(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102566977 B
(45)授权公告日 2017.06.30

(21)申请号 201110220091.6

(22)申请日 2011.07.28

(65)同一申请的已公布的文献号
申请公布号 CN 102566977 A

(43)申请公布日 2012.07.11

(30)优先权数据
10-2010-0125688 2010.12.09 KR

(73)专利权人 三星电子株式会社
地址 韩国京畿道水原市

(72)发明人 辛圭桓 曹承模

(74)专利代理机构 北京铭硕知识产权代理有限公司 11286
代理人 韩明星 王艳娇

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

(56)对比文件
US 6192468 B1,2001.02.20,
US 5909567 A,1999.06.01,
US 6088793 A,2000.07.11,
CN 1186981 A,1998.07.08,
TAO LI.Adapting Branch-Target Buffer to Improve the Target.《ACM TRANSACTIONS ON ARCHITECTURE AND CODE OPTIMIZATION》.2005,第2卷(第2期),

审查员 赖女女

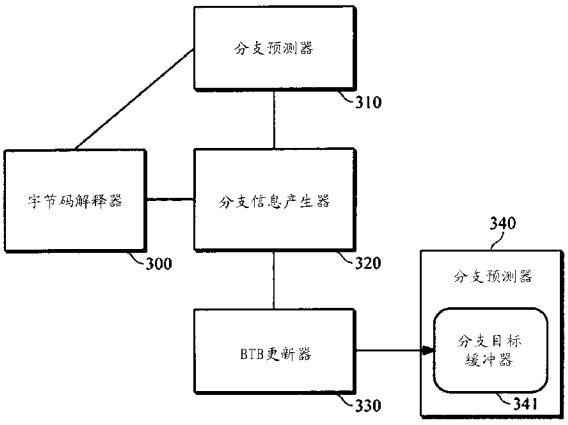
权利要求书2页 说明书7页 附图5页

(54)发明名称

字节码分支处理器及方法

(57)摘要

提供了一种字节码分支处理器及方法。提供了一种计算系统中的字节码解释器。所述解释器通过处理虚拟机(例如,Java和Dalvik)的主处理器来帮助进行分支预测,从而防止分支误预测并实现高性能。



CN 102566977 B

1.一种计算系统中的字节码分支处理器，包括：

解释器，配置为在虚拟机中以字节码格式运行程序；

分支信息产生器，配置为在当预定义数量的字节码在正由解释器处理的当前字节码之前被读取时，通过使用包括根据字节码处理的顺序而映射的先前操作码、当前字节码之前的之前的字节码的长度以及分支地址的表来获得与当前字节码之前的之前的字节码对应的分支的预测路径的分支地址和目标地址，其中，分支地址属于包括在对所述之前的字节码进行处理的先前处理机中的分支码，目标地址属于对先前处理机所分支到的当前字节码进行处理的当前处理机；以及

分支目标缓冲器BTB更新器，配置为用获得的分支地址和目标地址来更新字节码分支处理器中的分支目标缓冲器BTB，

其中，解释器配置为对分支预测进行处理，并通过检查BTB中的获得的分支地址和目标地址在当前字节码的分支预测中是否有效来去除分支误预测。

2.如权利要求1所述的字节码分支处理器，其中，所述解释器还配置为以具有可变长度的字节码的格式运行程序。

3.如权利要求1所述的字节码分支处理器，其中，所述解释器还配置为使用具有用于运行程序的可调用的操作码的处理机来处理字节码。

4.如权利要求3所述的字节码分支处理器，其中，分支码用于根据字节码处理的顺序跳转到具有当前操作码的当前处理机。

5.如权利要求1所述的字节码分支处理器，其中，BTB和BTB更新器配置为在不同的核上运行，并且存储在BTB中的分支信息通过BTB更新器被更新。

6.如权利要求1所述的字节码分支处理器，还包括：

分支预测器，配置为预测字节码中存在的条件分支的路径。

7.一种用于多核处理器的分支预测方法，所述方法包括：

产生指示用于进行处理的当前字节码的虚拟程序计数器vPC值；

产生指示在处于vPC值的当前字节码之前的之前的字节码的预加载vPC值；

通过使用包括根据字节码处理的顺序而映射的先前操作码、所述之前的字节码的长度以及分支地址的表来预测处于预加载vPC值的所述之前的字节码的分支地址和目标地址，其中，分支地址属于包括在对所述之前的字节码进行处理的先前处理机中的分支码，目标地址属于对先前处理机所分支到的当前字节码进行处理的当前处理机；

以预测的分支地址和目标地址来更新分支目标缓冲器BTB；

通过字节码解释器来用分支预测处理当前字节码；

通过检查BTB中的获得的分支地址和目标地址在当前字节码的分支预测中是否有效来去除分支误预测。

8.如权利要求7所述的方法，其中，在多核处理器的第一核上执行当前字节码的处理，并在多核处理器的第二核上执行对分支地址和目标地址的预测。

9.如权利要求8所述的方法，其中，对当前字节码进行处理以及对分支地址和目标地址进行预测分别在第一核和第二核上被同时地执行。

10.如权利要求7所述的方法，还包括：

响应于确定所述之前的字节码是条件分支，预测存在于所述字节码中的条件分支的路

径；

基于与预测路径对应的操作码来搜索映射由计算系统保留的操作码以及分支地址的表。

11.如权利要求7所述的方法，其中，如果计算系统是多核系统，则正在运行BTB的核与更新所述BTB的核被不同地分配。

字节码分支处理器及方法

[0001] 本申请要求于2010年12月9日提交到韩国知识产权局的第10-2010-0125688号韩国专利申请的权益，该申请的全部公开通过引用包含于此，以用于各种目的。

技术领域

[0002] 以下描述涉及一种计算系统中的字节码解释器(interpreter)，更具体地讲，涉及一种用于通过有效地去除在驱动虚拟机的字节码解释器中可能发生的分支误预测或流水线阻塞(stall)惩罚，来提高字节码解释器的性能的分支处理器和方法。

背景技术

[0003] 已经进行了大量的研究来提高虚拟机的性能，以处理Java字节码。尤其是通过即时编译(JITC，Just-in-time compilation)显著地增强了虚拟机的性能。

[0004] 然而，因为资源限制以及对用户敏感的延迟(latency)，嵌入式系统难以有效地引入JITC，因此嵌入式系统以有限的方式利用JITC。此外，由于应用程序的所有代码不按JITC格式被编译，因此解释器的性能仍旧至关重要。

[0005] 直接线程(direct-thread)方法是用于改善解释器的性能的普遍方法之一。在该方法中，在字节码处理机(bytecode handler)的末尾提取下一虚拟指令并直接进行分支。该方法最近已被用于Android Dalvik虚拟机。作为另一方法，ARM Jazelle DBX(直接字节码执行)在硬件中对字节码进行完全地处理。

[0006] 直接线程方法具有以下缺点：作为间接分支指令的分支指令使得实际的处理器(例如，x86或ARM)的分支预测器迷惑，导致大量的分支误预测以及性能方面的恶化。

[0007] 也就是说，在具有普通流水线结构的处理器中，如果发生分支误预测，则已被预测性地(speculatively)执行的所有指令被丢弃，并且处理器需要返回到分支开始的状态。

[0008] 具体地讲，随着高端超标量处理器(例如，ARM cortex A9)被引进到嵌入式装置中，由于分支误预测引起的性能恶化会被加剧。

[0009] 即使在相同的程序计数器(PC)中，在直接线程方法中使用的间接分支指令根据下一虚拟指令也跳转到不同的处理机的地址，因此基于PC的一般分支预测器可能不会正确地工作。为了解决这样的缺点，包括选择性的内联和关联线程技术(context threading)的各种方法被引进，但是引起了代码量增加以及发生调用/返回的开销的缺点。

[0010] 此外，诸如Jazelle DBX的硬件实施方案(一些较不频繁使用的指令被实现为软件处理机)具有高性能，但是需要大量的硬件资源，并且不能处理新型的字节码(例如，Dalvik)。

发明内容

[0011] 在一总体方面，提供了一种计算系统中的字节码分支处理器，包括：解释器，配置为在虚拟机中以字节码格式运行程序；分支信息产生器，配置为从分支的预测路径提前获得分支信息，其中，所述分支存在于在由解释器当前处理的字节码之前的字节码中；分支目

标缓冲器(BTB)更新器，配置为基于获得的分支信息更新计算系统中的分支目标缓冲器(BTB)。

[0012] 所述解释器还可配置为以具有可变长度的字节码的格式运行程序。

[0013] 所述解释器还可配置为使用具有用于运行程序的可调用的操作码的处理机来处理字节码。

[0014] 所述处理机还可配置为包括用于跳转到根据字节码处理的顺序处理字节码的操作码的代码信息。

[0015] 分支信息产生器还可配置为包括映射操作码、字节码的长度以及具有操作码的处理机中的分支代码的地址的表。

[0016] 分支信息产生器还可配置为通过将分支地址和目标地址进行映射来从所述表中产生分支信息，其中，所述分支地址与在由解释器当前处理的字节码之前的字节码的地址对应，所述目标地址与具有处理所述之前的字节码的操作码的处理机中的分支代码的地址对应。

[0017] 分支信息产生器还可配置为通过将从所述表获得的所述之前的字节码的长度与由解释器当前处理的字节码相加来产生分支地址，并基于具有处理所述之前的字节码的操作码的处理机中的分支代码来产生目标地址，其中，从所述表中获得所述分支代码。

[0018] BTB和BTB更新器可配置为在不同的核上运行，并且存储在BTB中的分支信息通过BTB更新器是可更新的。

[0019] 字节码分支处理器还可包括：分支预测器，配置为预测字节码中存在的条件分支的路径。

[0020] 在另一总体方面，提供了一种在计算系统中处理字节码分支的方法，所述方法包括：响应于存在于由虚拟机处理的字节码中的分支的预测路径的存在，使用计算系统的分支预测器更新所述路径；响应于分支预测器预测在由虚拟机当前处理的字节码之前的字节码是分支，确定所述之前的字节码是否是条件分支；响应于确定所述之前的字节码不是条件分支，搜索在保留操作码的计算系统中映射处理所述之前的字节码的操作码和分支地址的表；获得通过将与所述之前的字节码的地址对应的分支地址和与具有处理所述之前的字节码的操作码的处理机中的分支代码的地址对应的目标地址进行映射而产生的分支信息；基于获得的分支信息更新计算系统的分支目标缓冲器(BTB)。

[0021] 映射操作码、字节码的长度以及具有操作码的处理机中的分支代码的地址的表可被预先存储在计算系统中。

[0022] 获得分支信息的步骤可包括：通过将从所述表获得的所述之前的字节码的长度与由虚拟机当前处理的字节码相加来产生分支地址，并基于具有处理所述之前的字节码的操作码的处理机中的分支代码来产生目标地址，其中，从所述表获得分支代码。

[0023] 所述方法还可包括：响应于确定所述之前的字节码是条件分支，预测存在于所述字节码中的条件分支的路径；基于与预测路径对应的操作码来搜索映射由计算系统保留的操作码以及分支地址的表。

[0024] 如果计算系统是多核系统，则正在运行BTB的核可与更新所述BTB的核被不同地分配。

[0025] 在另一总体方面，提供了一种用于多核处理器的分支预测方法，所述方法包括：产

生指示用于进行处理的当前字节码的虚拟程序计数器(vPC)值;以字节码解释器来处理当前字节码;产生指示在当前正被处理的字节码之前的字节码的预加载vPC值;在所述vPC达到预加载vPC的值之前,对当前正被执行的字节码之前的字节码提前预测分支信息;以当前正被执行的字节码之前的字节码的预测的分支信息来更新分支目标缓冲器(BTB)。

[0026] 可在多核处理器的第一核上执行当前字节码的处理,并在多核处理器的第二核上执行对当前正被执行的字节码之前的字节码预测分支信息。

[0027] 第一核和第二核可同时地分别处理当前字节码以及对当前正被执行的字节码之前的字节码预测分支信息。

[0028] 仅仅添加了用于精确分支预测和快速分支解决方案的硬件,直接线程解释器性能可被改善。

[0029] 此外,添加的硬件可有助于字节码处理机的精确分支。

[0030] 分支处理器可实现为软件-硬件混合形式,或仅实现为硬件。

[0031] 即使在硬件实施方案中,也能够以少于硬件解释器(例如,Jazelle)的成本实现分支处理器。

[0032] 此外,分支处理器可支持不同的解释器(例如,Java和Dalvik),并能够处理以多种类型(例如,Dalvik)编写的字节码。

[0033] 分支处理器通过处理虚拟机(例如,Java和Dalvik)的主处理器来帮助进行分支预测,从而防止分支误预测并实现高解释器性能。

[0034] 此外,通过将简单逻辑添加到主机处理器来将分支目标缓冲器(BTB)更新至软件,可防止在直接线程解释器中发生大量的分支误预测。

[0035] 分支目标缓冲器填写机制可实现为除BTB修改逻辑以外的软件或硬件。

[0036] 此外,在软件实施方案中,可利用双核,从而实际的解释器的负荷可被最小化并且分支性能可被增强。

[0037] 从以下详细的描述、附图和权利要求中,其他特点和方面会是清楚的。

附图说明

[0038] 图1是示出虚拟机字节码和直接线程处理机的示例的示图。

[0039] 图2A和图2B是示出操作码-分支地址表和可编程的BTB结构的示例的示图。

[0040] 图3示出字节码分支处理器的示例。

[0041] 图4示出处理字节码分支的示例的流程图。

[0042] 图5示出用于多核处理器的分支预测方法的示例的流程图。

[0043] 贯穿附图和详细的描述,除非另有描述,否则相同的附图标号将被理解为是指相同的部件、特点和结构。为了清楚、例证和方便,可夸大这些部件的相对大小和描绘。

具体实施方式

[0044] 提供以下描述以帮助读者获得对在此描述的方法、设备和/或系统的全面理解。因此,在此描述的方法、设备和/或系统的各种改变、修改和等同物将被建议给本领域的普通技术人员。此外,为了更加清楚和简明,可省略已知功能和构造的描述。

[0045] 图1示出虚拟机字节码和直接线程处理机的示例。

[0046] 示例中示出的处理机(handler)用于在该处理机实现为虚拟机(例如,Java或Android Dalvik虚拟机)中的解释器时有利于处理器有效地处理分支。

[0047] 在图1示出的示例中,提供了具有可变长度的Dalvik字节码。具有固定长度的字节码(例如,Java)可被应用,但是因为可变长度的处理更加复杂,所以采用可变长度字节码Dalvik作为示例。

[0048] 图1示出的示例显示使用直接线程方法在汇编级(assembly level)处理字节码的解释器。

[0049] 图1中示出的示例中显示的解释器包括具有固定最大长度(在本示例中,最大长度为64字节)的字节码处理机。如果具有固定长度的处理机的大小不足以处理函数,则处理机可跳转到处于不同位置的另外的处理例程,并处理剩余的函数。

[0050] 这样,使用直接线程方法在汇编级处理字节码的处理机利用标准分支预测机制。也就是说,分支预测机制包括分支目标缓冲器和分支预测器。

[0051] 然而,可容易地应用不同类型的分支预测机制。

[0052] 图1中示出的示例用于通过实际系统的程序计数器(PC)与字节码的虚拟PC(vPC)之间的交互,来最大限度地使用一般硬件系统的分支预测。

[0053] 首先,调整之前的解释器使用处理机处理字节码。例如,在按照“const/4vA”、“move vA”和“if-ge vA”的顺序提供字节码的情况下,用于“const/4”的处理机被首先调用(100),虚拟PC在处理机中增加,随后检测到字节码是“move vA”,因此解释器跳转到move处理例程(GOTO_OPCODE)。

[0054] 调用用于“move”的处理机(110),随后,解释器转到“if-ge”处理例程,调用用于“if-ge”的处理机(120)。

[0055] 有一般解释器在其中运行的处理器会不可避免地具有多个分支误预测,这是因为所述处理器仅使用GOTO_OPCODE的PC来确定目标地址。

[0056] 图1示出的示例中显示的解释器在分支目标缓冲器(BTB)填写操作的过程中提前加载关于BTB的精确目标地址。然后,实际的硬件预测器被允许参照所述精确目标地址在对应时间来执行精确分支预测。

[0057] 可按照软件方式或硬件方式处理BTB填写。在软件方式的处理中,如果提供多核,则通过使用不同的处理器可减小延迟。

[0058] 详细地,在BTB填写线程(或函数)中,维持单独的虚拟PC值(被称为预加载vPC(preload-vPC))。

[0059] 所述预加载vPC指示在当前处理的vPC之前的字节码,其中,所指示的字节码的分支信息被反映在硬件中。如果vPC值接近预加载vPC,则重新开始BTB填写操作以获取额外的分支信息。

[0060] 图2A和图2B示出操作码-分支地址表和可编程的BTB结构的示例。

[0061] 在建立用于BTB填写操作的解释器的步骤中,操作码-分支地址表必须被提前制作。该表显示在每个操作码处理机中分支的实际位置。

[0062] 如果“move”代码的处理机起始位置是01x64(字节),则用于跳转到下一实际指令的“move”处理机中的GOTO_OPCODE()的地址被写入到表的“分支地址”栏中。此外,为了处理可变长度字节码(例如,Dalvik),每个操作码的长度被写入到“字节码长度”栏中。

[0063] 在BTB填写操作中，从当前vPC或先前扫描的vPC位置开始提前顺序地读取预定义数量的字节码，并且在表中搜索每个读取的字节码。

[0064] 由于在实际的vPC之前执行BTB填写操作，因此BTB填写操作被称为“在前vPC(pre-vPC)”。为了读取继在前vPC之后的字节码，应该读取“字节码长度”。

[0065] 然后，下一字节码地址变为(在前vPC+当前字节码长度)。此外，通过读取表的“分支地址”栏，可获得用于对应字节码处理机执行分支的地址。用于BTB的分支信息由[分支地址，目标地址]对(pair)组成。

[0066] 例如，用于(vPC+第一字节码)的分支信息是[分支地址(vPC+第一字节码)，处理机位置(vPC+第二字节码)]。通过执行k次BTB填写操作，产生k段分支信息。

[0067] 为了实际地反映分支信息，硬件分支预测器应该允许更新分支信息。在软件BTB填写操作的情况下，分支信息可被映射到特定I/O地址或存储器空间，以更新软件中的BTB记录。上述BTB被称为“可编程BTB”。

[0068] 图2B示出的示例中显示的BTB与一般BTB结构几乎相同，但是可以将条目(entry)更新到软件或单独的硬件。

[0069] 图3示出字节码分支处理器的示例。分支处理器可包括字节码解释器300、分支预测器310和340、分支信息产生器320和BTB更新器330。

[0070] 字节码解释器300在虚拟机中按字节码格式运行程序。

[0071] 此外，字节码解释器300可按具有可变长度的字节码的格式运行程序。为了运行程序，使用具有可调用的操作码的处理机来处理字节码。

[0072] 所述处理机可包括用于跳转到操作码的代码的位置信息，其中，所述操作码根据字节码处理的顺序处理所述字节码。

[0073] 分支预测器310可预测在字节码中存在的条件分支的路径。

[0074] 分支信息产生器320可从在由字节码解释器300当前处理的字节码之前的字节码上的分支的预测路径来提前获得分支信息。

[0075] 此外，分支信息产生器320可包括根据字节码处理的顺序映射操作码、字节码长度以及操作码中分支代码的地址的表。

[0076] 此外，分支信息产生器320可通过将当前处理的字节码与从表获得的在前字节码的长度相加来计算下一字节码的地址，并可连续产生分支地址与目标地址的对。

[0077] BTB更新器330可使用获得的分支信息来更新计算系统的分支目标缓冲器(BTB)341。

[0078] 图4示出处理字节码分支的示例的流程图。

[0079] BTB填写机制可具有可以以软件或硬件实现的用于vPC本身的单独的分支预测机制。

[0080] 使用相同的软件逻辑来执行硬件实施方案，因此所述逻辑将被描述为软件。

[0081] 在BTB填写机制中，执行填写直到遇到初始分支，并且当在遇到分支的时刻不存在分支预测时停止填写。

[0082] 当已经经过合适数量(通常，一个或两个)的分支时停止填写。响应于更新分支结果队列，分支预测器被更新，并且在分支误预测的情况下，针对在前的BTB填写操作而已被执行的vPC指针返回到分支误预测点，并从该点继续进行BTB填写。

[0083] 当在分支结果队列中找到分支时，分支预测器被更新(401)，并且执行用于下一字节码的BTB填写操作。

[0084] BTB填写可在单个处理器中通过函数调用而被执行，或者可在双核处理器或多核处理器中被实现为不同线程，或者BTB填写可被实现为硬件逻辑。

[0085] 确定字节码是否是分支结果队列中的分支(402)。

[0086] 如果字节码不是分支，则使用处于在前vPC位置的操作码来搜索操作码-分支地址表(403)。

[0087] 此外，通过读取表的“分支地址”栏，可获得字节码处理机实际分支的地址。基于获得的地址，产生[分支地址，处理机地址]对(404)。

[0088] 为了读取继在前vPC之后的字节码，读取字节码长度。下一字节码的地址为(在前vPC+当前字节码的长度)(405)。

[0089] 将要包括在BTB中的分支信息产生为[分支地址，目标地址]对(406)。为了实际反映产生的分支信息，需要硬件分支预测器来允许分支信息更新。

[0090] 响应于硬件分支预测器允许分支信息更新，在BTB中更新分支信息(406)。

[0091] 此外，在BTB填写操作中，当vPC遇到非条件分支、条件分支、调用-直接(invoke-direct)或调用-虚拟(invoke-virtual)时，查寻用于vPC的软件实现的BTB。BTB可简单实现为2路集关联BTB。

[0092] 如果找到BTB并且分支是条件分支，则可应用一般分支预测器(例如，gShare)(408)。

[0093] 结果，即使在落空(fall-through)或跳转(taken)的情况下，也可获得目标地址，并因此能够获得上述地址对。然而，在vPC分支预测实施方案中，需要检查地址对在字节码的实际分支中是否适当(409)。

[0094] 为此，需要在解释器的字节码处理机中更新实际分支地址。

[0095] 为了防止与vPC预测器的操作(大多，与BTB填写线程相同的线程)冲突，提供共享的原始队列，使得解释器用作产生器，vPC分支预测用作消耗器。

[0096] 也就是说，当判定分支时，解释器将[vPC，目标地址]对放置在共享队列中，vPC分支预测通过检查共享队列来确认预测的分支，并更新分支信息。这与传统的硬件分支预测的更新逻辑相同。

[0097] 这样，通过迭代地更新(406)BTB中的分支信息，做出远先于当先处理的字节码的字节码的分支预测，并完成BTB的更新(407)，分支预测进入睡眠/退让(yield)状态，并且如果在执行字节码时用于在前的字节码的分支预测结果不足，则再次执行从402起的操作。

[0098] 图5示出用于多核处理器的分支预测方法的示例的流程图。

[0099] 参照图5，处理器可产生虚拟程序计数器(vPC)值(501)。所述虚拟程序计数器(vPC)值可指示用于进行处理的当前字节码。

[0100] 如果虚拟程序计数器(vPC)值被产生，则随后处理器可以以字节码解释器来处理当前字节码(502)。在步骤502，可由步骤501的虚拟程序计数器(vPC)指示当前字节码。

[0101] 另外，处理器可产生预加载的虚拟程序计数器(预加载vPC)值(503)。预加载的虚拟程序计数器(预加载vPC)值可指示在步骤502中当前正被处理的字节码之前的字节码。

[0102] 如果预加载的虚拟程序计数器(预加载vPC)值被产生，则随后在虚拟程序计数器

(vPC)达到预加载的虚拟程序计数器(预加载vPC)之前，处理器可对当前正被执行的字节码之前的字节码提前预测分支信息(504)。

[0103] 另外，处理器可使用当前正被执行的字节码之前的字节码的预测的分支信息来更新分支目标缓冲器(BTB)(505)。

[0104] 在多核处理器上执行图5中示出的示例中显示的方法。例如，可在多核处理器的第一核上执行当前字节码的处理，并可在多核处理器的第二核上执行对当前正被执行的字节码之前的字节码预测分支信息。此外，所述第一核和第二核可同时地分别处理当前字节码以及对当前正被执行的字节码之前的字节码预测分支信息。

[0105] 上述方法和/或操作可被记录、存储或固定在一个或多个计算机可读存储介质中，所述计算机可读存储介质包括将由计算机实施的程序指令，以使处理器运行或执行所述程序指令。所述介质还可单独地包括程序指令、数据文件、数据结构等或数据文件、数据结构等与程序指令的组合。计算机可读存储介质的示例包括：磁介质(例如，硬盘、软盘和磁带)；光介质(例如，CDROM盘和DVD)；磁光介质(例如，光盘)以及专门配置用于存储和执行程序指令的硬件装置(例如，只读存储器(ROM)、随机存取存储器(RAM)、闪存等)。程序指令的示例包括(例如，由编译器产生的)机器代码以及包含可由计算机使用解释器执行的更高级代码的文件两者。描述的硬件装置可配置用作一个或多个软件模块以执行上述操作和方法，反之亦然。此外，计算机可读存储介质可分布在通过网络连接的计算机系统中，并且计算机可读代码或程序指令可以以分散方式被存储和执行。

[0106] 以上已描述了多个示例。然而，应该理解，可进行各种修改。例如，如果按不同的顺序执行描述的技术，和/或如果描述的系统、体系结构、装置或电路中的组件按不同的方式组合和/或被其他组件或其等同物代替或补充，则可能实现合适的结果。因此，其他实施方案在权利要求的范围之内。

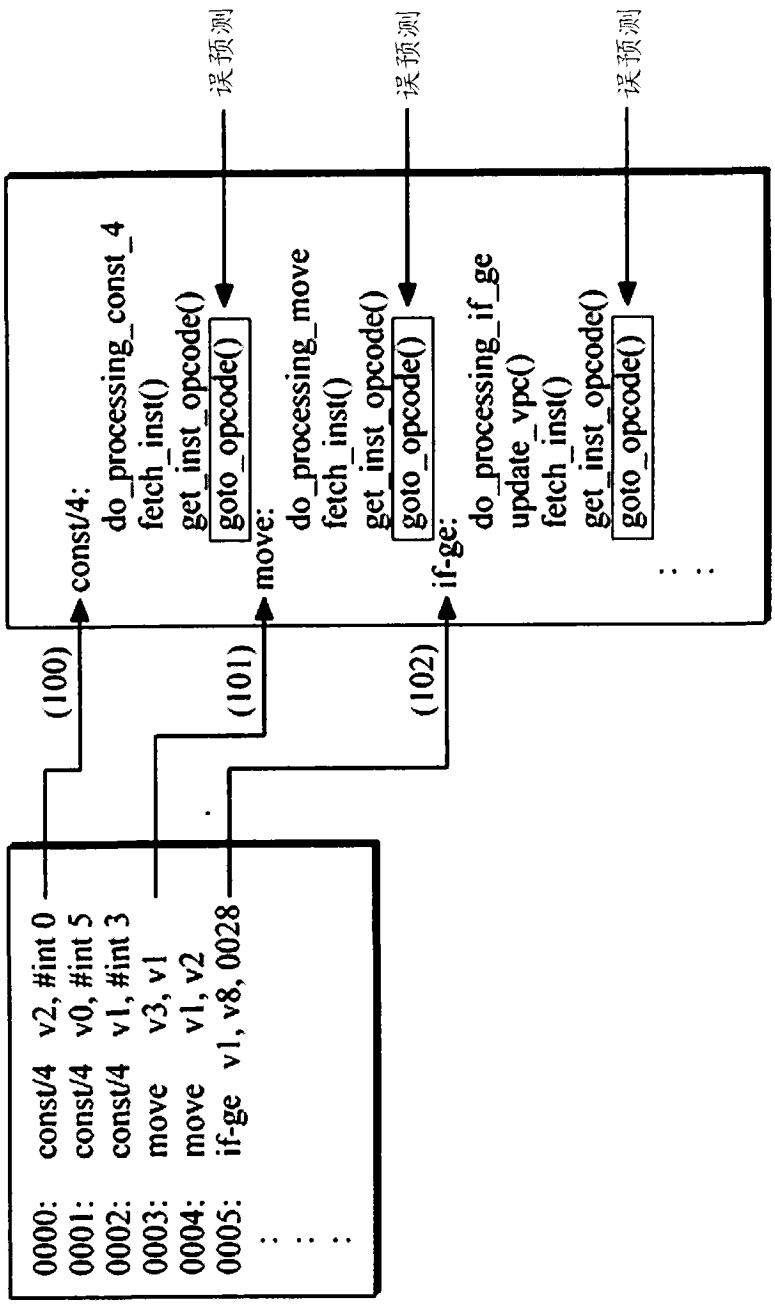

误预测
误预测
误预测
const/4:
do_processing_const_4
fetch_inst()
get_inst_opcode()
goto_opcode()
move:
do_processing_move
fetch_inst()
get_inst_opcode()
goto_opcode()
if-ge:
do_processing_if_ge
update_vpc()
fetch_inst()
get_inst_opcode()
goto_opcode()
(100)
(101)
(102)
0000: const/4 v2, #int 0
0001: const/4 v0, #int 5
0002: const/4 v1, #int 3
0003: move v3, v1
0004: move v1, v2
0005: if-ge v1, v8, 0028

图1

操作码	字节码长度	分支地址
0 (=move)	2	基地址+0*64+28
1	6	基地址+1*64+36
2	4	基地址+2*64+32
3 (=if-ge)	2	基地址+3*64+48

图2A

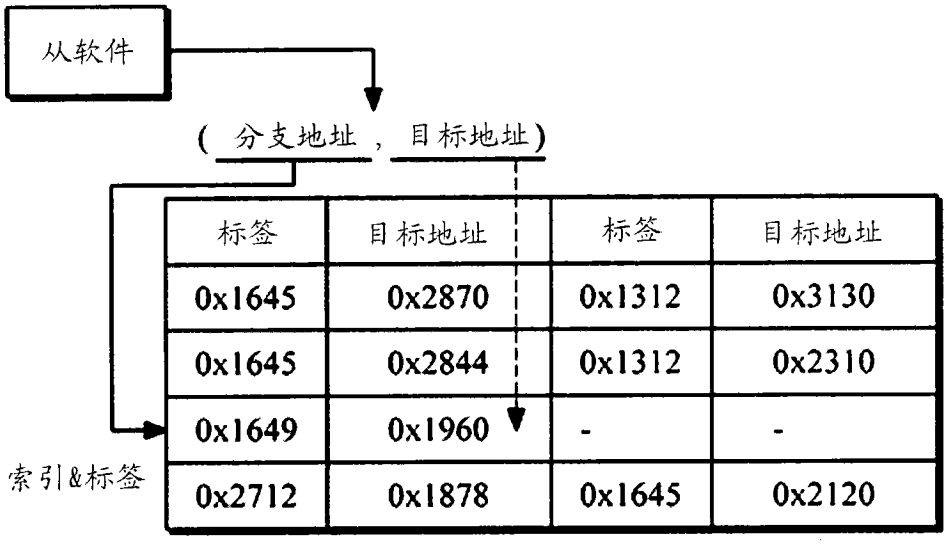


图2B

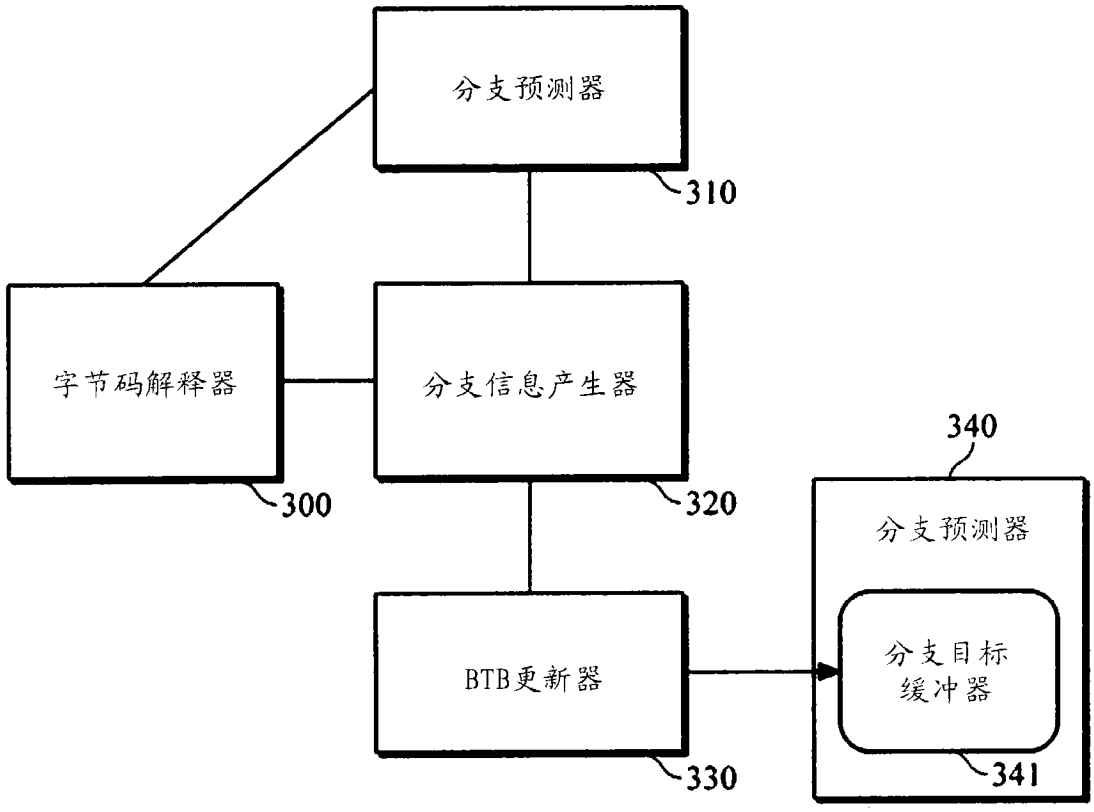

分支预测器
310
字节码解释器
300
分支信息产生器
320
340
分支预测器
分支目标
缓冲器
341
BTB更新器
330

图3

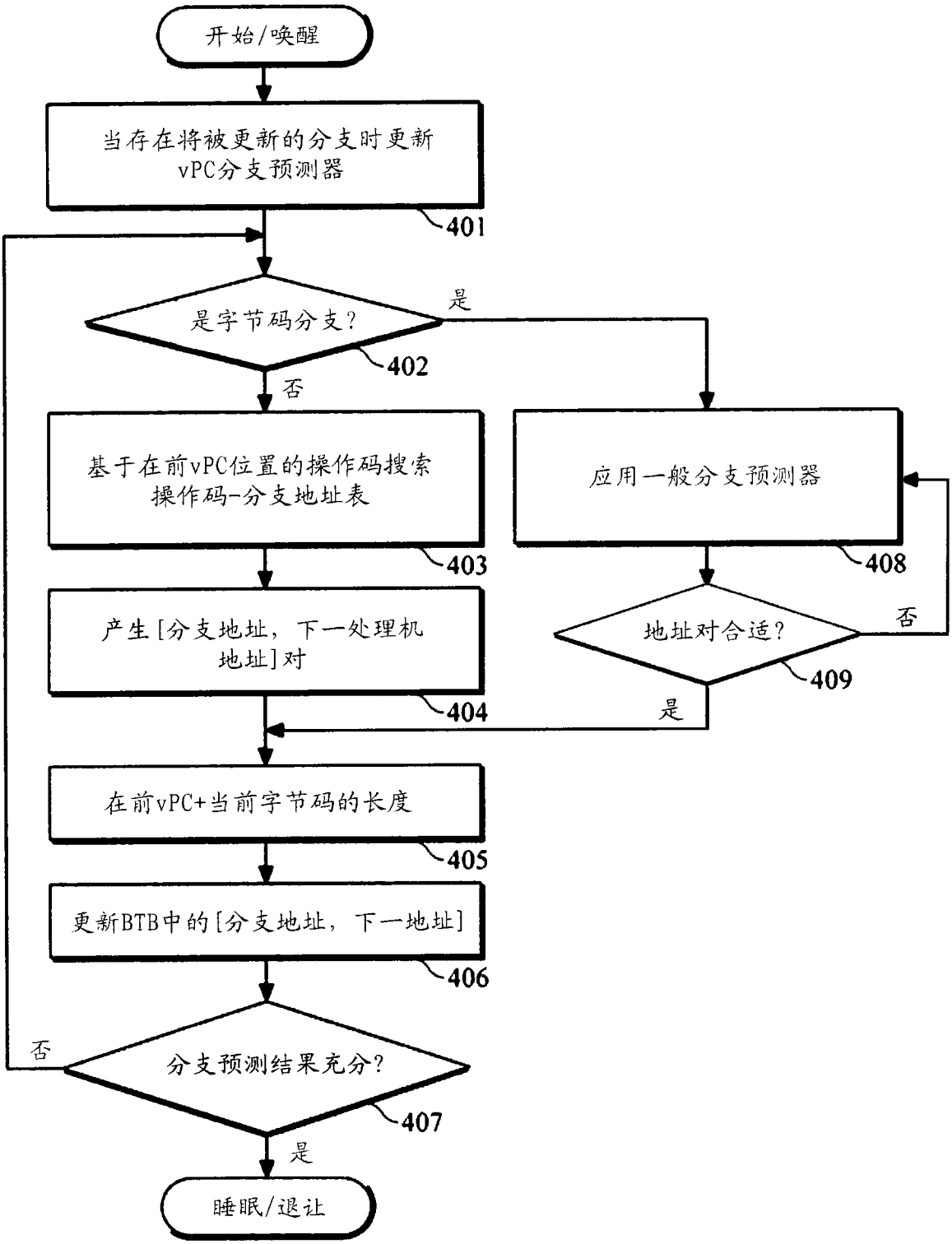

开始/唤醒
当存在将被更新的分支时更新vPC分支预测器
401
是字节码分支？
402
是
否
基于在前vPC位置的操作码搜索操作码-分支地址表
403
产生[分支地址，下一处理机地址]对
404
应用一般分支预测器
408
地址对合适？
409
否
是
在前vPC+当前字节码的长度
405
更新BTB中的[分支地址，下一地址]
406
分支预测结果充分？
407
否
是
睡眠/退让

图4

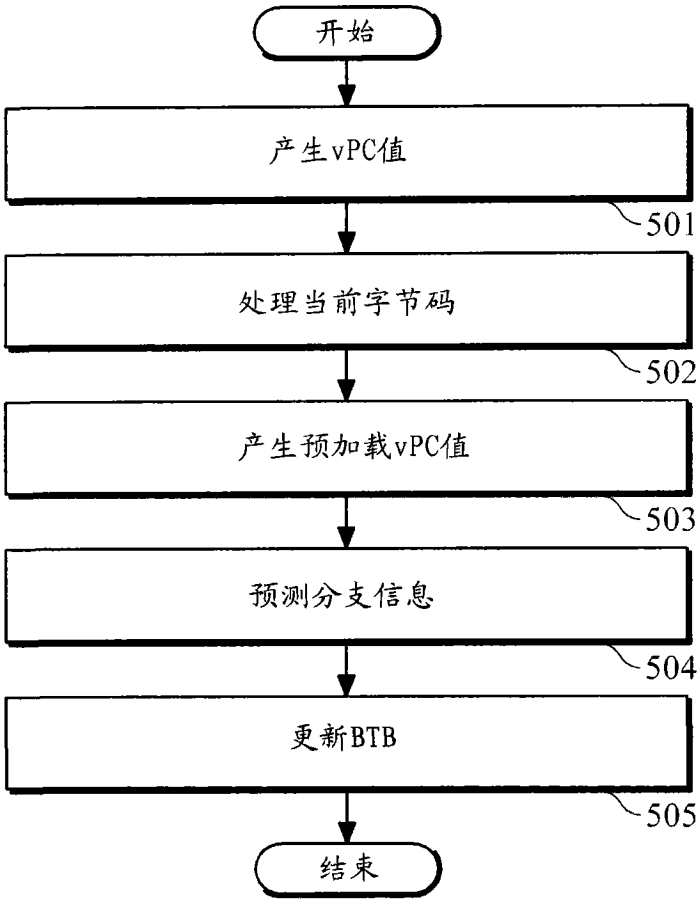

开始
产生vPC值
501
处理当前字节码
502
产生预加载vPC值
503
预测分支信息
504
更新BTB
505
结束

图5