[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/38

[12] 发明专利说明书

[21] ZL 专利号 01816274.6

[45] 授权公告日 2005 年 6 月 22 日

[11] 授权公告号 CN 1207659C

[22] 申请日 2001.8.30 [21] 申请号 01816274.6
[30] 优先权
[32] 2000. 9. 8 [33] US [31] 60/231,452
[86] 国际申请 PCT/US2001/041962 2001.8.30
[87] 国际公布 WO2002/021268 英 2002.3.14
[85] 进入国家阶段日期 2003.3.25
[71] 专利权人 太阳微系统公司
地址 美国加利福尼亚州
[72] 发明人 S·乔杜里 M·特伦布莱
审查员 杨晓明

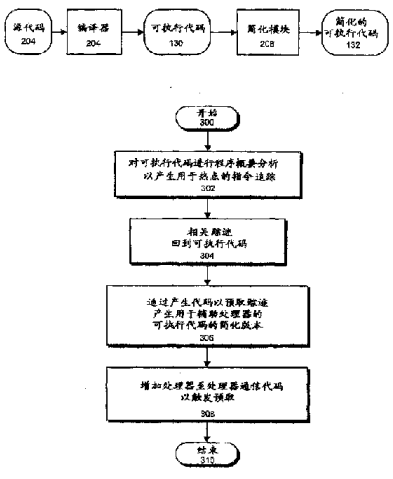
[74] 专利代理机构 北京纪凯知识产权代理有限公司
代理人 赵蓉民 沙 捷

权利要求书 4 页 说明书 10 页 附图 6 页

[54] 发明名称 以辅助处理器预取用于主要处理器的指令的方法和装置

[57] 摘要

本发明的一个实施例提供了一种系统，其通过使用一个辅助处理器在一个主要处理器之前执行预取操作来预取指令。 通过在该主要处理器上执行可执行代码且同时在该辅助处理器上执行一个简化版本的可执行代码来操作该系统。 这个简化版本的可执行代码比该可执行代码执行更加迅速，并且在该主要处理器需要该指令之前，为该主要处理器执行预取操作。 该系统也存储该预取指令到一个由该主要处理器可访问的高速缓存，从而无需从主内存取回该预取指令，该主要处理器即能够访问该预取指令。 在本发明的一个实施例中，在执行该可执行代码之前，该系统编译源代码为用于该主要处理器的可执行代码。 然后，该系统对该可执行代码进行程序概要分析以产生该可执行代码频繁涉及部分的指令追踪。 然后，该系统通过产生预取指令而将该指令追踪部分预取到一个该主要处理器可访问的高速缓存中，产生用于该辅助处理器的可执行代码的简化版本。 该系统也插入通信指令到用于主要处理器的该可执行代码和用于辅助处理器的可执行代码的简化版本中，以从该主要处理器传送进展信息给该辅助处理器。 这个进展信息触发该辅助处理器执行预取操作。



ISSN1008-4274

1．一种通过用辅助处理器在主要处理器开始之前执行预取操作来预取指令的方法，包括：

在该主要处理器上执行可执行代码；

同时在该辅助处理器上执行一个该可执行代码的简化版本，其中该可执行代码的简化版本比该可执行代码执行更加迅速，并且在该主要处理器需要该指令之前，为该主要处理器执行预取操作以取得预取指令；和

将该预取指令存储到一个该主要处理器可访问的高速缓存中，使得无需从主内存取回该预取指令，该主要处理器即能访问该预取指令。

2．根据权利要求1的方法，进一步包括在执行该可执行代码之前：

编译源代码为用于该主要处理器的可执行代码；

对该可执行代码进行程序概要分析以产生指令追踪；

通过产生预取指令而将该指令追踪部分预取到一个该主要处理器可访问的高速缓存中，产生用于该辅助处理器的该可执行代码的简化版本，和

将通信指令插入到用于该主要处理器的该可执行代码，和插入到用于该辅助处理器的该可执行代码的简化版本，以从该主要处理器传送进展信息给该辅助处理器；

其中该进展信息触发该辅助处理器以执行该预取操作。

3．根据权利要求2的方法，其中该编译源代码的步骤和该产生可执行代码的简化版本的步骤是由一个编译器执行的。

4．根据权利要求2的方法，其中如果给该辅助处理器的该进展信息表示该辅助处理器已经沿着错误的路径预取指令，该可执行代码的简化版本使得该辅助处理器中断预取。

5．根据权利要求1的方法，其中将该可执行代码的简化版本配置为：

从记录分支历史信息和调用历史信息的专用的硬件读取控制流程

历史信息；

基于该控制流程历史信息，构成一个经由该可执行代码推算的路径；和

沿着该推算的路径执行预取操作，以预取用于该主要处理器的指令。

6．根据权利要求1的方法，其中产生该可执行代码的简化版本包括：

构成一个用于该可执行代码的控制流程图；

从该控制流程图除去循环；

除去与该控制流程图无关的可执行代码指令；和

将该预取指令插入到该可执行代码的简化版本中，以从该可执行代码预取用于该主要处理器的指令。

7．根据权利要求1的方法，其中执行该预取操作包括预取包含用于该主要处理器的多个指令的高速缓存模块。

8．根据权利要求1的方法，进一步包括经由一个单向通信信道周期性地从该主要处理器发送进展信息给该辅助处理器。

9．一种利于从内存进行预取的装置，包括：

一个配置用以执行可执行代码的主要处理器；

一个配置用以同时执行该可执行代码的简化版本的辅助处理器，其中该可执行代码的简化版本比该可执行代码执行更加迅速，并且在该主要处理器需要该指令之前，为该主要处理器执行预取操作以取得预取指令；和

一个高速缓存，其可为该主要处理器访问并且配置用以存储所预取的指令，使得无需从主内存取回该预取指令，该主要处理器即能访问该预取指令。

10．根据权利要求9的装置，进一步包括一个编译结构，该编译结构配置用以：

编译源代码为用于该主要处理器的可执行代码；

对该可执行代码进行程序概要分析以产生指令追踪；

通过产生预取指令以预取该指令追踪部分进入一个由该主要处理器可访问的高速缓存，产生用于该辅助处理器的该可执行代码的简化版本；和

将通信指令插入到用于主要处理器的该可执行代码，和插入到用于辅助处理器的该可执行代码的简化版本，以从该主要处理器传送进展信息给该辅助处理器；

其中该进展信息触发该辅助处理器以执行该预取操作。

11．根据权利要求10的装置，其中如果给该辅助处理器的该进展信息表示该辅助处理器已经沿着错误的路径预取指令，该可执行代码的简化版本使得该辅助处理器中断预取。

12．根据权利要求9的装置，进一步包括记录用于该可执行代码的分支历史信息和调用历史信息的专用的硬件，并且其中该可执行代码的简化版本被配置为：

从记录分支历史信息和调用历史信息的专用的硬件读取控制流程历史信息；

基于该控制流程历史信息，构成一个经由该可执行代码推算的路径；和

沿着该推算的路径执行预取操作，以预取用于该主要处理器的指令。

13．根据权利要求9的装置，其中在产生该可执行代码的简化版本中，编译结构被配置为：

构成一个用于该可执行代码的控制流程图；

从该控制流程图除去循环；

除去与该控制流程图无关的可执行代码指令；和

将该预取指令插入到该可执行代码的简化版本中，以从该可执行代码预取用于该主要处理器的指令。

14．根据权利要求9的装置，其中该预取操作被配置为预取包含用于该主要处理器的多个指令的高速缓存模块。

15．根据权利要求9的装置，其中该主要处理器和该辅助处理器驻留在相同的半导体芯片上。

16．根据权利要求9的装置，其中该主要处理器和该辅助处理器驻留在不同的半导体芯片上。

17．根据权利要求 9 的装置，其中该辅助处理器是该主要处理器的简化版本，该辅助处理器没有与内存存取操作无关的硬件。

以辅助处理器预取用于主要处理器的指令的方法和装置

背景资料

发明领域

本发明涉及用于改善计算机系统性能的技术。更具体地说，本发明涉及一种在主要处理器开始操作之前通过用辅助处理器执行预取操作而从内存中预取指令的方法和装置。

相关技术

由于半导体集成密度提高，允许更多的晶体管被集成在一个微处理器芯片上，计算机设计师正在研究以不同的方法使用这些晶体管以提高计算机系统性能。为此目的，计算机设计者正在着手合并多个中央处理单元(CPU)为单个半导体芯片。这可以导致用于计算任务的性能增益，该计算任务可以被并行化(分解)为可同时执行的独立的部分。

令人遗憾的是，在许多包含固有的代码串行部分的应用中，得自并行化的性能增益会受到限制。对于这些固有的代码串行部分，由于内存延迟问题，性能受到进一步限制。

随着处理器时钟速度持续以指数速度改善，内存延迟问题正在变得更加恶化。以现在的处理器时钟速度，要用多达200个处理器时钟周期从主要内存拉出一个高速缓存线（cache line）。

计算机设计者们目前使用许多技术降低内存延迟：（1）可利用乱序执行（out-of-order）计划加载和存储，从而尽可能地隐藏内存延迟。令人遗憾的是，乱序执行典型地受限于隐藏少许的内存延迟的时钟周期。（2）当地址无效时可利用无故障加载（non-faulting load）指令推测性地加载一个数据值而不引起故障。（3）可利用导引加载（steering load）指令推测地加载一个数据值进入二级缓存（L2 cache）而非一级缓存（L1 cache），以使得一级缓存不会被未用过的数据值所污染。令人遗憾的是，使用无故障加载和导引加载可能导致不必要的加载。这会浪费指令高速缓存空间并且占用寄存器。（4）某些研究人员已经试

验使用硬件预取引擎，但是这些硬件预取引擎典型地对于不规则的内存存取模式是无效的。

在指令取出操作期间，内存延迟也可能成为一个问题。请注意，一个指令高速缓存失误可能引起与数据数据高速缓存失误一样多的延迟。也请注意，推算下一个很可能要执行的指令是非常困难的，因为通常要有许多的分支和功能调用分散到以现代编程语言编写的程序代码中。

现在需要的是一种在指令取出操作期间减少内存延迟的方法和装置。

发明内容

本发明的一个实施例提供了一种系统，该系统通过使用一个辅助处理器而在一个主要处理器之前执行预取操作来预取指令。通过在该主要处理器上执行可执行代码，并且同时在该辅助处理器上执行一个简化版本的可执行代码来操作该系统。这个简化版本的可执行代码比该可执行代码执行起来更加迅速，并且在该主要处理器需要指令之前，为该主要处理器执行预取操作。该系统也将该预取指令存储到一个可由该主要处理器访问的高速缓存中，从而无需从主要内存取回该预取指令，该主要处理器就能够访问该预取指令。

在本发明的一个实施例中，在执行该可执行代码之前，该系统将源代码编译成用于该主要处理器的可执行代码。然后，该系统对该可执行代码进行概要程序分析（profile），以产生用于该可执行代码频繁涉及部分的指令追踪。然后，该系统通过产生预取指令而将该指令追踪部分预取到一个该主要处理器可访问的高速缓存中，产生用于该辅助处理器的可执行代码的简化版本。该系统也将通信指令插入到该可执行代码（其用于主要处理器）和可执行代码的简化版本（其用于辅助处理器）中，以将进展信息从该主要处理器传送给该辅助处理器。这个进展信息触发该辅助处理器执行该预取操作。

在本发明的一个实施例中，编译源代码的过程和产生可执行代码的简化版本的过程是由一个编译器执行的。

在本发明的一个实施例中，如果给辅助处理器的该进展信息表示

该辅助处理器已经沿着错误的路径预取指令，该可执行代码的简化版本使得该辅助处理器中断预取。

在本发明的一个实施例中，该可执行代码的简化版本被配置为从专用的硬件读取控制流程历史信息，该专用硬件记录分支历史信息和调用历史信息。然后，该可执行代码的简化版本基于该控制流程历史信息构成一个经由该可执行代码推算的路径，然后执行沿着该推算路径的预取操作，以便预取用于该主要处理器的指令。

在本发明的一个实施例中，产生该可执行代码的简化版本，包括构成一个用于该可执行代码的控制流程图。在这样操作时，该系统从该控制流程图中除去循环，并且除去与该控制流程图无关的可执行代码指令。该系统也将该预取指令插入到该可执行代码的简化版本中，以从该可执行代码中预取用于该主要处理器的指令。

在本发明的一个实施例中，执行预取操作包括预取包含用于该主要处理器的多个指令的高速缓存模块。

在本发明的一个实施例中，该系统经由一个单向通信信道周期性地从该主要处理器发送该进展信息给该辅助处理器。

在本发明的一个实施例中，该主要处理器和该辅助处理器驻留在相同的半导体芯片上。

在本发明的一个实施例中，该主要处理器和该辅助处理器驻留在不同的半导体芯片上。

在本发明的一个实施例中，该辅助处理器是该主要处理器的一个简化版本。

附图简要说明

图1举例说明一个按照本发明实施例的计算机系统。

图2举例说明按照本发明一个实施例的编译过程。

图3是一个流程图，举例说明按照本发明一个实施例产生可执行代码的简化版本的过程。

图4举例说明按照本发明一个实施例用以简化可执行代码的结构例子。

图5是一个流程图，举例说明按照本发明一个实施例如何去构成可

执行代码的简化版本的细节。

图6是一个流程图，举例说明按照本发明的一个实施例，当确定已经沿着错误的路线进行的时候，如何弹射出（bail out）可执行代码的简化版本。

图7是一个流程图，举例说明按照本发明的一个实施例，如何通过从专用的硬件读取控制流量历史信息而操作该可执行代码的简化版本。

图8举例说明按照本发明的一个实施例的用于存储控制流量历史信息的专用硬件。

图9举例说明按照本发明的一个实施例的简化可执行代码的范例部分。

详细说明

给出以下说明书以使得所有的所属技术领域的专业人员能够实现和使用本发明，并且在上下文中提供了特定的应用及其要求。对那些本领域技术人员来说，显然易于对所公开的实施例作各种各样的修改，并且在此处所规定的原理可以适用于其他的实施例和应用，且无需脱离本发明的精神和范围。因此，本发明并非意在受限于所示实施例，而是要求有与在此公开的原理和特点一致的最宽范围。

在这个详细的说明书中描述的数据结构和代码典型地存储于计算机可读存储介质上，该计算机可读存储介质可以是任何存储供计算机系统使用的代码和/或数据的设备或者介质。这包括（但不局限于）磁性和光存储设备，诸如磁盘驱动器、磁带、CD（光盘）和DVD（数字视频磁盘），计算机指令信号包含在传输媒介中（当该信号被调制时有或者没有载波）。例如，该传输媒介可以包括一个通信网络，诸如因特网。

计算机系统

图1举例说明一个按照本发明实施例的计算机系统101。计算机系统101包括一个主要处理器102和一个辅助处理器104。所有的这些构件驻留于硅片100上（虽然在本发明的其它实施例中它们可以驻留于多个硅片上）。

处理器102和104包括指令高速缓存112和120，其分别包括由处理器102和104执行的指令。

处理器102和104还另外包括加载缓冲器114和122以及存储缓冲器116和124，分别用于缓冲与数据高速缓存107和106的通信。更具体地说，主要处理器102包括加载缓冲器114（用于缓冲从数据高速缓存107接收的加载）和存储缓冲器116（用于对数据高速缓存107缓冲存储）。类似地，辅助处理器104包括加载缓冲器122（用于缓冲从数据高速缓存106接收的加载）和存储缓冲器124（用于缓冲对数据高速缓存106所做存储）。

处理器102和104另外通过单向通信信道128～129而连接在一起，单向通信信道128～129便于在主要处理器102和辅助处理器104之间进行快速通信。请注意，通信信道128允许主要处理器102去写入在辅助处理器104内的寄存器126。类似地，通信信道129允许辅助处理器104去写入在主要处理器102内的寄存器127。

与使用共享内存通信机构不同，写入寄存器126(或者127)不会引起高速缓存失误或者相关业务量。此外，主要处理器102没有必要等到辅助处理器104接收到该通信才继续处理。相比之下，如果主要处理器102经由内存与辅助处理器104进行通信，为了通信该系统将必须等候存储缓冲器116被清除。

在本发明的一个实施例中，数据高速缓存106和107是16K字节、带有32字节高速缓存线的4路相关设置数据高速缓存。

数据高速缓存106、数据高速缓存107、指令高速缓存112以及指令高速缓存120经由开关110被连接至二级缓存113。开关110可以包括用于转换信号线路的任何类型的电路。在本发明的一个实施例中，开关110是一个纵横开关。

二级缓存是一个很大的合成一体的高速缓存，用于存储主要处理器102和辅助处理器104两者的指令和数据。二级缓存113被连接至存储控制器111，存储控制器111自身连接至动态随机存取存储器(DRAM)108（其位于芯片之外）。

DRAM 108包含用于主要处理器102的可执行代码130。在系统操作期间，可执行代码130经由存储控制器111、二级缓存113和开关110被

加载进主要处理器102的指令高速缓存112中。DRAM 108还包括用于辅助处理器104的简化的可执行代码132。简化的可执行代码132是可执行代码130的简化版本，其产生与可执行代码130相同的存储参考模式。在系统操作期间，简化的可执行代码132经由存储控制器111、二级缓存113和开关110被加载进辅助处理器104的指令高速缓存120中。DRAM 108另外包含数据134，其经由存储控制器111、二级缓存113和开关110而往返传送于数据高速缓存106～107。

请注意，本发明可以在包含多个处理器的所有计算机系统中使用，并且不局限于举例说明的计算机系统结构。

也请注意，在本发明的一个实施例中，辅助处理器104用于在主要处理器102使用指令之前对该指令进行存取。在这个实施例中，辅助处理器104不必实际执行在程序中指定的所有计算，辅助处理器104只需要执行足够的计算，以允许辅助处理器104去确定该程序的指令存取模式。因此，辅助处理器104可以是主要处理器102的远为简单的版本，除去不是为执行指令地址计算所必需的电路（例如除法器和乘法器电路）。

编译过程

图2举例说明按照本发明一个实施例的编译过程。在该编译过程期间，源代码202通过编译器204馈给，以产生用于主要处理器102的可执行代码130。可执行代码130然后通过简化模块208而馈给，以产生用于辅助处理器104的简化可执行代码132。请注意，简化模块208可以是编译器204的一部分，或者可以是从编译器204分离出来。

在本发明的一个实施例中，在运行时间之前进行该编辑和简化过程。在另一个实施例中，在运行时间期间进行该编辑和简化过程，同时执行该程序。

产生简化的可执行代码

图3是一个流程图，举例说明按照本发明一个实施例而在简化模块208内产生简化的可执行代码132的过程。该系统开始通过对可执行代码130进行程序概要分析（模拟执行）以标识热点，其中内存延迟导致延迟（步骤302）。该系统使用在该程序概要分析过程期间获得的信息而在可执行代码130中建立热点指令踪迹。在本发明的一个实施例中，

该系统构成大约2000个指令的每个大约100个离散的踪迹。该系统然后相关这些踪迹，返回至相应的源代码（步骤304）。

接着，该系统通过产生用以预取该踪迹的代码而产生简化的可执行代码132（步骤306）。在下面参考图4、5和9更详细地描述这个过程。

该系统还将处理器至处理器通信代码插入到可执行代码130和简化可执行代码132两者之中（步骤308）。这个通信代码使主要处理器102去交换进展信息给辅助处理器104。这个进展信息能够使辅助处理器104确定是否它沿着正确路径预取指令、以及是否它提前主要处理器102过远去继续预取指令。

请注意，辅助处理器104提前主要处理器102过远去预取指令是不合乎需要的，因为提前过远而预取的指令很少会为主要处理器102使用，并且可能替换主要处理器102更直接需要的数据。很少可能使用的预取指令会占用存储带宽，并且可能导致高速缓存污染，其降低计算机系统性能。因此，在启动预取指令之前，希望辅助处理器104等待，直至主要处理器102的执行路径接近于在辅助处理器104中相应的代码为止。

也请注意，单个预取操作典型地恢复包括多个指令的整个高速缓存线。例如，一个预取的高速缓存线可以包含64字节，其中包含16个4字节指令。

如果这个进展信息表示辅助处理器104沿着错误的路线通过可执行代码130预取指令，该系统可能引起辅助处理器104弹射出预取指令追踪。

图4举例说明按照本发明一个实施例的用于简化的可执行代码132可能结构。在这个实施例中，简化的可执行代码132被构成为一个循环。该代码首先开始于读取一个变量“val”，该变量“val”可以位于共享内存中或者在图1所示的寄存器126内。如果该val不等于零，该系统基于包含在val中的值而执行一个开关语句，该开关语句执行包含预取操作的代码的特定部分。通过这种方法，主要处理器102可以经由该变量val交换进展信息给辅助处理器104。这个进展信息使特定的预取指令被执行，以便辅助处理器104在主要处理器102需要该指令时提前预取指令。该代码然后重新设置val为零，并且继续该循环。请注意，对于val读和

写不是同时发生的。

图5是一个流程图，举例说明按照本发明一个实施例如何构成该简化的可执行代码132的细节。该系统开始于构成一个用于可执行代码130的控制流程图（步骤502）。然后，该系统从该控制流程图除去小的循环（步骤504），因为小的循环假定被预取进入二级缓存113中。因此，一旦小的循环被预取，其执行直至该循环完成。该系统还从可执行代码130中除去与该控制流程图无关的所有的指令（步骤506）。这可以被做，因为该无关的程序代码不会影响经由程序代码的该执行路径，由此将不会对需要的预取操作有影响。例如，可以除去执行数学计算确定的程序代码。

接着，该系统将预取指令插入到简化的可执行代码132以预取用于可执行代码130的相应部分的指令（步骤508）。

弹射出过程

图6是一个流程图，举例说明按照本发明的一个实施例，当确定已经沿着错误的路线进行的时候，简化的可执行代码132如何弹射出。当在辅助处理器104上执行简化的可执行代码132的时候，该系统开始读取由主要处理器102写入的进展信息（步骤602）。这个进展信息表示主要处理器102的执行路径已经在可执行代码130内到达何处。

如果这个进展信息表示辅助处理器104沿着错误的路线进行，辅助处理器104中断预取该踪迹（步骤604）。辅助处理器可以选择性地开始对应于正确路径的预取。

请注意，在该预取指令过程期间，辅助处理器104可以决定只沿着支路的一边预取，且主要处理器102可以实际上决定沿着该支路的另外一边进行。在这种情况下，辅助处理器104已经采取了错误的路线。

还请注意，辅助处理器104并非一定要弹射出。沿着错误的路线预取指令的唯一的不利结果是高速缓存污染和不必要的存储内容转储，其对于很小的踪迹也许并非重要因素。

用于存储历史信息的专用硬件

图7是一个流程图，举例说明按照本发明一个实施例简化如何操作可执行代码132。该系统开始于从专用的硬件读取控制流程历史信息，该专用硬件记录可执行代码130的控制流程历史信息（步骤702）。例

如，这个控制流程历史信息可以包含一个采用的分支记录和其执行的功能调用。然后，该系统基于该控制流程历史信息构成一个经由可执行代码130推算的路径（步骤704）。例如，该推算的路径可以基于经由可执行代码130采用的上一次的路径确定，或者基于经由可执行代码130采用的最频繁的路径确定。接着，简化的可执行代码132使得辅助处理器104沿着该推算的路径执行预取操作（步骤706）。

图8举例说明按照本发明的一个实施例，用于存储控制流程历史信息的专用硬件。在这个实施例中，该专用硬件存储用于功能调用和返回的调用地址。对于每个调用，该硬件存储一个是否连续的分支被采用或未采用的记录。例如，第一行指定一个驻留在地址“X”的功能调用，继之以一个采用的分支，一个未采用的分支，然后是三个采用的分支。第二行指定一个驻留在地址“Y”的功能调用，然后没有分支。第三行指定一个从功能调用返回至地址“Y+4”。最后，第四行指定一个驻留在地址“Z”的功能调用，继之以二个采用的分支和一个未采用的分支。

简化的可执行代码的范例

图9举例说明按照本发明的一个实施例的简化的可执行代码132的范例部分，其包含预取指令。

图9的左手侧举例说明可执行代码130的部分结构。这个部分被分成多个模块902、904、906、908、910和912，每个模块是64字节长，并且可以在单个预取操作中被预取。请注意，可执行代码130首先执行模块902，其执行一个或者到模块904或者到模块906的分支。模块904和906两者执行对模块908的功能调用。模块908然后执行一个或者到模块910或者到模块912的分支。

图9的右手侧举例说明简化的可执行代码132的部分的相应结构。简化的可执行代码132首先预取模块902、904和906。然后，变量“addr”随着功能调用的目标地址而加载，然后简化的可执行代码132预取模块908、910和912。

请注意，简化的可执行代码132沿着在可执行代码130中分支的两侧预取。在本发明的另一个实施例中，为了消除不必要的指令加载和指令高速缓存污染，简化的可执行代码132只沿着推算的支路预取。

还请注意，该预取操作将可执行代码130模块传送到图1中的二级缓存113。做为选择，该预取操作可以使得被预取的指令直接进入用于主要处理器102的指令高速缓存112。

仅为了说明和描述起见，上文的说明书已经介绍了发明的实施例。说明书并非意图穷举本发明或将本发明限制于所公开的形式。因此，对于本领域专业技术人员来说许多的修改和变化将是显而易见的。另外，以上所述的公开不意在限制本发明。本发明的范围由所附的权利要求限定。

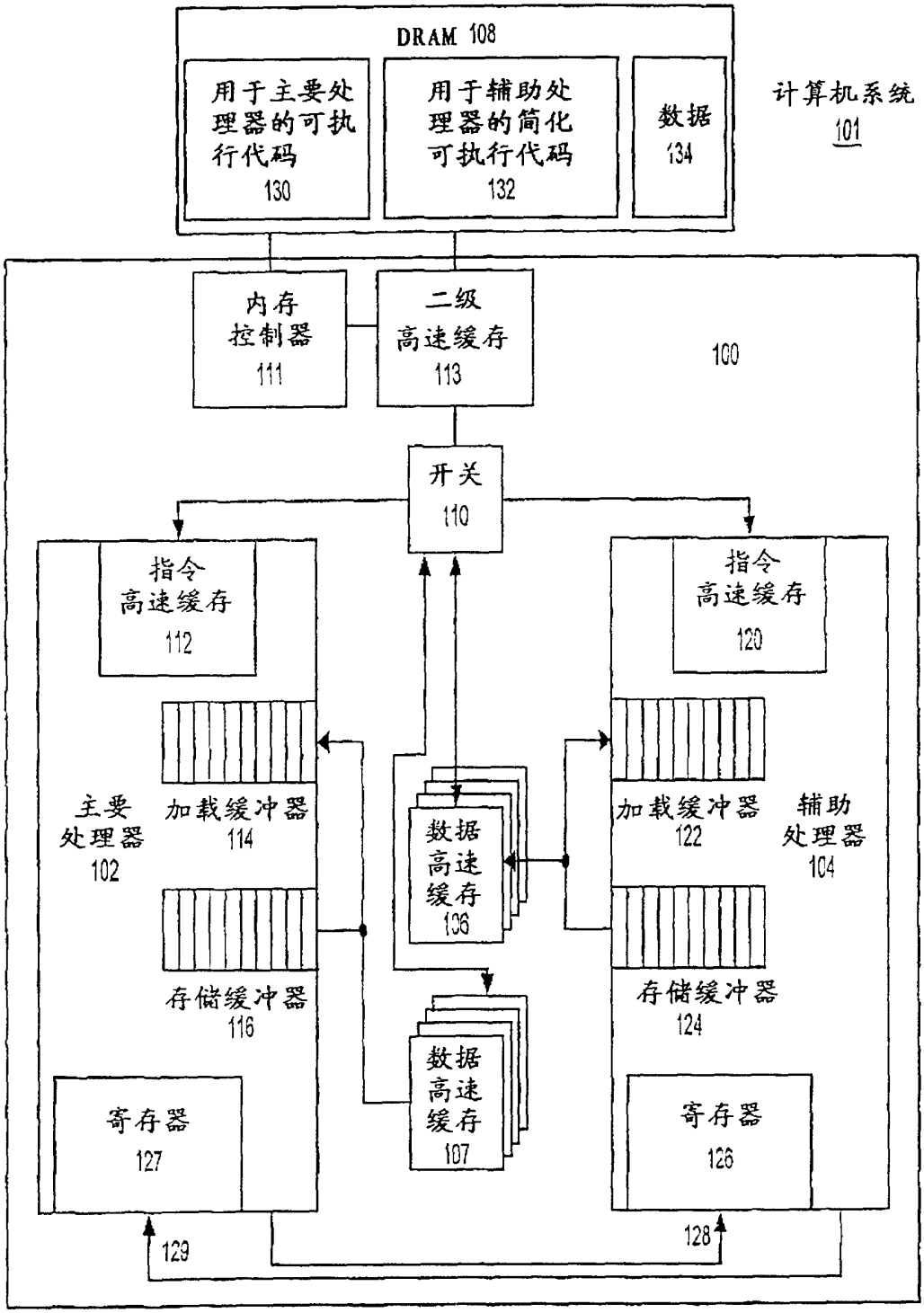

DRAM 108
用于主要处理器的可执行代码 130
用于辅助处理器的简化可执行代码 132
数据 134
计算机系统 101
内存控制器 111
二级高速缓存 113
100
开关 110
指令高速缓存 112
指令高速缓存 120
主要处理器 102
加载缓冲器 114
数据高速缓存 106
加载缓冲器 122
辅助处理器 104
存储缓冲器 116
存储缓冲器 124
数据高速缓存 107
寄存器 127
寄存器 126
129
128

图1

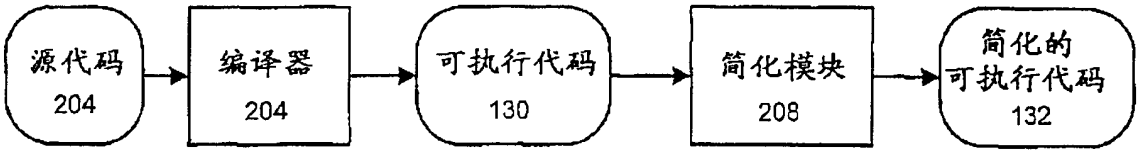

源代码
204
编译器
204
可执行代码
130
简化模块
208
简化的
可执行代码
132

图2

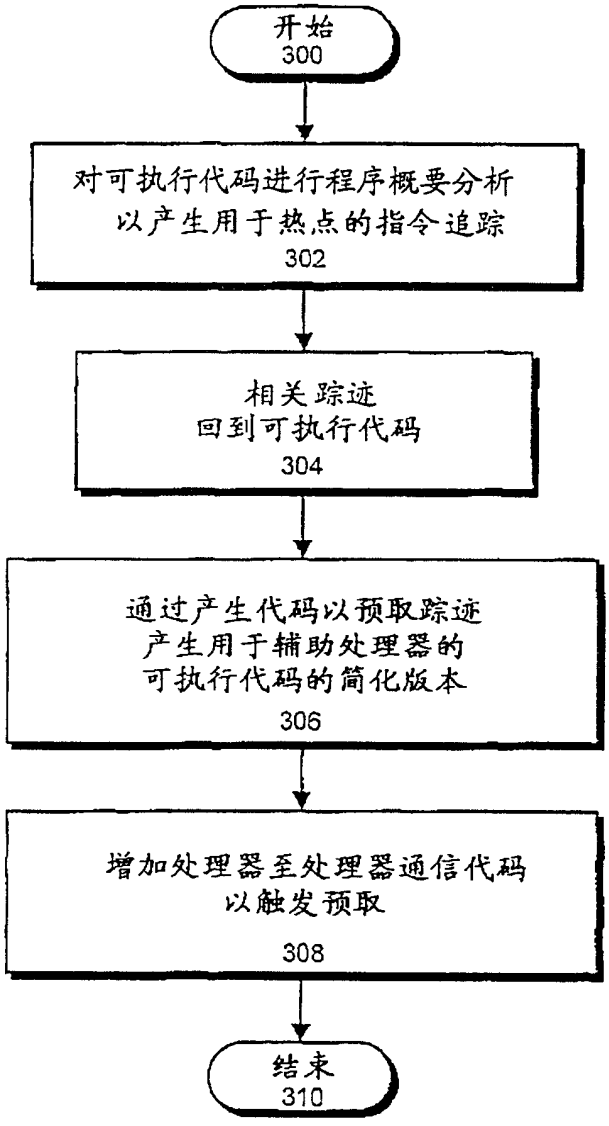

开始
300
对可执行代码进行程序概要分析
以产生用于热点的指令追踪
302
相关踪迹
回到可执行代码
304
通过产生代码以预取踪迹
产生用于辅助处理器的
可执行代码的简化版本
306
增加处理器至处理器通信代码
以触发预取
308
结束
310

图3

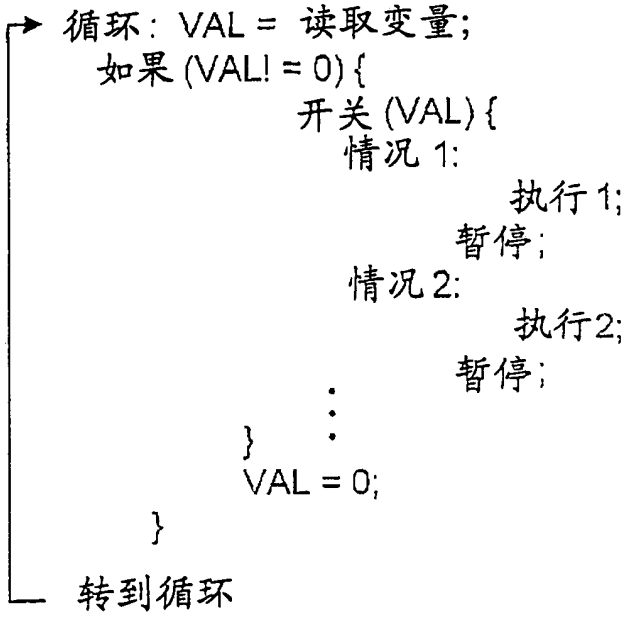


图4

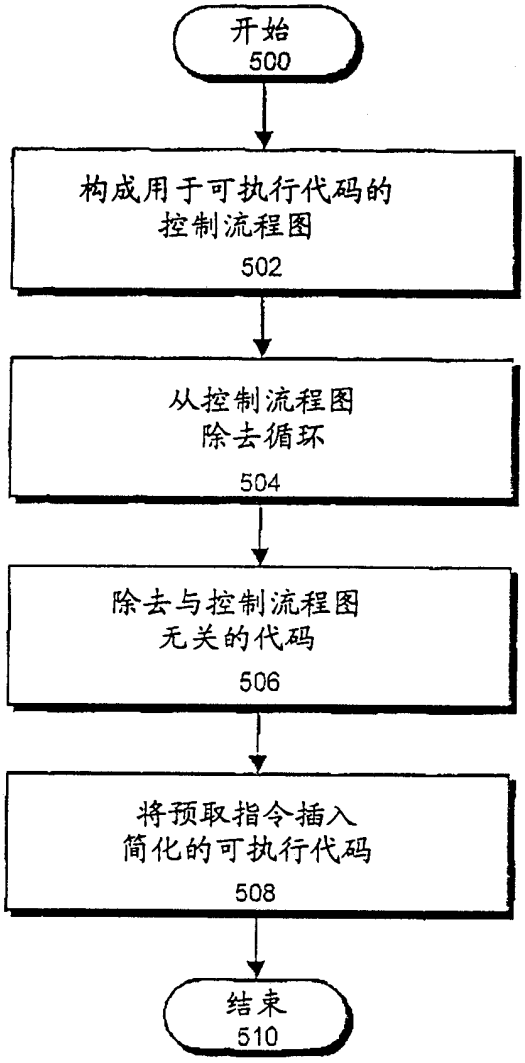

开始
500
构成用于可执行代码的
控制流程图
502
从控制流程图
除去循环
504
除去与控制流程图
无关的代码
506
将预取指令插入
简化的可执行代码
508
结束
510

图5

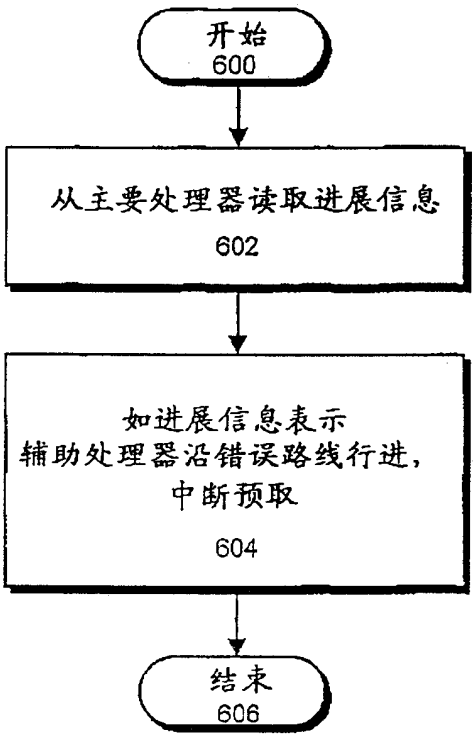

开始
600
从主要处理器读取进展信息
602
如进展信息表示
辅助处理器沿错误路线行进，
中断预取
604
结束
606

图6

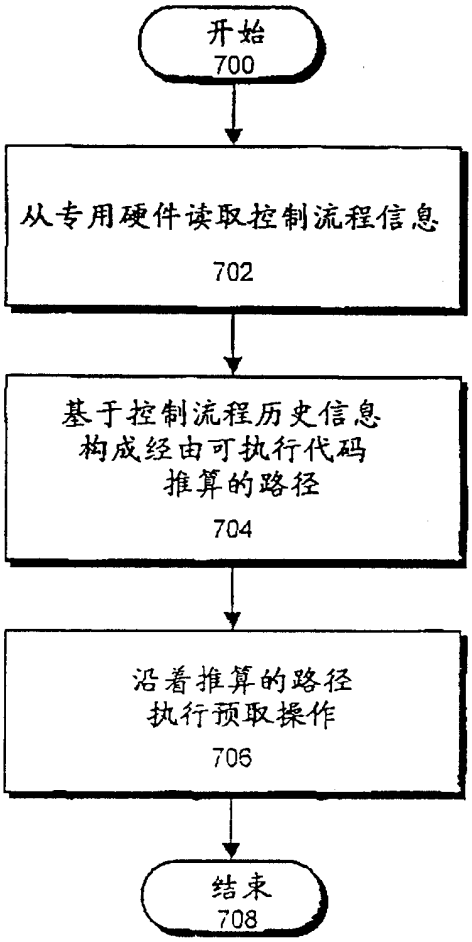

开始
700
从专用硬件读取控制流程信息
702
基于控制流程历史信息
构成经由可执行代码
推算的路径
704
沿着推算的路径
执行预取操作
706
结束
708

图7

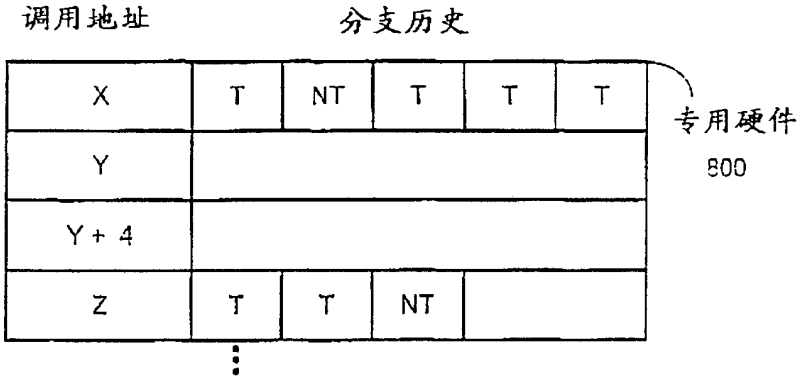

调用地址
分支历史
X
T
NT
T
T
T
Y
Y + 4
Z
T
T
NT
专用硬件
800

图8

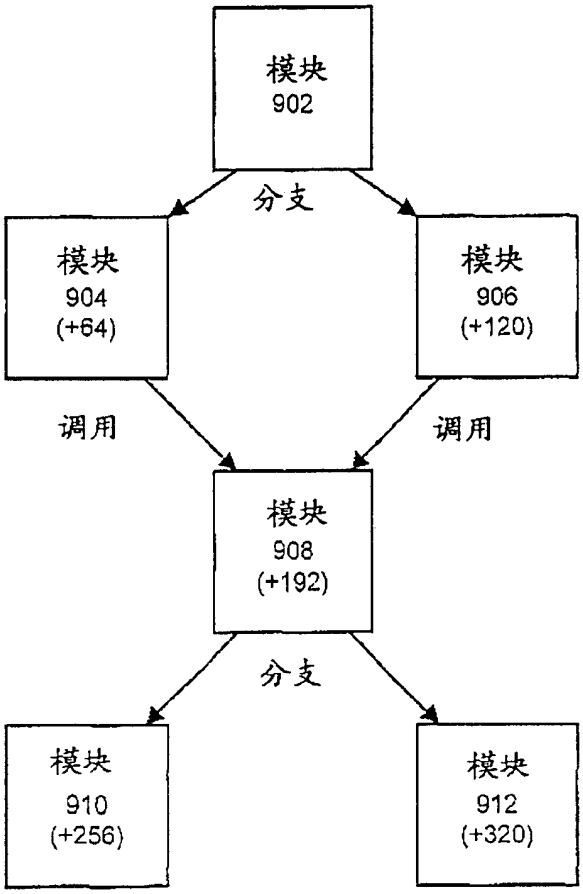

模块
902
分支
模块
904
(+64)
模块
906
(+120)
调用
调用
模块
908
(+192)
分支
模块
910
(+256)
模块
912
(+320)
预取ADDR;
ADDR = ADDR + 64;
预取ADDR;
ADDR = ADDR + 64;
预取ADDR;
ADDR = 调用的地址;
预取ADDR;
ADDR = ADDR + 64;
预取ADDR;
ADDR = ADDR + 64;
预取ADDR;

图9