(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 103914279 A
(43)申请公布日 2014.07.09

(21)申请号 201310743131.4
(22)申请日 2013.12.30
(30)优先权数据
13/730,407 2012.12.28 US
(71)申请人 辉达公司
地址 美国加利福尼亚州
(72)发明人 吉列尔莫·J·罗扎斯
亚历山大·克莱贝尔
詹姆斯·范·策恩 保罗·塞维斯
布拉德·霍伊特
斯里达兰·罗摩克里希纳
亨斯·凡德斯库特 罗斯·泽格尔肯
达雷尔·D·博格斯
马格努斯·埃克曼
阿温达哈·巴克他 戴维·邓恩
(74)专利代理机构 北京市磐华律师事务所 11336
代理人 谢栒 魏宁
(51)Int.Cl.
G06F 9/30(2006.01)

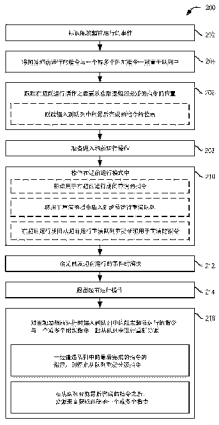
权利要求书1页 说明书10页 附图6页

(54)发明名称

被排队的指令在超前运行之后的重新分派

(57)摘要

被排队的指令在超前运行之后的重新分派。本文公开微处理器和在超前运行操作期间操作微处理器的方法的各实施例。操作微处理器的一个示例方法包括标识与触发超前运行的指令相关联的触发超前运行的事件，以及响应于触发超前运行的事件的标识，进入超前运行操作并将触发超前运行的指令与一个或多个附加指令一起插入队列。示例方法还包括响应于触发超前运行的事件的解决而恢复微处理器的非超前运行操作以及将触发超前运行的指令与一个或多个附加指令一起从队列重新分派到执行逻辑。



CN 103914279 A

1. 一种微处理器，包括：

调度器逻辑，用于发出一个或多个指令用于执行；

执行逻辑，用于执行由所述调度器逻辑所发出的所述一个或多个指令；

队列，用于保持一个或多个指令用于到所述执行逻辑的重新分派；以及

超前运行控制逻辑，用于管理所述微处理器的超前运行操作，所述超前运行控制逻辑配置为：

检测触发超前运行的事件，

响应于检测到所述触发超前运行的事件，(i) 使所述微处理器操作在超前运行中，并且 (ii) 将与所述触发超前运行的事件相关联的触发超前运行的指令与一个或多个附加指令一起插入到所述队列中，

一经解决所述触发超前运行的事件则恢复非超前运行操作，以及

一经恢复非超前运行操作，则将所述触发超前运行的指令和所述一个或多个附加指令从所述队列重新分派到所述执行逻辑。

2. 根据权利要求 1 所述的微处理器，其中所述调度器逻辑将所述一个或多个指令从所述队列重新分派到所述执行逻辑。

3. 根据权利要求 1 所述的微处理器，进一步包括分配指针，所述分配指针配置为指示用于被选择用于插入到所述队列中的指令的插入位置。

4. 根据权利要求 1 所述的微处理器，进一步包括读指针，所述读指针配置为指示所述队列中的被读取用于重新分派的位置。

5. 根据权利要求 4 所述的微处理器，其中所述读指针进一步配置为一经重新分派存储在其中的指令，则释出所述被读取用于重新分派的位置。

6. 根据权利要求 1 所述的微处理器，进一步包括超前运行重演队列，所述超前运行重演队列配置为保持被标识用于在超前运行期间重演的一个或多个指令。

7. 根据权利要求 1 所述的微处理器，进一步包括：

跟踪指针，其配置为跟踪插入到所述队列中的最后完成的指令的位置；以及

重新启动指令指针，其配置为跟踪在重新分派所述最后完成的指令之后要从获取逻辑所分派的指令的位置。

8. 根据权利要求 7 所述的微处理器，其中所述重新启动指令指针配置为响应于所述跟踪指针的更新而被更新。

9. 根据权利要求 1 所述的微处理器，其中所述队列配置为保持被标识用于在非超前运行操作期间重演的一个或多个指令。

10. 根据权利要求 1 所述的微处理器，其中将所述触发超前运行的指令与所述一个或多个附加指令一起插入包括在所述微处理器在超前运行操作中的同时冻结所述队列。

被排队的指令在超前运行之后的重新分派

背景技术

[0001] 微处理器中的指令可引起停顿，这潜在地延迟指令处理并绑定微处理器资源。因为停顿通常不可预测，一些解决第一停顿的微处理器可能推测性地执行指令以揭示其他潜在的停顿。虽然如果潜在停顿被揭示则推测性执行可加速执行，但每次实施推测性执行时，在推测性执行结束后重新启动微处理器并将微处理器重置到预推测性执行状态可能减慢执行。

附图说明

[0002] 图 1 示意性地示出根据本公开的实施例的示例微处理器。

[0003] 图 2 示出根据本公开的实施例的、说明操作微处理器的示例方法的流程图。

[0004] 图 3 示出用来说明图 4 和 5 中所描绘的方法的示例程序。

[0005] 图 4 示意性地示出根据本公开的实施例的、在超前运行（runahead）中操作微处理器的方法。

[0006] 图 5 示意性地示出根据本公开的实施例的、在超前运行中操作微处理器的另一方法。

[0007] 图 6 示意性地示出根据本公开的实施例的另一示例微处理器。

[0008] 图 7 示意性地示出根据本公开的实施例的另一示例微处理器。

具体实施方式

[0009] 在现代微处理器中，指令常常在管线中执行。可以将这类指令单独地或作为微操作束（bundle）分派到管线内的各执行级。无论发出用于执行时指令的形式如何，当发出指令时，在分派时可能并不知道在指令的执行期间是否将发生未命中或异常。在微处理器能够对后续指令取得进展之前需要一些时间以解决一些未命中 / 异常。因此，微处理器在尝试解决长时延事件的根本原因的同时可能停顿。常见管线停顿的一个非限制性示例是导致高速缓存未命中的加载操作。

[0010] 在一些微处理器中，停顿可能触发进入配置为检测其他潜在停顿的操作的超前运行模式中的入口。换句话说，微处理器可检测可能使微处理器停顿的长时延事件。在尝试解决该长时延事件（例如触发超前运行的事件）的同时，微处理器可推测性地执行附加指令以尝试揭示其他可能的停顿。通过揭示其他可能的停顿，微处理器可开始解决作为那些可能的停顿的基础的长时延事件，同时解决触发超前运行的事件，这潜在地节约时间。

[0011] 如本文所使用的，超前运行操作描述产生自长时延事件的任何合适的推测性执行方案，并配置为揭示可能导致停顿的一个或多个其他潜在长时延事件。这类长时延事件的非限制性示例包括高速缓存未命中（例如存储未命中和 / 或加载未命中）、转译后备缓冲区（例如指令和 / 或数据转译后备缓冲区）中的未命中、和一些长时延浮点运算（例如非正规（denormal）平方根运算）。

[0012] 一旦检测到触发超前运行的事件，微处理器的状态（例如寄存器值和其他合适的

状态）可被设置检查点，使得一旦已解决触发超前运行的事件并且超前运行操作结束，则微处理器可返回到超前运行前（pre-runahead）状态。设置检查点保存微处理器的当前状态，这允许以后恢复到这类状态。设置检查点可包括例如拷贝寄存器的内容以复制寄存器。在超前运行操作期间，微处理器在工作状态中执行，但不提交指令的结果以避免更改微处理器的状态。提交更新微处理器状态并可包括例如覆写被设置检查点的寄存器。在触发超前运行的事件被解决后，微处理器退出超前运行并重新启动。如本文所使用的，重新启动微处理器是指返回到被设置检查点的超前运行前状态，使得正常的非超前运行操作恢复。在超前运行之后重新启动微处理器是指在指令流中的超前运行前位置处重新启动指令的执行，使得该指令的执行在重新启动后继续，犹如超前运行从未发生过，尽管采取了各种动作以对在超前运行期间所揭示的潜在长时延事件（例如预获取未命中等）进行解决。

[0013] 典型地，重新进入非超前运行操作涉及随着对执行触发超前运行的指令的较早尝试不成功，重新分派触发超前运行的指令用于执行。然而，从指令高速缓存或统一高速缓存重新获取指令可能增加重新进入非超前运行执行的时延。因此，本文公开了与微处理器和操作微处理器的方法相关的各实施例。在一个示例中，在超前运行操作结束之后从队列重新分派触发超前运行的指令。从队列重新分派触发超前运行的指令而不是对其加以重新获取可节省可能另外花费在获取指令以及在一些设定中解码指令的时间。

[0014] 例如，在一些实施例中，在超前运行之外的普通操作期间具有第一目的的队列（例如重演队列）可在超前运行期间用于另一目的。具体来讲，队列可重新用来在超前运行期间保持触发超前运行的指令和一个或多个附加指令。在一些实施例中，重演队列可在超前运行期间用于该目的。在根据这类实施例的一个场景中，重演队列可用来在正常操作期间重演超前运行之外的指令。当遭遇触发超前运行的指令时，该指令与一个或多个附加指令一起被添加到重新目的化（repurposed）的重演队列。采用该场景继续，一旦那些指令被添加并且进行超前运行，则经重新目的化的重演队列被冻结。一经从超前运行退出，则从经重新目的化的重演队列分派触发超前运行的指令和附加指令，这如上文所述在超前运行之后潜在地加速处理。

[0015] 图 1 示意性地示出可与本文所描述的系统和方法有关所采用的微处理器 100 的实施例。微处理器 100 包括配置为存储指令的存储器系统 102 和配置为处理指令的管线 104。关于存储器系统 102 的附加细节在图 7 中加以说明并在附随文本中加以描述。

[0016] 存储器系统 102 可配置为存储任何合适类型的指令。例如，指令可以以指令集架构（ISA）指令、微操作（以合适的束化形式或非束化）的形式来存储、存储为相关 ISA 指令的转译等等。在一些示例中，单独的微操作可与一个或多个指令或指令的一部分相对应。换句话说，在一个场景中单个指令可存储为微操作束，而在另一场景中多个指令可存储为微操作束。如本文所使用的，束是指成组在一起的一个或多个微操作。

[0017] 图 1 中以简化形式示出的管线 104 在程序执行期间处理存储在存储器系统 102 中的一个或多个指令。管线指令处理可允许多于一个指令并发地处于在检索和执行的不同级中。换言之，被包括在管线 104 中的各级可允许一些指令准备好用于由上游级执行而下游级执行其他指令并收回另一些其他指令。

[0018] 图 1 中示出的示例管线 104 包括获取逻辑 106、解码逻辑 108、调度器逻辑 110、执行逻辑 112 和回写逻辑 114。将理解的是，仅出于例示性目的提供图 1 中示出的管线 104

的实施例中所显示的逻辑部分，并且可以以任何合适的方式布置下文更详细地描述的功能性。例如，管线 104 的一些实施例可包括执行单元中的一个或多个内的经单独管线化的部分(例如获取逻辑 106、解码逻辑 108 等中的一个或多个可包括经单独管线化的部分)，而一些实施例可将单个管线执行单元内的两个或更多个逻辑部分的部分加以组合。

[0019] 在图 1 中示出的实施例中，获取逻辑 106 从存储器系统 102 检索指令，典型地是从由 L2-L3 高速缓存和主存储器所支持的统一或专用 L1 高速缓存检索指令。解码逻辑 108 例如通过解析操作符 / 操作码、操作数和寻址模式来解码指令。一经被解析，指令就随后由调度器逻辑 110 所调度用于由执行逻辑 112 执行。执行逻辑 112 可包括配置为执行由调度器逻辑 110 所发出的指令的一个或多个执行级(例如执行级 0 到 4 在图 1 中示出)。任何合适数目和类型的执行级可包括在执行逻辑 112 内。执行逻辑 112 还可与存储器系统 102 通信(例如经由被包括在执行逻辑 112 中的一个或多个加载 / 存储单元)以处置在操作期间所遭遇的加载和 / 或存储操作。一旦由执行逻辑 112 处理，可存储所完成的指令以准备用于由回写逻辑 114 提交。回写逻辑 114 通过将所完成的指令的结果提交到微处理器寄存器、高速缓存和 / 或存储器来更改微处理器 100 的架构状态。换句话说，回写逻辑 114 针对微处理器 100 实施提交功能性。

[0020] 在图 1 中示出的实施例中，微处理器 100 包括超前运行控制逻辑 116。超前运行控制逻辑 116 针对微处理器 100 控制进入和退出超前运行模式。在一些实施例中，超前运行控制逻辑 116 还可控制与进入和退出超前运行相关的其他操作。例如，当进入超前运行时，微处理器 100 的部分可被设置检查点以留存微处理器 100 的状态，同时微处理器 100 的未被设置检查点的(non-checkpointed)工作状态版本在超前运行期间推测性地执行指令。在一些这类实施例中，一经从超前运行退出，超前运行控制逻辑 116 就可将微处理器 100 还原到被设置检查点的状态。

[0021] 微处理器 100 还包括队列 118，其配置为保持要由调度器逻辑 110 重新分派或“重演”的一个或多个指令。如本文所使用的，指令重演是指指令被重新分派 / 重新发出用于执行而不用被重新获取和 / 或重新解码。例如，队列 118 可用来响应于管线不连续性而存储被选择以重演的指令，所述管线不连续性使该指令在初始分派之后无法完成。可重演指令一次或多次直到指令达到完成状态为止。

[0022] 在正常的非超前运行操作期间，被选择用于重演的指令可传递入和传递出队列 118，使得队列 118 将要重演的指令重新循环到执行逻辑 112。在一些实施例中，该指令一经分派则从队列 118 中释出指令。随着指令被分派而排空指令的队列 118 可释放队列 118 中的空间，使得可添加被选择用于重演的新指令，并还可维护指令流流畅。然而，引起重演的事件也可以是触发超前运行的事件；因此先于进入超前运行来排空队列 118 将一经重新进入非超前运行操作就使获取逻辑 106 重新获取该指令。在其他实施例中，指令可以不是一经分派就从队列 118 中释出。反而，指令可维持在队列 118 中直到这些指令的执行完成并且不能实施重新分派为止。在该实施例中，指令将不需要被重新插入到队列中用于设置检查点或超前运行。

[0023] 因此，在一些实施例中，可一经检测到触发超前运行的事件就采用触发超前运行的指令来填充队列 118。通过将触发超前运行的指令插入到队列 118 中，该指令在超前运行结束后将可用于重新分派。当进入超前运行时可冻结队列 118，使得触发超前运行的指令一

经从超前运行退出就将可用于分派而无论在超前运行期间发生什么。将理解的是，在一些实施例中，冻结的队列 118 可被包括在设置检查点的微处理器 100 中以准备用于进入超前运行，而在一些其他实施例中，其可以是分开的进程或者完全不实施。

[0024] 此外，一个或多个附加指令可与触发超前运行的指令一起插入到队列 118 中。将附加指令插入到队列 118 中可节省可能另外花费在当微处理器在超前运行之后恢复正常操作时获取和解码那些指令上的时间。换句话说，与重新获取触发超前运行的指令或将其（例如从被设置检查点的状态）重新发出相反，将触发超前运行的指令和一个或多个附加指令插入到队列 118 中可允许在超前运行终了并且微处理器恢复非超前运行操作之后那些指令从队列 118 到执行逻辑 112 的快速重新分派。

[0025] 虽然为了清楚起见在图 1 中以线性方式示出队列 118，但将理解的是队列 118 可具有任何合适的形式。例如，队列 118 可布置为循环队列。进一步地，可以以任何合适的方式从队列 118 插入和分派指令。例如，可使用“先入先出”格式从队列 118 插入和分派指令。在另一示例中，可根据从初始分派开始的相对寿命来分派指令，该相对寿命可以与队列内的寿命不同。队列 118 还可具有任何合适的大小。在一个非限制性示例中，队列 118 可包括 16 个条目。

[0026] 此外，在一些实施例中，在正重新分派保持在队列 118 中的指令的同时，获取逻辑 106 可并发地检索一个或多个后续指令，并且解码逻辑 108 可解码那些后续指令中的一个或多个。反过来，用于新获取的指令的检索和解码时间可与用于被排队的指令的重新分派过程重叠。对后续指令的检索和假如适当的话进行解码可引起被获取 / 被解码的后续指令随时可用于由调度器逻辑 110 进行的分派。反过来，微处理器 100 可经历从来自队列 118 的指令的重新分派到新获取的指令的分派的几乎无缝的转变。

[0027] 图 2 示出用于操作微处理器的方法 200 的实施例的一部分的流程图。将理解的是，可以通过包括本文所描述的硬件的任何合适的硬件来实施方法 200。在一些实施例中，一个或多个软件模块可实施方法 200 的各方面。进一步地，将理解的是，图 2 中示出的和下文所描述的方法 200 的实施例呈现为用于讨论目的的示例。图 2 中描述的过程中的任何一个可通过其他合适的过程加以补充、省略和 / 或合适地重新排序而不脱离本公开的范围。

[0028] 在 202，方法 200 包括标识触发超前运行的事件。在一些实施例中，可由合适的超前运行控制逻辑标识触发超前运行的事件，但应理解的是微处理器的任何合适的部分可检测这类事件。例如，被包括在微处理器执行逻辑中的加载 / 存储执行级可检测高速缓存未命中或存储未命中；浮点执行级可检测非正规平方根运算等。任何合适的长时延事件可视为触发超前运行的事件。这类长时延事件可在微处理器设计期间被预确定、在微处理器操作期间来动态地被确定等等。

[0029] 在 204，方法 200 包括在队列中插入触发超前运行的指令。一个或多个附加指令可与触发超前运行的指令一起插入到队列中。例如，图 1 示出在超前运行操作的开始处的队列 118。队列 118 包括三个指令：指令 A、指令 B 以及指令 C，指令 A 包括两个束（示出为束 A1 和束 A2），指令 C 包括多于一个束（示出为束 C1）。在图 1 中示出的示例中，在执行逻辑 112 的级 2 处所检测到的触发超前运行的事件导致相关联的触发超前运行的指令（指令 A）经由路径 120 如所示被插入队列 118。在执行逻辑 112 中跟随指令 A 的指令 B 和指令 C 的束 C1 也被置入队列 118。将理解的是，任何合适数目的附加指令可与触发超前运行的指

令一起插入到队列 118 中。在一些实施例中，可基于队列 118 的大小、用于从存储器系统中所检索的指令的典型检索时间和 / 或解码时间等等来选择被选择用于插入到队列 118 中的指令的数目。换句话说，被选择用于插入的指令的数目可在设计阶段期间被预确定和 / 或在微处理器操作期间动态地被确定。

[0030] 将理解的是，可在处理期间的任何合适的点或位置处将指令插入到队列 118 中。微处理器可具有配置为将指令路由到队列中的多个路径。例如，图 1 描绘从级 2 延伸到队列 118 的插入路径 120，并描绘从级 4 延伸到队列 118 的另一插入路径 122。虽然任何合适数目的插入路径均是可能的而不脱离本公开的范围，但在一些实施例中，可选择将指令从可能检测到触发超前运行的事件的级路由到队列的路径。例如，插入路径可将加载 / 存储级与队列链接。如另一示例，插入路径可将最后级与队列链接来作为在指令退出管线之前收集那些指令的方式。在一些实施例中，超前运行控制逻辑可管理到队列中的指令的选择和 / 或插入。在一些实施例中，一个或多个执行级可管理到队列中的指令的选择和 / 或插入。在一些其他实施例中，到队列中的指令的选择和插入可分布在微处理器的各部分当中。

[0031] 在一些实施例中，可在由分配指针所指示的位置处插入被选择用于插入到队列中的指令。分配指针指示下一指令要插入在队列中的何处。例如，图 1 示出示例分配指针 124，其指示用于将另一指令插入到队列 118 中的位置（示出在条目 126A 处）。当指令插入到队列中时分配指针 124 被更新。因此，在紧挨在指令 C 的束 C1 到队列 118 中的插入之前的时间，分配指针 124 置于条目 126B。

[0032] 当在超前运行之后重新启动 / 重新进入正常操作时，获取逻辑在队列中的最后指令被重新分派之后被引导到指令指针，使得在超前运行之前插入到队列中的最后指令被重新分派之后处理继续。指令指针可按顺序跟随队列中的最后指令，或者反之，诸如在采取分支的情况下。因此，在 206，方法 200 包括跟踪在超前运行操作之后要从获取逻辑所分派的指令的位置。跟踪在超前运行操作之后要从获取逻辑所分派的指令的位置可包括跟踪跟随队列中的最后指令的指令的位置。在一些实施例中，重新启动指令指针可用来跟踪跟随队列中的最后指令的位置。例如，图 1 描绘重新启动指令指针 128，其配置为将获取逻辑 106 引导到用于在重新分派队列中的最后指令之后要获取的下一指令的指令指针。

[0033] 在队列保持 ISA 指令的实施例中，跟踪在超前运行操作之后要从获取逻辑所分派的指令的位置可以是直截了当的，因为每个 ISA 指令可具有与其相关联的指令指针。在一些实施例中，诸如微操作束被保持在队列中的实施例中，可以是与指令相关联的指令指针被包括在仅一个束中。

[0034] 在这类实施例中，在 206 的跟踪在超前运行操作之后要从获取逻辑所分派的指令的位置可包括跟踪被插入到队列中的最后完成的指令的位置（例如用于跟随其所有组成部分保持在队列中的指令的指令的位置 / 地址）。例如，指令指针可被包括在形成束的指令集中的最后束中。因为从微操作束中标识指令的结束可能是困难的，所以可使用跟踪指针以维护对在插入到队列中的最后完成的指令与可呈现在队列中的其他束之间的边界的跟踪。例如，图 1 示出指向被包括在指令 B 中的最后束的跟踪指针 130。

[0035] 跟踪最后完成指令的位置并因此跟踪在队列中的最后完成的指令和其他指令之间的边界还可包括随着完成的指令插入到队列中而更新该位置 / 边界信息。进一步地，更新位置 / 边界信息还可包括随着完成的指令插入到队列中，一经从超前运行退出则针对要

由获取逻辑所获取的指令来更新位置信息。反过来，微处理器(例如在一些实施例中，调度器逻辑)将具有对在超前运行之后一经重新进入正常操作则可被发送到获取逻辑的有效的重新启动指令指针的访问权限，并将能够停止从队列重新分派指令以及开始适当地从获取逻辑和/或解码逻辑分派束。在图 1 中示出的示例中，一经对跟踪指针 130 进行更新则可更新重新启动指令指针 128，在 132 示意性地示出对重新启动指令指针 128 的更新。

[0036] 在 208，方法 200 包括准备进入超前运行操作。在一些实施例中，在 208 针对超前运行的准备可以包括冻结队列并将微处理器的状态设置检查点，其可包括将寄存器条目值和与那些寄存器条目相关联的各状态和/或位设置检查点。被设置检查点的值在超前运行的持续期期间保留在被设置检查点的状态中。那些值、状态和/或位的工作状态版本至少最初保留在微处理器一经进入超前运行的工作状态版本中，但那些值、状态和/或位中的一个或多个在超前运行操作期间可能改变。将理解的是，在一些实施例中，过程 204 和/或 206 可与在 208 的准备进入超前运行并发地发生和/或被包括在 208 的准备进入超前运行中。在一些实施例中，合适的超前运行控制逻辑可控制准备进入超前运行的一个或多个方面。例如，在图 1 中示出的实施例中，超前运行控制逻辑 116 可控制设置检查点的微处理器 100。

[0037] 在 210，方法 200 包括进入并操作在超前运行模式中。例如，图 1 中示出的超前运行控制逻辑 116 可控制进入到超前运行中和在超前运行期间的操作。在超前运行期间，微处理器推测性地执行指令，同时尝试揭示其他潜在的长时延事件、停顿、分支误预测等等。被设置检查点的微处理器状态在超前运行期间不更改，因为状态的更改(例如回写事件或指令提交)可能损害微处理器的一旦解决触发超前运行的事件则返回到超前运行前条件的能力。进一步地，因为在超前运行执行期间所生成的结果可能是无效的，例如，因为其可能依靠未命中的数据，所以将超前运行执行的结果存储到高速缓存位置可能损坏保持在存储器系统中的数据。因此，在超前运行模式期间，各寄存器位置可被抑制(poison)或另行标记为保持无效数据，使得可提高指令吞吐量和其他长时延事件的潜在发现。在超前运行结束之后，那些寄存器位置还原到其超前运行前状态使得正常处理可继续。

[0038] 将理解的是，微处理器操作可能时常处于变化之中，甚至是在超前运行操作期间。因此，取决于在超前运行期间如何操作微处理器，微处理器的在超前运行期间检测其他长时延事件的能力可能以变化的成功率实现。

[0039] 例如，图 3 示意性地示出以静态所描绘的程序 300 的实施例。如图 3 所示，程序 300 包括 8 个指令。第二指令将来自存储器的值加载到寄存器 R1 中。第六指令使用 R1 中所存储的值来计算从其加载数据的地址。

[0040] 图 4 示意性地示出用于在超前运行操作期间(例如在进行中的超前运行片断期间)执行程序 300 的方法 400 的实施例。在图 4 示出的示例中，指令 2 在超前运行期间在 L1 高速缓存中未命中。已标识 L1 高速缓存未命中，微处理器可开始从较高高速缓存级别(例如 L2 高速缓存)检索数据。同时，微处理器抑制 R1 并继续在超前运行中执行程序 300。因为指令 6 引用 R1，所以不使用存储在 R1 中的值计算地址并且不实施加载指令。反而，用于指令 6 的目标寄存器位置被抑制并且微处理器前进，继续执行程序 300。然而，因为指令 6 不尝试加载，所以不知道用于指令 6 的输入存储器位置是否将引起高速缓存命中或未命中。换句话说，通过在指令 6 处抑制目标寄存器并前进，微处理器可能已牺牲标识另一触发超

前运行的事件的机会。虽然出于讨论目的提供图 4 中示出的示例，但将理解的是，当 L1 未命中时抑制目标寄存器而不确定是否伴有 L2 未命中可导致一设定，其中例如指令可频繁地引用一个或多个被抑制的位置。

[0041] 因此，在一些实施例中，可选择一个或多个指令用于在超前运行期间进行重新分派 / 重演。例如，可选择与短持续期事件相关的指令用于重演。在短持续期事件中，原始分派可以不完成，但在短时延之后的重演可以完成。在超前运行期间重演指令可减少在超前运行期间系统中累积的抑制量，这潜在地暴露否则可能未命中的其他事件。将理解的是，在超前运行期间完成的指令可更新当前微处理器状态而非被设置检查点的状态，这允许微处理器一经退出超前运行就返回到被设置检查点的状态。

[0042] 例如，图 5 示意性地示出用于使用指令重演在超前运行操作期间执行程序 300 的方法 500 的实施例。在图 5 示出的示例中，指令 2 在超前运行期间在 L1 高速缓存中未命中。与图 4 示出的示例相似，微处理器开始从较高高速缓存级别（例如 L2 高速缓存）检索数据。同时，微处理器继续在超前运行中执行程序 300，并且触发未命中的指令（指令 2）以及在一些实施例中一个或多个附加指令被选择用于重演。当微处理器获得来自 L2 高速缓存的数据时，其被递送到 L1 高速缓存和到指令 2，该指令 2 与指令 3 和 4 一起被重新分派。在一些实施例中，可将指令的重新分派定时，使得与来自高速缓存（例如图 5 示出的示例中的 L2 高速缓存）的数据的到达并发地发出触发未命中的指令，但将理解的是可采用任何合适的重新分派指令的方式而不脱离本公开的范围。

[0043] 在重新分派指令 4 之后，指令 5-8 的规则分派继续。指令 6 的执行引用 R1，所以可计算地址；一旦计算了地址，则发送用于该地址的加载指令。在图 5 示出的示例中，指令 6 触发高速缓存未命中。反过来，触发未命中的指令（指令 6）以及在一些实施例中一个或多个附加指令被选择用于重演。当微处理器获得来自 L2 高速缓存的数据时，其将被递送到 L1 高速缓存和到指令 6，该指令 6 被重新分派。因此，指令 6 生成地址并触发高速缓存未命中，这在图 4 示出的示例中并未实现。不同于图 4 示出的不采用重演的超前运行的示例，图 5 示出的示例说明在一些设定中，在超前运行期间重演指令的能力可提供机会以标识否则可能未命中的长时延事件，尽管在一些实施例中可能存在与超前运行中的重演相关联的指令吞吐量的减少。

[0044] 将理解的是，在一些实施例中，用来保持触发超前运行的指令和一个或多个附加指令的队列可能不用来保持被选择用于在超前运行期间重演的指令。重新使用队列用于超前运行期间的重演可能引起在超前运行期间的指令的丢失，这潜在地损害一旦解决触发超前运行的事件则微处理器重新启动 / 重新进入非超前运行操作的能力。因此，在一些实施例中，用于超前运行期间的重演的指令的选择可使微处理器抑制指令的目标寄存器并继续超前运行。在一些其他实施例中，在超前运行期间遭遇重演条件可使微处理器退出超前运行。将理解的是，这些示例是非限制性的。在一些实施例中，可在微处理器设计期间预确定用来确定是结束超前运行还是抑制目标寄存器的标准。

[0045] 在一些其他实施例中，可使用与用来保持触发超前运行的指令的队列不同的专用超前运行重演队列来在超前运行期间重演指令。例如，图 6 示意性地示出包括超前运行重演队列 602 的微处理器 600 的实施例，该超前运行重演队列 602 配置为保持被选择用于在超前运行期间重演的一个或多个指令，而保持触发超前运行的指令的队列 118 在超前运行

期间维持冻结。虽然为了清楚起见在图 6 中以线性方式示出超前运行重演队列 602，但将理解的是超前运行重演队列 602 可具有任何合适的形式。例如，超前运行重演队列 602 可布置为循环队列。进一步地，可以以任何合适的方式从超前运行重演队列 602 插入和分派指令。例如，可使用“先入先出”格式从超前运行重演队列 602 插入和分派指令。在另一示例中，可根据从初始分派开始的相对寿命来分派指令，该相对寿命可以与超前运行重演队列 602 内的寿命不同。超前运行重演队列 602 还可具有任何合适的大小。在一个非限制性示例中，超前运行重演队列 602 可包括 16 个条目。

[0046] 在图 6 示出的示例中，被选择用于重演的指令可经由将管线 104 的部分与超前运行重演队列 602 链接的一个或多个插入路径 604 来插入到超前运行重演队列 602 中。分配指针 606 可用来指示超前运行重演队列 602 内的插入位置。当指令(或其一部分)插入到超前运行重演队列 602 中时分配指针 606 可被更新。

[0047] 如上文所引入的，在一些实施例中，可管理来自超前运行重演队列 602 的指令的重新分派，使得与另一事件的发生并发地发出被选择用于重演的指令。例如，可与来自高速缓存的指令相关的数据的到达并发地重新分派触发高速缓存未命中的该指令。可使用读指针以指示超前运行重演队列 602 中的、被选择读取以准备用于分派到调度器逻辑 110 的位置。换句话说，读指针可指示在超前运行重演队列 602 中的、被读取用于重新分派的位置。指示在超前运行重演队列 602 中的、被读取用于重新分派的位置可允许在重新分派指令之前读取与该位置相关联的该指令的各从属。例如，在图 6 示出的实施例中，读指针 608 指示超前运行重演队列 602 中的保持指令 X 的位置。因此，指令 X 将是经由分派路径 610 被路由到调度器逻辑 110 用于重新分派到执行逻辑 112 的下一指令。当被选择的指令(或其一部分)被分派到调度器逻辑 110 时读指针 608 可被更新。

[0048] 在一些实施例中，超前运行重演队列 602 可包括释出指针 612，其指示超前运行重演队列 602 中的、准备好逻辑地并在一些情况下物理地从队列中移除的位置。换句话说，释出指针 612 指向指令，该指令是要通过覆写、删除、或类似地消除来从超前运行重演队列 602 中移除的下一指令。在图 6 示出的示例中，释出指针 612 指示由指令 X 所占用的位置。该位置将是要被释出的下一位置，这引起指令 X 从超前运行重演队列 602 中消除。当被选择的指令(或其一部分)已被分派到调度器逻辑 110 时释出指针 612 可被更新。在一些实施例中，读指针 608 可实施释出指针 612 的功能，使得已经按照由读指针 608 的指示来读取 / 分派则释出位置。

[0049] 虽然图 6 示出的示例将超前运行重演队列 602 描绘为与队列 118 分开，但将理解的是，在一些实施例中，可使用合适大小的队列以保持触发超前运行的指令和被选择用于在超前运行期间重演的指令。例如，队列的一部分可在超前运行期间冻结而另一部分是活动的用于重演。在一些其他实施例中，可使用队列 118 的被设置检查点的版本以保持触发超前运行的指令，而使用队列 118 的工作状态版本以保持被选择用于在超前运行期间重演的指令，从而实施超前运行重演队列 602 的功能。

[0050] 无论在超前运行期间是否实施重演，一旦初始的触发超前运行的事件被解决，则微处理器退出超前运行并返回到正常的非超前运行操作。因此，采用图 2 继续，在 212，方法 200 包括确定触发超前运行的事件的解决，以及在 214，退出超前运行操作。可以由微处理器的任何合适的部分以任何合适方式实施对触发超前运行的事件已被解决的确定。例

如，在图 1 示出的实施例中，超前运行控制逻辑 116 配置为确定触发超前运行的事件的解决并使微处理器退出超前运行。退出超前运行可包括还原微处理器 100 的被设置检查点的版本、转储管线 104、解冻队列 118 以及重新引导获取逻辑 106 以开始从重新启动指令指针 128 获取指令。在包括超前运行重演队列 602 的实施例中，退出超前运行还可包括转储超前运行重演队列 602。

[0051] 在 216，方法 200 包括对当触发超前运行时插入到队列中的触发超前运行的指令与一个或多个附加指令一起进行重新分派。触发超前运行的指令与附加指令一起被重新分派。在分派最后的附加指令之后，微处理器开始分派新获取的指令。

[0052] 在超前运行之后从队列重新分派触发超前运行的指令可包括从由读指针所指示的位置分派触发超前运行的指令。换句话说，读指针可指示队列 118 中的被读取用于重新分派的位置。指示队列 118 中的被读取用于重新分派的位置可允许在重新分派指令之前读取与该位置相关联的该指令的各从属。例如，在图 1 示出的实施例中，读指针 134 指示队列 118 中的保持指令 A 的束 A1 的位置。因此，束 A1 将是被分派到调度器逻辑 110 的、经由分派路径 136 被路由用于重新分派到执行逻辑 112 的下一指令。当被选择的指令（或其一部分）被分派到调度器逻辑 110 时读指针 134 可被更新。然而将理解的是，在其他实施例中，分派路径 136 可将指令直接地分派到执行逻辑 112 和其中的任何级。

[0053] 在一些实施例中，队列 118 可包括释出指针 138，其指示队列 118 中的、准备好逻辑地并在一些情况下物理地从队列中移除的位置。换句话说，释出指针 138 指向指令，该指令是要通过覆写、删除、或类似地消除来从队列 118 中移除的下一指令。在图 1 示出的示例中，释出指针 138 指示由指令 A 的束 A1 所占用的位置。该位置将是要释出的下一位置，这引起束 A1 从队列 118 中的消除。当被选择的指令（或其一部分）已被分派到调度器逻辑 110 时释出指针 138 可被更新。在一些实施例中，读指针 134 可实施释出指针 138 的功能，使得一经按照读指针 134 的指示来读取 / 分派则释出位置。

[0054] 一旦与触发超前运行的指令一起插入到队列中的最后指令被从队列重新分派，则微处理器停止从队列重新分派指令并开始分派由获取逻辑所检索的指令。因此，在 216 的重新分派包括一经遭遇插入到队列中的最后完成的指令的指示，则停止从队列的指令的重新分派。例如，在图 1 示出的实施例中，一经遭遇跟踪指针 130，则微处理器可停止从队列 118 重新分派指令。进一步地，在 216 的重新分派包括在从队列重新分派最后完成的指令之后分派由获取逻辑所检索的指令。

[0055] 在一些实施例中，在 216 的重新分派可包括将与用来跟踪跟随队列中的最后指令的位置的重新启动指令指针相关联的位置发送到管线的前端（例如在一些实施例中，获取逻辑），但可在 214 或一经退出超前运行的另一合适的点处发送该位置而不脱离本公开的范围。例如，在图 1 示出的实施例中，由获取逻辑 106 从由重新启动指令指针 128 所指示的位置所检索的指令被分派到执行逻辑 112。在一些实施例中，可从调度器逻辑 110 和 / 或从获取逻辑 106 分派新获取的指令。例如，可将从由重新启动指令指针 128 所指示的位置所检索的新获取的指令从获取逻辑 106 分派到解码逻辑 108 用于在架构指令的情况下的解码。在一些实施例中，在新获取的指令包括架构指令的本地转译的设定中，从由重新启动指令指针 128 所指示的位置所检索的新获取的指令可绕过解码逻辑 108。无论指令是否被解码，一经到达调度器逻辑 110，新获取的指令在插入到队列 118 中的最后完成的指令被重新

分派之后被分派到执行逻辑 112。

[0056] 图 7 示意性地描绘示出被包括在存储器系统 102 的一些示例中的附加细节的微处理器 700 的实施例。图 7 中示出的微处理器 700 的实施例描绘存储器系统 102。存储器系统 102 包括存储器层级 702，其可包括 L1 处理器高速缓存 702A、L2 处理器高速缓存 702B、L3 处理器高速缓存 702C、主存储器 702D（例如一个或多个 DRAM 芯片）、二级存储 702E（例如磁性、固态和 / 或光学存储单元）和 / 或三级存储 702F（例如磁带群）。应该理解示例存储器 / 存储部件以访问时间和容量增大的顺序列出，尽管有可能的例外。

[0057] 存储器控制器 704 可用来处置协议并提供主存储器 702D 所需的信号接口以及可用来调度存储器访问。存储器控制器可实现在处理器裸片（die）上或分开的裸片上。应该理解以上所提供的存储器层级是非限制性的并且可以使用其他存储器层级而不脱离本公开的范围。

[0058] 图 7 示出的微处理器 700 的实施例还包括一个或多个处理器寄存器 706。在一些实施例中，寄存器 706 可被包括在通用寄存器堆中或另外合适地分布在微处理器 700 内。寄存器 706 存储在操作期间所使用的数据和 / 或指令。例如，管线级可从输入寄存器获得数据，使用获得的数据实施指定操作，并随后将结果存储在目标寄存器。将理解的是，标签“目标寄存器”和“输入寄存器”是相对术语。例如，用于第一操作的目标寄存器可起到用于另一操作的输入寄存器的作用；类似地，用于一个操作的输入寄存器可能已起到用于先前操作的目标寄存器的作用，以此类推。

[0059] 在一些实施例中，可采用配置为在寄存器 706 中指示相关联位置的可信度的抑制位来扩充一个或多个寄存器 706。换句话说，每个抑制位指示被包括在相关联寄存器 706 中的数据的有效性 / 无效性。例如，可使用抑制位以指示在超前运行操作期间特定寄存器条目是否成为无效的。

[0060] 图 7 示出的微处理器 700 的实施例包括管线 104。管线 104 包括获取逻辑 106、解码逻辑 108、调度器逻辑 110、执行逻辑 112 和回写逻辑 114。将理解的是，被包括在图 7 示出的实施例中的管线 104 中的级以及图 1 和 6 中示出的那些级是典型 RISC 实现方案的示例，但其并非意在限制。例如，在一些实施例中，可以在管线的上游提供获取逻辑和调度器逻辑功能性，诸如编译 VLIW 指令或指令集转译。在一些其他实施例中，调度器逻辑可被包括在微处理器的获取逻辑和 / 或解码逻辑中。更一般地，微处理器可包括获取、解码和执行逻辑，其中的每一个可包括一个或多个级，而由执行逻辑实行 mem（例如加载 / 存储）和回写功能性。本公开同样适用于这些和其他微处理器实现方案，包括可使用 VLIW 指令和 / 或其他逻辑指令的混合实现方案。

[0061] 在所描述的示例中，可以每次一个地获取和执行指令，这可能要求多个时钟周期。在此期间，大部分数据路径可能未被使用。补充或取代单个指令获取，可使用预获取方法来改善性能并避免与读取和存储操作（即读取指令以及将这类指令加载到处理器寄存器和 / 或执行队列中）相关联的时延瓶颈。因此，将理解的是可以使用任何合适的获取、调度和分派指令的方式而不脱离本公开的范围。

[0062] 本书面描述使用示例来公开本发明，包括最佳模式，并且还使相关领域的普通技术人员能够实践本发明，包括制造和使用任何设备或系统以及实施任何所包含的方法。本发明的可专利范围由权利要求所定义，并且可以包括本领域普通技术人员所理解的其他示例。这类其他示例旨在处于权利要求的范围内。

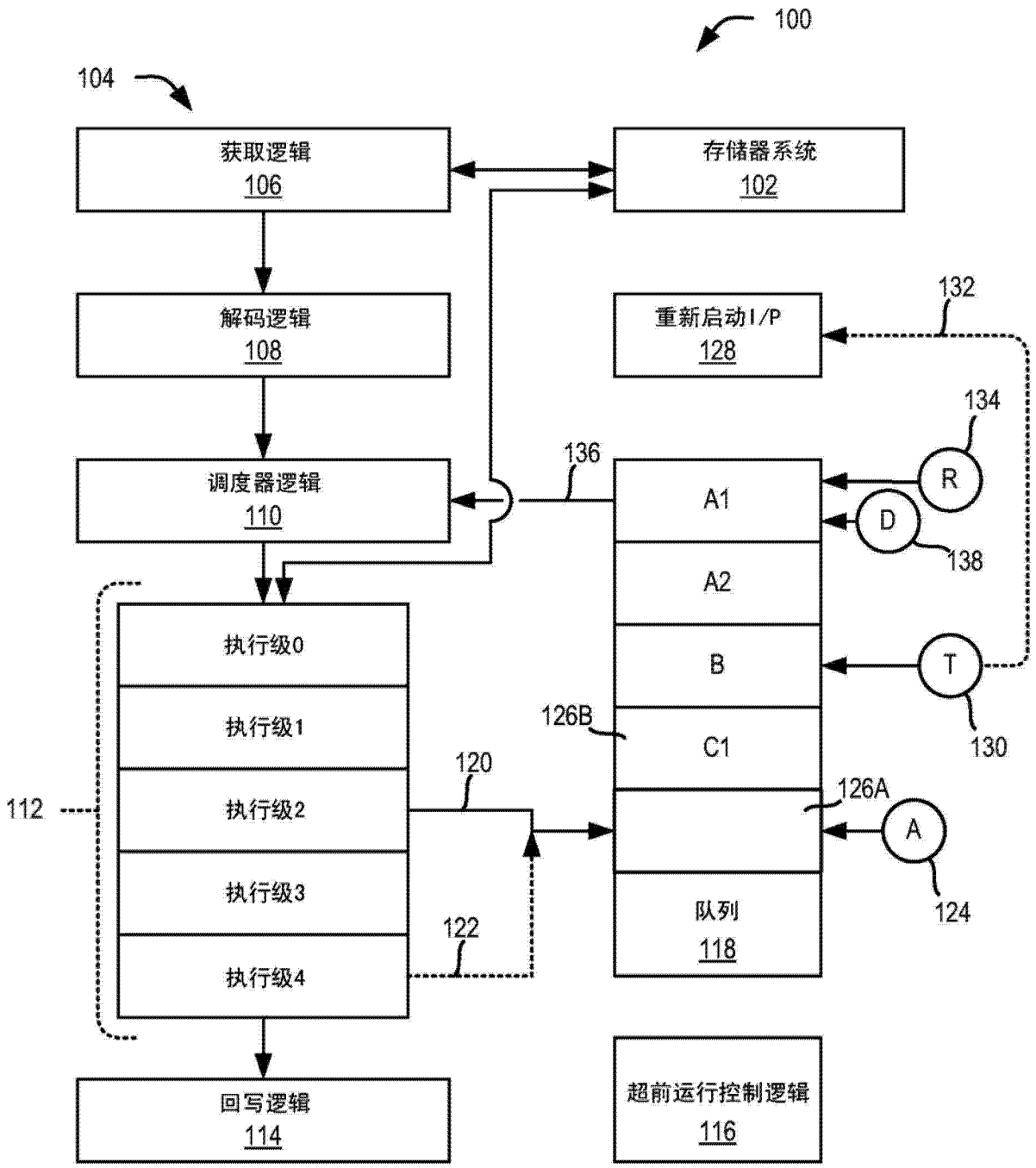

100
104
获取逻辑
106
存储器系统
102
解码逻辑
108
重新启动I/P
128
132
134
调度器逻辑
110
136
A1
R
D
138
A2
B
T
130
126B
C1
120
112
126A
A
执行级0
执行级1
执行级2
执行级3
执行级4
122
124
队列
118
回写逻辑
114
超前运行控制逻辑
116

图 1

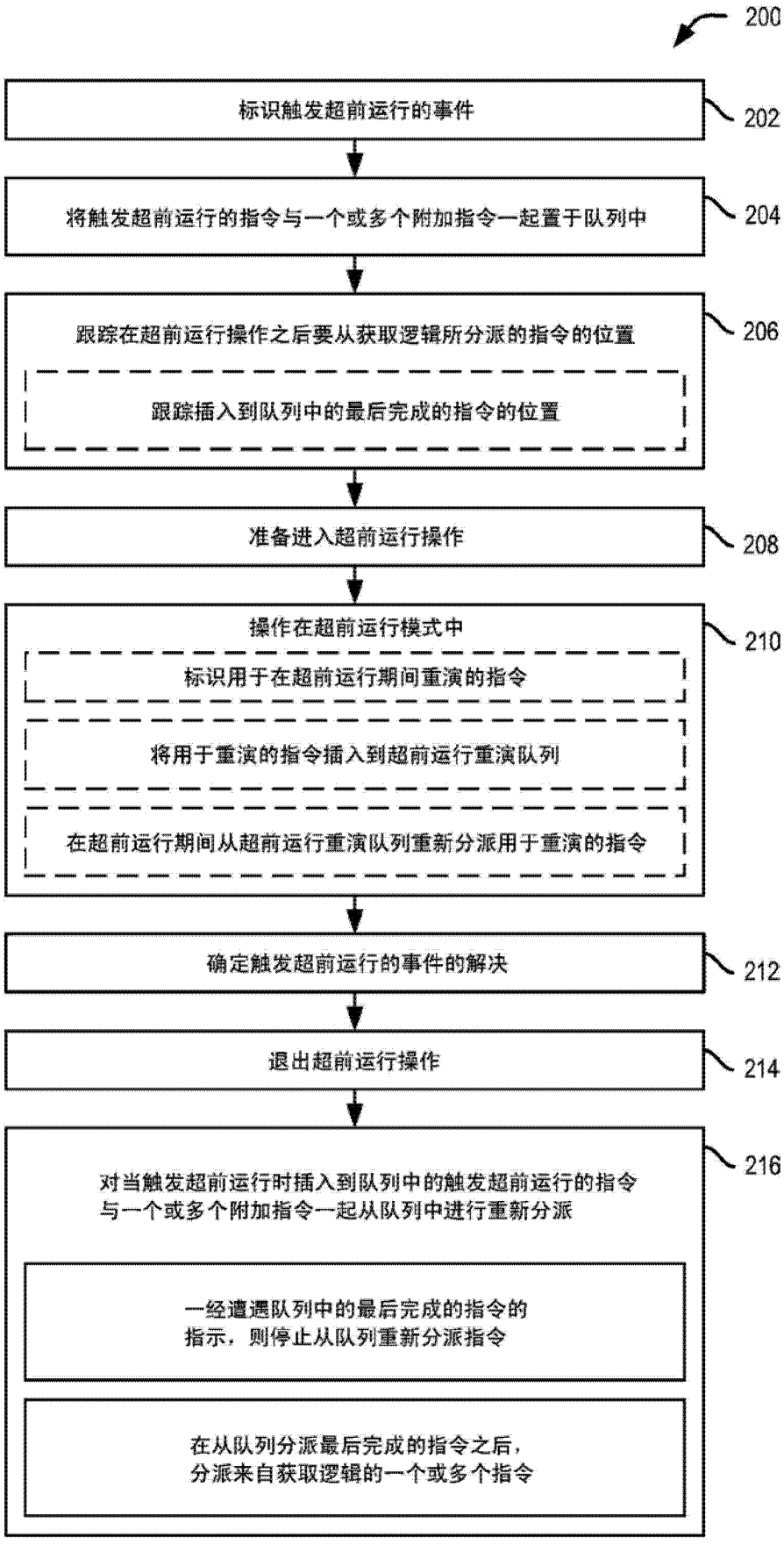

200
标识触发超前运行的事件
202
将触发超前运行的指令与一个或多个附加指令一起置于队列中
204
跟踪在超前运行操作之后要从获取逻辑所分派的指令的位置
跟踪插入到队列中的最后完成的指令的位置
206
准备进入超前运行操作
208
操作在超前运行模式中
标识用于在超前运行期间重演的指令
将用于重演的指令插入到超前运行重演队列
在超前运行期间从超前运行重演队列重新分派用于重演的指令
210
确定触发超前运行的事件的解决
212
退出超前运行操作
214
对当触发超前运行时插入到队列中的触发超前运行的指令
与一个或多个附加指令一起从队列中进行重新分派
一经遭遇队列中的最后完成的指令的
指示，则停止从队列重新分派指令
在从队列分派最后完成的指令之后，
分派来自获取逻辑的一个或多个指令
216

图 2

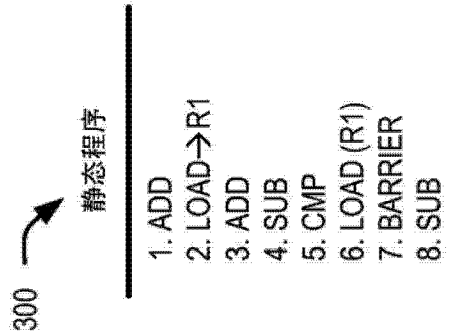

300
静态程序
1. ADD
2. LOAD→R1
3. ADD
4. SUB
5. CMP
6. LOAD (R1)
7. BARRIER
8. SUB

图 3

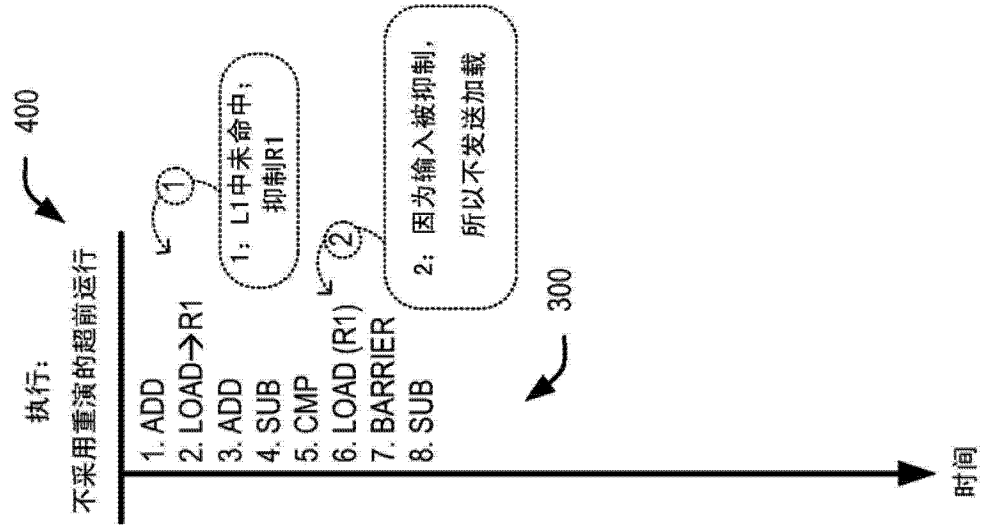

400
执行：
不采用重演的超前运行
1. ADD
2. LOAD→R1
3. ADD
4. SUB
5. CMP
6. LOAD (R1)
7. BARRIER
8. SUB
1
1：L1中未命中；
抑制R1
2
2：因为输入被抑制，
所以不发送加载
300
时间

图 4

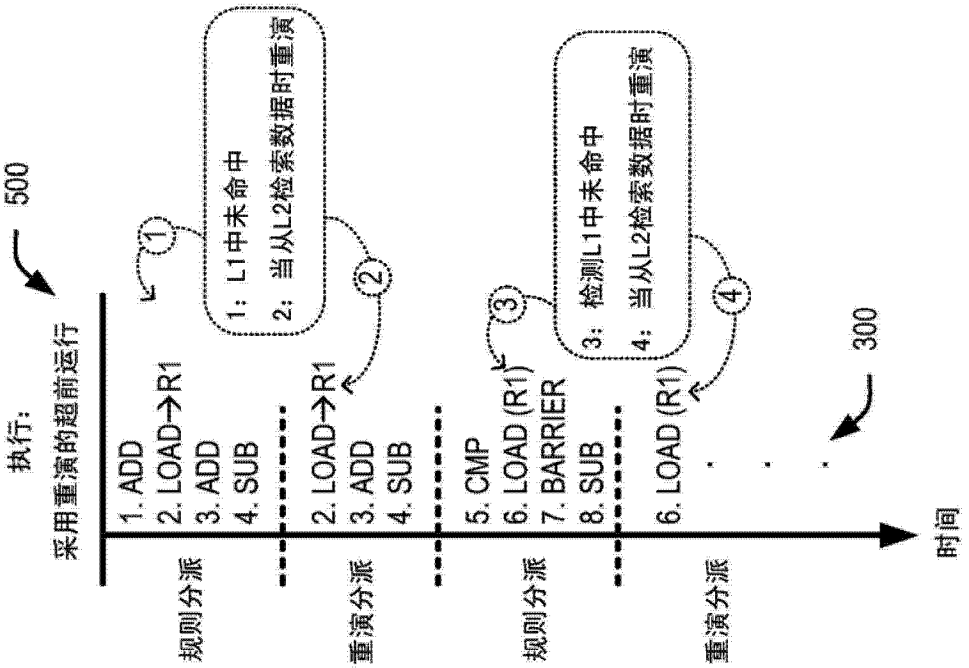

500
执行：
采用重演的超前运行
1. ADD
2. LOAD→R1
3. ADD
4. SUB
2. LOAD→R1
3. ADD
4. SUB
5. CMP
6. LOAD (R1)
7. BARRIER
8. SUB
6. LOAD (R1)
1：L1中未命中
2：当从L2检索数据时重演
3：检测L1中未命中
4：当从L2检索数据时重演
规则分派
重演分派
规则分派
重演分派
300
时间

图 5

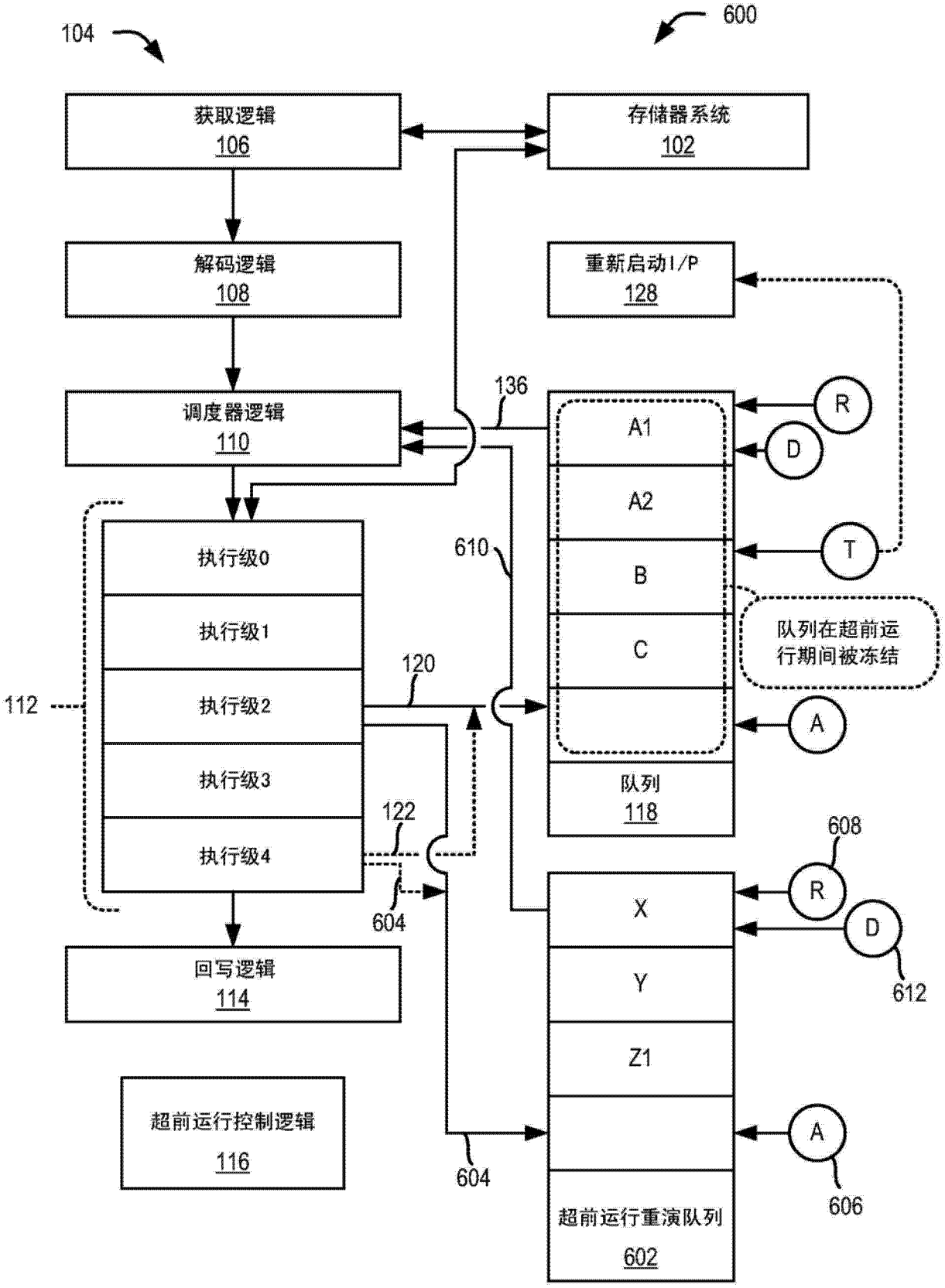

104
600
获取逻辑
106
存储器系统
102
解码逻辑
108
重新启动I/P
128
调度器逻辑
110
136
A1
A2
B
C
R
D
T
A
队列
118
队列在超前运行期间被冻结
610
执行级0
执行级1
执行级2
执行级3
执行级4
112
120
122
604
回写逻辑
114
超前运行控制逻辑
116
X
Y
Z1
608
R
D
612
A
606
超前运行重演队列
602

图 6

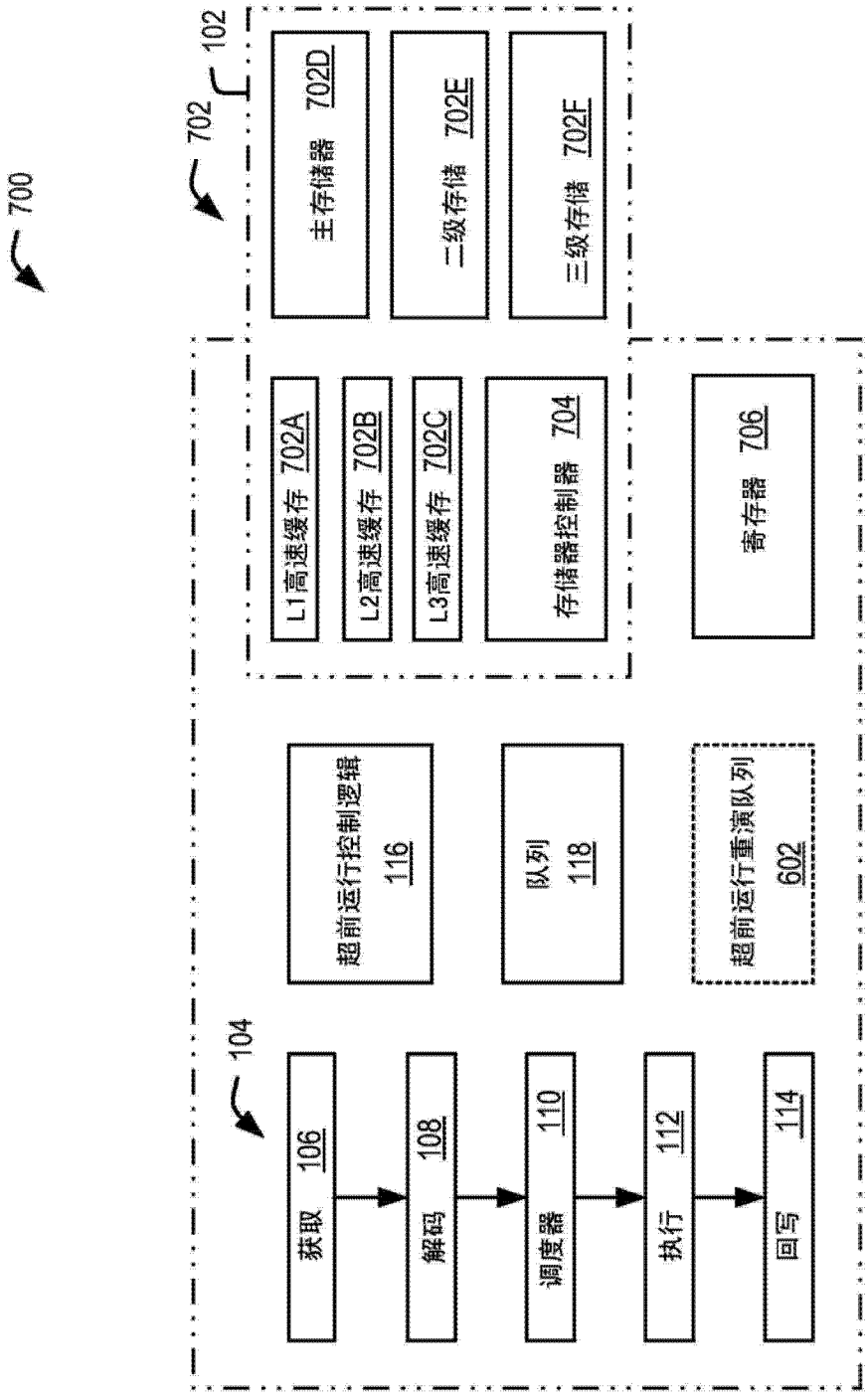

700
102
702
主存储器 702D
二级存储 702E
三级存储 702F
L1高速缓存 702A
L2高速缓存 702B
L3高速缓存 702C
存储器控制器 704
寄存器 706
超前运行控制逻辑 116
队列 118
超前运行重演队列 602
104
获取 106
解码 108
调度器 110
执行 112
回写 114

图 7