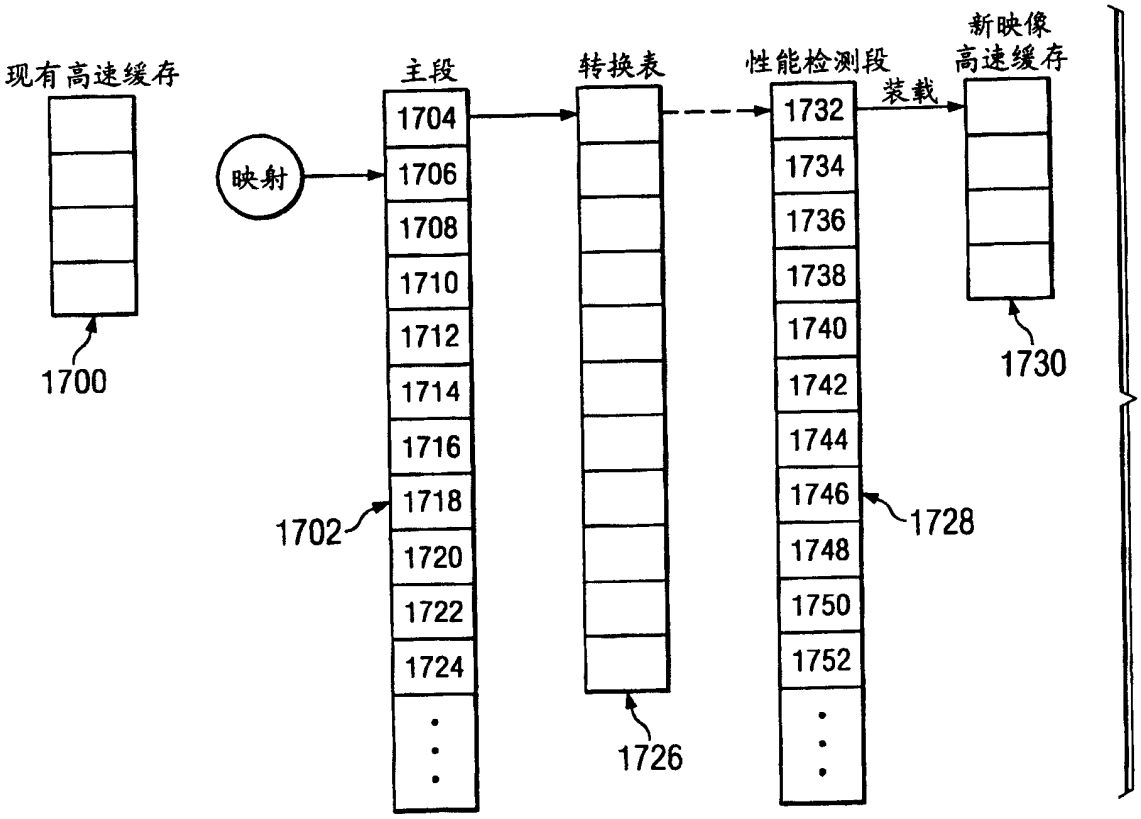

现有高速缓存
1700
映射
主段
1704
1706
1708
1710
1712
1714
1716
1718
1720
1722
1724
1702
转换表
1726
性能检测段
1732
1734
1736
1738
1740
1742
1744
1746
1748
1750
1752
1728
装载
新映像
高速缓存
1730

图 17

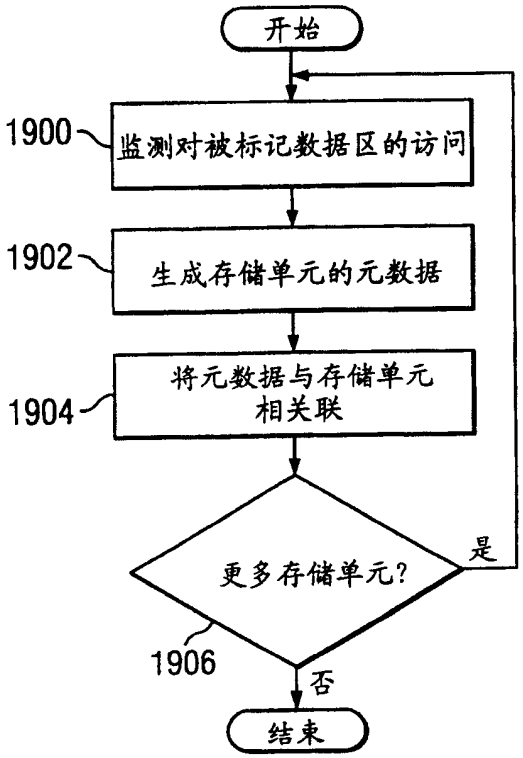

开始
1900
监测对被标记数据区的访问
1902
生成存储单元的元数据
1904
将元数据与存储单元相关联
更多存储单元?
是
1906
否
结束

图 19

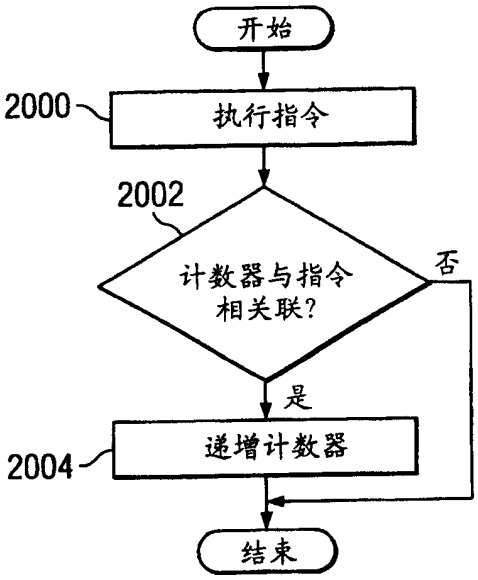

开始
2000
执行指令
2002
计数器与指令相关联?
否
是
2004
递增计数器
结束

图 20

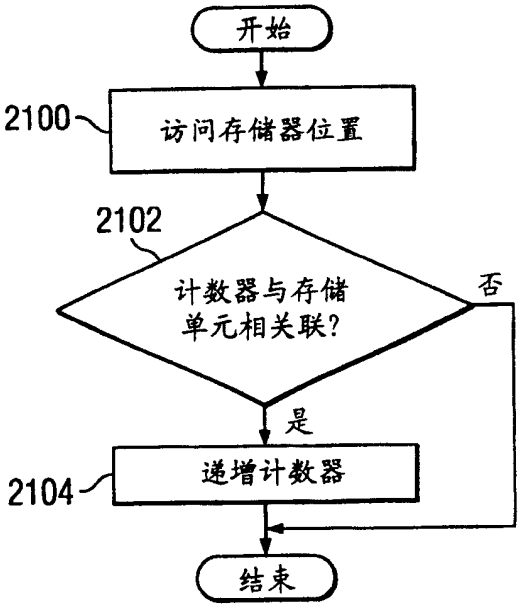

开始
2100
访问存储器位置
2102
计数器与存储单元相关联?
否
是
2104
递增计数器
结束

图 21

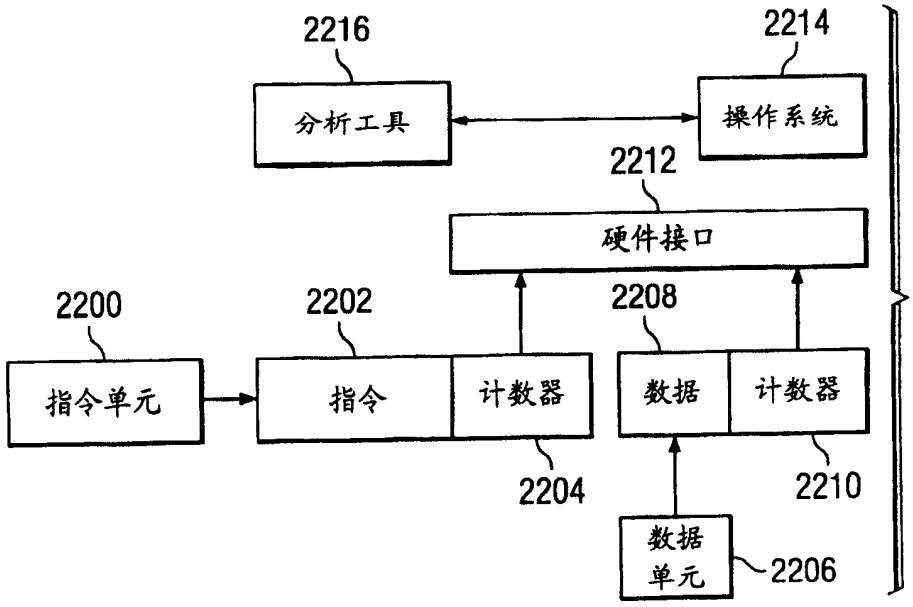

2216
分析工具
2214
操作系统
2212
硬件接口
2200
指令单元
2202
指令
计数器
2204
2208
数据
计数器
2210
数据
单元
2206

图 22

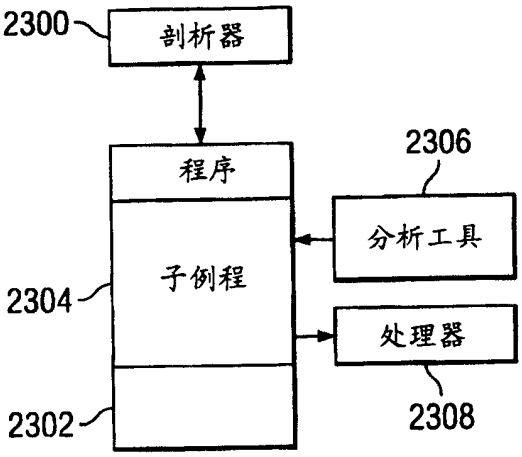

2300
剖析器
程序
2306
分析工具
子例程
2304
处理器
2302
2308

图 23

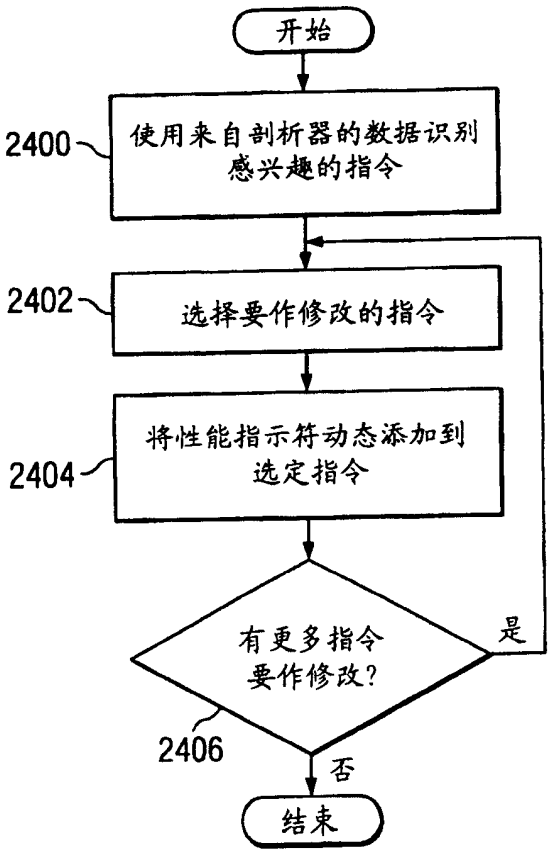

开始
2400
使用来自剖析器的数据识别感兴趣的指令
2402
选择要作修改的指令
2404
将性能指示符动态添加到选定指令
有更多指令要作修改?
是
2406
否
结束

图 24

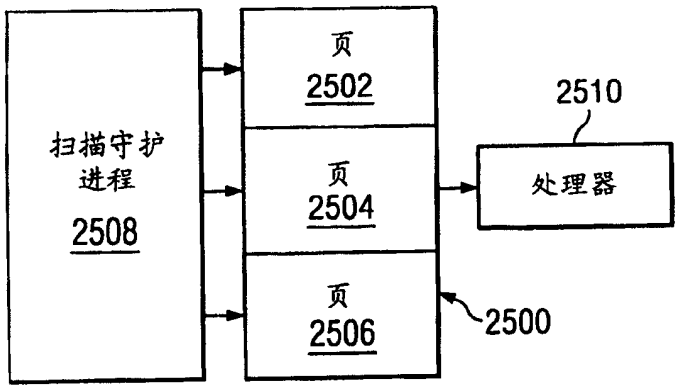

扫描守护
进程
2508
页
2502
页
2504
页
2506
2510
处理器
2500

图 25

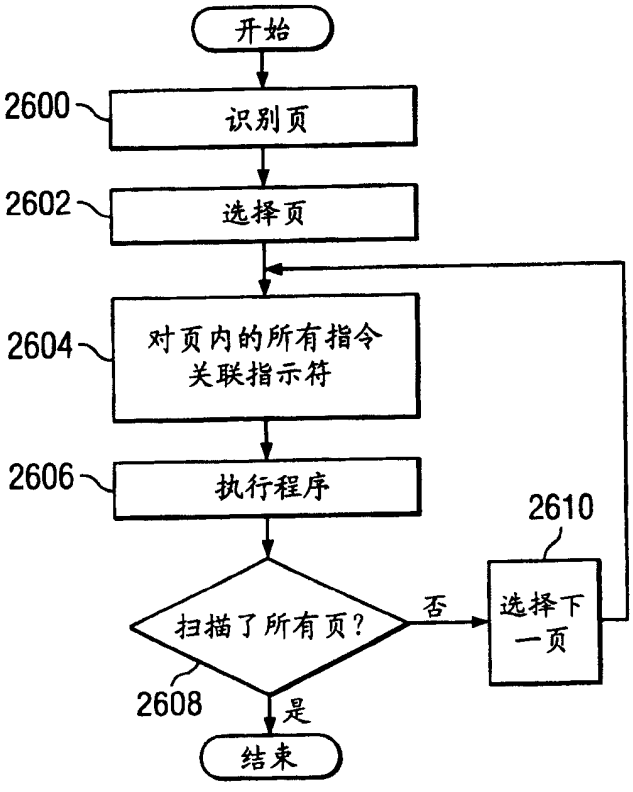

开始
2600
识别页
2602
选择页
2604
对页内的所有指令
关联指示符
2606
执行程序
2610
扫描了所有页?
否
选择下
一页
2608
是
结束

图 26

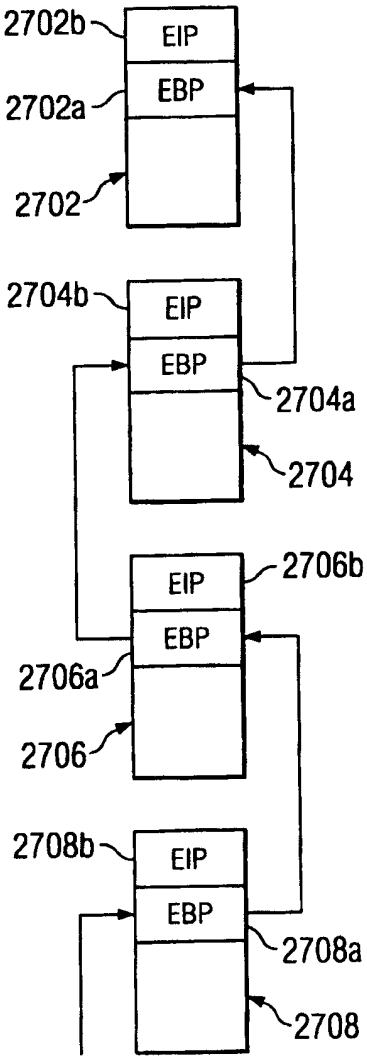

2702b
EIP
EBP
2702a
2702
2704b
EIP
EBP
2704a
2704
2706b
EIP
EBP
2706a
2706
2708b
EIP
EBP
2708a
2708

图 27

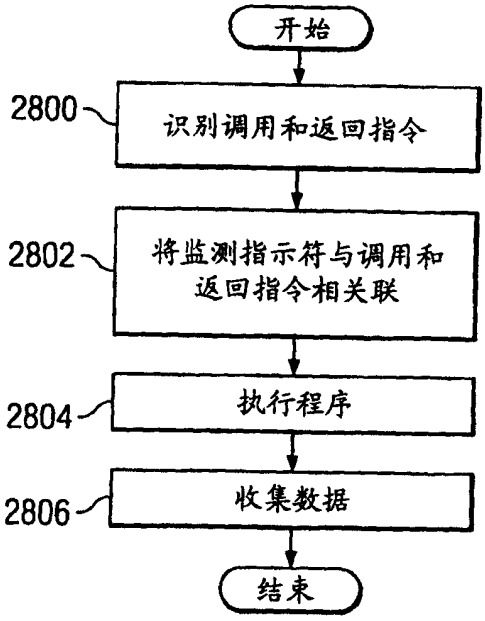

开始
2800
识别调用和返回指令
2802
将监测指示符与调用和返回指令相关联
2804
执行程序
2806
收集数据
结束

图 28

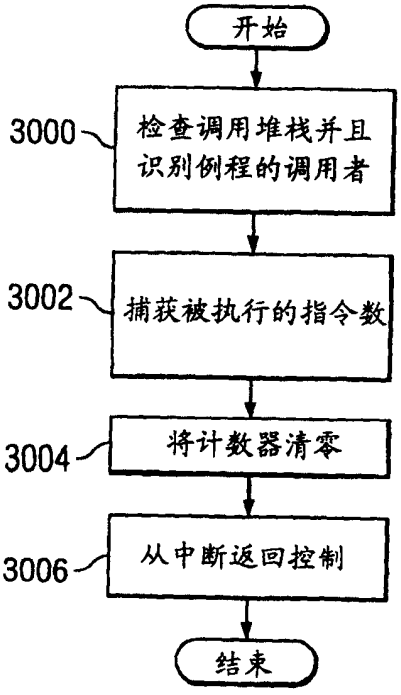

开始
3000
检查调用堆栈并且识别例程的调用者
3002
捕获被执行的指令数
3004
将计数器清零
3006
从中断返回控制
结束

图 30

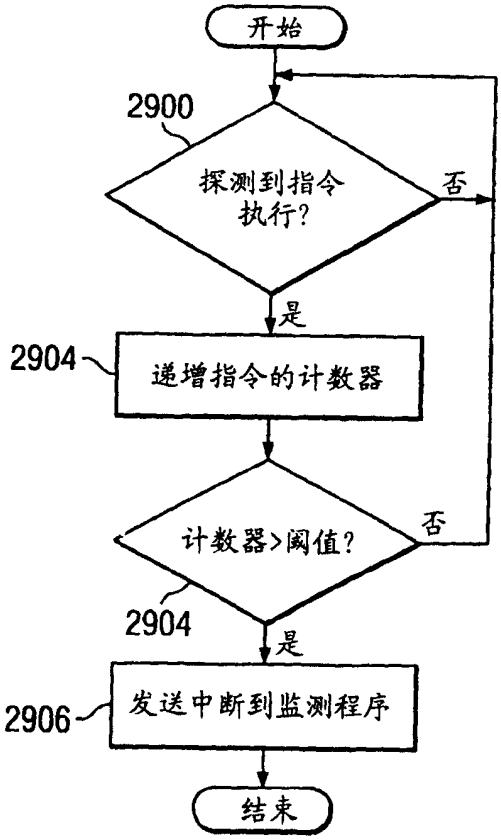

开始
2900
探测到指令执行?
否
是
2904
递增指令的计数器
计数器>阈值?
否
2904
是
2906
发送中断到监测程序
结束

图 29

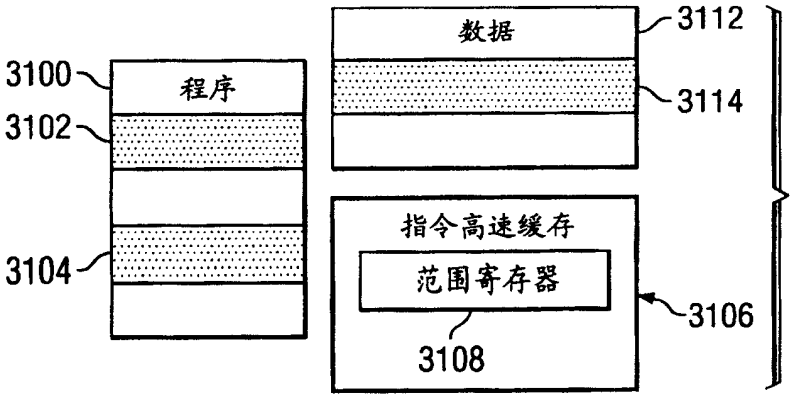

数据
3112
3100
程序
3114
3102
3104
指令高速缓存
范围寄存器
3106
3108

图 31

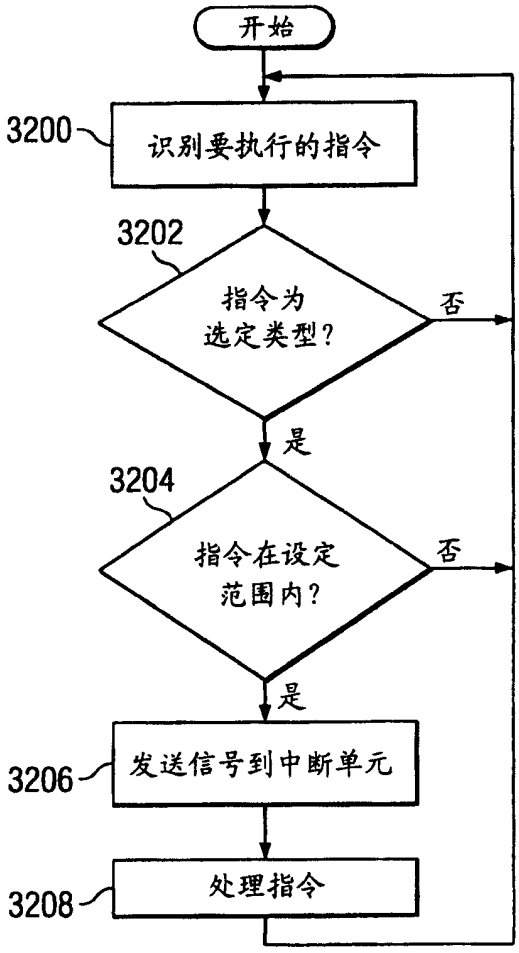

开始
3200
识别要执行的指令
3202
指令为
选定类型?
否
是
3204
指令在设定
范围内?
否
是
3206
发送信号到中断单元
3208
处理指令

图 32