



(12)发明专利

(10)授权公告号 CN 103593169 B

(45)授权公告日 2017.09.05

(21)申请号 201310619576.1

(56)对比文件

(22)申请日 2013.11.29

CN 1397877 A, 2003.02.19,

(65)同一申请的已公布的文献号

CN 101727312 A, 2010.06.09,

申请公布号 CN 103593169 A

CN 103336681 A, 2013.10.02,

(43)申请公布日 2014.02.19

US 8332866 B2, 2012.12.11,

(73)专利权人 深圳中微电科技有限公司

US 5819308 A, 1998.10.06,

地址 518057 广东省深圳市南山区高新南
区科技南12路18号长虹科技大厦706-
8室

CN 102436367 A, 2012.05.02,

(72)发明人 梅思行 劳咏仪 王世好

CN 1560734 A, 2005.01.05,

(74)专利代理机构 深圳市科吉华烽知识产权事
务所(普通合伙) 44248

US 2004/0236926 A1, 2004.11.25,

代理人 刘显扬

审查员 张力

(51)Int.Cl.

G06F 9/38(2006.01)

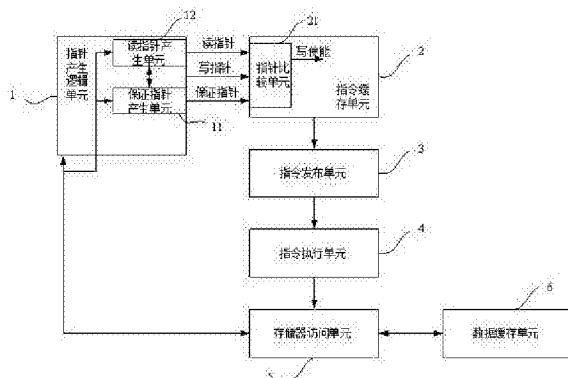
权利要求书2页 说明书7页 附图3页

(54)发明名称

一种多线程处理器中的指令输出装置、方法
及其处理器

(57)摘要

本发明涉及一种多线程处理器中的指令输出装置,包括指针产生逻辑单元、指令缓存单元、指令发布单元、指令执行单元和存储器访问单元;所述指令缓存单元在未收到来自所述存储器访问单元的、表示数据冒险存在的中断信号时按照取指周期依次输出所述读指针指向的指令;收到所述中断信号时,所述指令缓存单元暂停当前线程指令的输出,并将读指针替换为保证指针数值。本发明还涉及一种指令输出方法及其处理器。实施本发明的一种多线程处理器中的指令输出装置、方法及其处理器,具有以下有益效果:其处理的时间较少、控制简单、在等待数据时不会占用宝贵的硬件资源。



1. 一种多线程处理器中的指令输出装置，其特征在于，包括指针产生逻辑单元、指令缓存单元、指令发布单元、指令执行单元和存储器访问单元；所述指针产生单元产生表示当前运行线程的本取指周期内输出的指令所在的读指针、表示当前运行线程的本取指周期内输入指令的存储位置的写指针以及表示当前线程中已输出但未执行指令中最先输出的一条指令位置的保证指针，并分别输出所述读指针、写指针和保证指针到所述指令缓存单元；所述指令发布单元将其得到的指令传输到所述指令执行单元；所述指令执行单元通过所述存储器访问单元取得当前其接收指令涉及的数据并实现该指令；

所述指令缓存单元在未收到来自所述存储器访问单元的、表示数据冒险存在的中断信号时按照取指周期依次输出所述读指针指向的指令；收到所述中断信号时，所述指令缓存单元暂停当前线程指令的输出，并将读指针替换为保证指针数值；

所述指令缓存单元还包括用于比较输入的写指针和保证指针的指针比较单元；所述指针比较单元比较所述写指针和所述保证指针，且在二者相同时，使得本取指周期取得的当前线程指令停止写入所述指令缓存单元。

2. 根据权利要求1所述的多线程处理器中的指令输出装置，其特征在于，所述指针产生逻辑单元还包括保证指针产生单元，所述保证指针产生单元在开始取指时取得所述当前运行线程的读指针的值作为保证指针的初始值，并在所述保证指针指向的指令执行后，更新所述保证指针值。

3. 根据权利要求2所述的多线程处理器中的指令输出装置，其特征在于，所述保证指针的更新包括按照设定的步长指向本线程下一条已输出或未输出的指令。

4. 根据权利要求3所述的多线程处理器中的指令输出装置，其特征在于，所述指针产生逻辑单元还包括用于产生读指针的读指针产生单元，所述读指针产生单元依照指令输出产生读指针并在收到来自所述存储器访问单元的第一设定信号时，重启所述指令输出被暂停的线程，并重新输出其与所述保证指针一致的读指针所指向的指令到所述指令发布单元；所述第一设定信号表示至少在上一个或多个取指周期检测的、未有效存在的输出指令涉及的数据在本取指周期已经有效存在。

5. 根据权利要求4所述的多线程处理器中的指令输出装置，其特征在于，所述存储器访问单元还与存储器连接，所述存储器是与外部存储器连接的数据缓冲区域或数据缓冲单元；所述有效存在为涉及的数据已经读入所述数据缓冲区域或数据缓冲单元并存储。

6. 根据权利要求5所述的多线程处理器中的指令输出装置，其特征在于，所述指针产生逻辑单元还包括停止信号产生单元，所述停止信号产生单元分别输出放弃当前指令执行的控制信号到所述指令发布单元和所述指令执行单元。

7. 一种多线程处理器中的指令输出方法，其特征在于，包括如下步骤：

A) 产生分别用于当前运行线程指令输出和输入的读、写指针以及用于表示已输出但尚未确认是否无数据冒险的最后一条指令的位置的保证指针；

B) 判断是否接收到表示所述保证指针指向的指令存在数据冒险的中断信号，如是，暂停本线程的指令输出，并将读指针替换为当前保证指针值；否则，执行步骤C)；

C) 比较所述写指针和所述保证指针是否相同，如相同，不写入指令，并执行下一步骤；否则，写入取得的指令并执行下一步骤；

D) 继续输出本线程指令，并在输出指令已执行时，更新保证指针值；执行本步骤后，返

回步骤A)；

当处于暂停本线程指令输出时,还包括如下步骤:

E) 判断是否接收到第一设定信号,如是,使用本线程当前的读指针值作为读指针,并返回步骤A);否则,重复本步骤;其中,所述第一设定信号表示至少在上一个或几个取指周期检测的、未有效存在的输出指令涉及的数据在本取指周期已经有效存在。

8. 根据权利要求7所述的指令输出方法,其特征在于,所述步骤B)中,进一步包括:

输出控制信号到指令发布单元和指令执行单元,取消其中正在等待其涉及的数据到来的已发布指令。

9. 一种多线程处理器,包括用于输出指令并执行的、设置在其指令输出流水线上的指令输出装置,其特征在于,所述指令输出装置是如权利要求1-6任意一项所述的多线程处理器中的指令输出装置。

一种多线程处理器中的指令输出装置、方法及其处理器

技术领域

[0001] 本发明涉及处理器的指令实现,更具体地说,涉及一种多线程处理器中的指令输出装置、方法及其处理器。

背景技术

[0002] 在典型的处理器指令处理流水线上,当指令发布(instruction is issued)后,可能产生数据冒险(data hazard)的情况,此时,该指令将不得不停止并等待,直到数据冒险被清除后才能重新执行。例如,一个上载/存储的指令将不知道其需要的数据是否被存入数据缓存或内部的存储器。在后一种情况下(即一个存储的指令),其数据将直到多个时钟周期后才会到达,使得该指令一直停止并等待其数据由外部的存储器到达。这种停止不但带来了极大的时间临界性(time critical),而且使在指令处理流水线的各个阶段保持控制该停止的指令变得复杂。如果该流水线处理的指令是多线程中的一个,其复杂程度将倍增。同时,该停止的指令也将占用指令实现模块进行等待,从而对整个线程的进度带来影响。

发明内容

[0003] 本发明要解决的技术问题在于,针对现有技术的上述处理时间增多、复杂、占用硬件资源的缺陷,提供一种处理时间较少、简单、不会占用硬件资源的多线程处理器中的指令输出装置、方法及其处理器。

[0004] 本发明解决其技术问题所采用的技术方案是:构造一种多线程处理器中的指令输出装置,包括指针产生逻辑单元、指令缓存单元、指令发布单元、指令执行单元和存储器访问单元;所述指针产生单元产生表示当前运行线程的本取指周期内输出的指令所在的读指针、表示当前运行线程的本取指周期内输入指令的存储位置的写指针以及表示当前线程中已输出但未执行指令中最先输出的一条指令位置的保证指针,并分别输出所述指针到所述指令缓存单元;所述指令发布单元将其得到的指令传输到所述指令执行单元;所述指令执行单元通过所述存储器访问单元取得当前其接收指令涉及的数据并实现该指令;

[0005] 所述指令缓存单元在未收到来自所述存储器访问单元的、表示数据冒险存在的中断信号时按照取指周期依次输出所述读指针指向的指令;收到所述中断信号时,所述指令缓存单元暂停当前线程指令的输出,并将读指针替换为保证指针数值。

[0006] 更进一步地,所述指令缓存单元还包括用于比较输入的写指针和保证指针的指针比较模块;所述指针比较单元比较所述写指针和所述保证指针,且在二者相同时,使得本取指周期取得的当前线程指令停止写入所述指令缓存单元。

[0007] 更进一步地,所述指针产生逻辑单元还包括保证指针产生单元,所述保证指针产生单元在开始取指时取得所述当前运行线程的读指针的值作为保证指针的初始值,并在所述保证指针指向的指令执行后,更新所述保证指针值。

[0008] 更进一步地,所述保证指针的更新包括按照设定的步长指向本线程下一条已输出或未输出的指令。

[0009] 更进一步地，所述指针产生逻辑单元还包括用于产生读指针的读指针产生单元，所述读指针产生单元依照指令输出产生读指针并在接收到来自所述存储器访问单元的第一设定信号时，重启所述被暂停的线程，并重新输出其与所述保证指针一致的读指针所指向的指令到所述指令发布单元；所述第一设定信号表示至少在上一个或多个取指周期检测的、未有效存在的输出指令涉及数据在本取指周期已经有效存在。

[0010] 更进一步地，所述存储器访问单元还与存储器连接，所述存储器是与外部存储器连接的数据缓冲区域或数据缓冲单元；所述数据有效存在为涉及的数据已经读入所述数据缓冲区域或数据缓冲单元并存储。

[0011] 更进一步地，所述指针产生逻辑单元还包括停止信号产生单元，所述停止信号产生单元分别输出放弃当前指令执行的控制信号到所述指令发布单元和所述指令执行单元。

[0012] 本发明还涉及一种多线程处理器中的指令输出方法，包括如下步骤：

[0013] A) 产生分别用于当前运行线程指令输出和输入的读、写指针以及用于表示已输出但尚未确认是否无数据冒险的最后一条指令的位置的保证指针；

[0014] B) 判断是否接收到表示所述保证指针指向指令存在数据冒险存在的中断信号，如是，暂停本线程的指令输出，并将读指针替换为当前保证指针值；否则，执行步骤C；

[0015] C) 比较所述写指针和所述保证指针是否相同，如相同，不写入指令，并执行下一步骤；否则，写入取得的指令并执行下一步骤；

[0016] D) 继续输出本线程指令，并在输出指令已执行时，更新保证指针值；执行本步骤后，返回步骤A；

[0017] 当处于暂停本线程指令输出时，还包括如下步骤：

[0018] E) 判断是否接收到第一设定信号，如是，使用本线程当前的读指针值作为读指针，并返回步骤A；否则，重复本步骤；其中，所述第一设定信号表示至少在上一个或几个取指周期检测的、未有效存在的输出指令涉及数据在本取指周期已经有效存在。

[0019] 更进一步地，所述步骤B)中，进一步包括：

[0020] 输出控制信号到指令发布单元和指令执行单元，取消其中正在等待其涉及的数据到来的已发布指令。

[0021] 本发明还涉及一种多线程处理器，包括用于输出指令并执行的、设置在其指令输出流水线上的指令输出装置，所述指令输出装置是上述任意一项所述的多线程处理器中的指令输出装置。

[0022] 实施本发明的一种多线程处理器中的指令输出装置、方法及其处理器，具有以下有益效果：由于在传统的指令流水线上都是使用读指针输出或发布指令，比较写指针和读指针的值来判断指令缓存是否写满，这使得输出指令在指令缓存中的存储位置将被输入的指令覆盖，从而使得输出指令在遇到数据冒险时只能占用指令流水线的硬件资源进行等待；而本发明则增加了一个表示指令涉及数据状态的保证指针，通过比较写指针和保证指针的值来判断输入的指令是否可以写入指令缓存，使得其数据未准备好的指令不用占据硬件资源而是在数据缓冲单元中等待（即使其已经输入，也可以被取消，但在指令缓存中的内容不会被覆盖），使得硬件资源可以对别的线程或指令进行处理；在相关数据到位后，才输出或再次输出指令，并执行该指令。所以，其处理的时间较少、控制简单、在等待数据时不会占用宝贵的硬件资源。

附图说明

[0023] 图1是本发明一种多线程处理器中的指令输出装置、方法及其处理器实施例中指令输出装置的结构示意图；

[0024] 图2是本发明所述实施例中一种情况下的指令输出装置结构示意图；

[0025] 图3是所述实施例中指令输出方法的流程图。

具体实施方式

[0026] 下面将结合附图对本发明实施例作进一步说明。

[0027] 如图1所示，在本发明一种多线程处理器中的指令输出装置、方法及其处理器实施例中，该指令输出装置包括指针产生逻辑单元1、指令缓存单元2、指令发布单元3、指令执行单元4和存储器访问单元5；其中，指针产生单元产生表示指令存储地址的读指针(read-ptr)传输到指令缓存单元2；指令缓存单元2在得到上述指令读指针并在取指周期开始时，传输该指针指向地址存储的指令到指令发布单元3；上述指针产生逻辑单元1同时还产生写指针(write-ptr)传输到指令缓存单元2，指令缓存单元2将依据写指针的指向，将在本取指周期开始时取得的指令写入指令缓存单元2中由该写指针指向的位置，但是，这个指令的写入是有条件的，这个条件正是本实施例中的技术方案区别于传统的指令输出方案的重要区别之一，稍后将详细描述；此外，上述指针产生逻辑单元1还产生保证指针(commit-ptr)，该保证指针表示该指针所指向的指令缓存单元1中的地址存储的指令所涉及的数据已经有效存在，在本实施中，该指令已经被输出；也就是说，保证指针所指向的指令肯定是已经被输出，但是还没有被执行的指令；如果一个指令已经被执行，则其涉及的数据必然已经到位；如果一个指令被输出，则并不意味着其涉及的数据已经到位或准备好。在现有技术中，一个指令已经被输出(或被发射)，而其涉及的数据又没有由外部存储器或别的地方进入内部存储器，导致该指令只能占用硬件资源等待这些数据，则认为出现数据冒险。在本实施例中，保证指针的设置就是为了消除出现数据没有到位时对硬件资源的占用。指令发布单元3将其得到的指令进行必要的处理之后传输到指令执行单元4；指令执行单元4通过存储器访问单元5取得当前其接收指令涉及的数据并实现该指令。在本实施例中，上述在取指周期开始时取得的指令写入指令缓存单元2的条件是写指针的值不等于保证指针的值(或者说写入的地址没有指向虽然已经被输出，但是还没有被执行的指令所在的区域)时，将本取指周期内取得的指令写入写指针指向的位置；也就是说，在本实施例中，如果一条已经输出的指令尚未被确定是否存在数据冒险(其涉及的数据是否已经在内存中有效存在)或被执行，则该指令在指令缓存单元2中就将保存，不会被覆盖，以便于出现数据冒险其在输出后的处理步骤中被消除并在数据冒险被消除后该指令的重新输出或发布；如果写指针和保证指针指向同一个存储位置，则写入的指令将覆盖之前该位置上存储的指令，而该指令虽然已经输出，一旦出现数据冒险，该指令只能在输出后的步骤中等待数据冒险的消除，这正是传统的指令输出方法，这将使得输出的指令在数据冒险时占用硬件资源等待，从而带来处理时间的延长及资源的浪费。一般来讲，在指令输出时，其正常输出的指令地址是按照设定的步长递增或递减的，该步长就是指令本身的长度，例如，加入指令为单字节指令，且输出由0开始，则第一条指令地址为0，第二条指令地址为1，并依此类推；假设保证指针的值是2，则新的指

令写入只能写在0和1两个地址,不能写在2,更不能写在2以后的地址。也就是说,保证指针被视为一个点,其划分了可以写入新的指令的区域和不可以写入新的指令的区域。

[0028] 在本实施例中,指针产生逻辑单元1产生表示本取指周期内输出指令缓存单元2指令所在的读指针、表示本取指周期内输入指令缓存单元2的指令存储位置的写指针以及表示由指令缓存单元2已经输出的、尚未执行的、紧随已执行的最后一条指令的指令位置(即在当前未消除数据冒险等待时间的指令中最先输出的指令的位置)的保证指针,并分别输出这些指针到指令缓存单元2;在本实施例中,指令缓存单元2按照取指周期的安排,输出读指针指向的指令到指令发布单元3,指令发布单元3将其得到的指令传输到指令执行单元4;指令执行单元4通过存储器访问单元5取得当前其接收指令涉及的数据并实现或执行该指令。同样地,指令缓存单元2也按照取指周期的安排在取指周期开始时由外部存储区域中取得一条输入的指令,但是,这条取得的输入指令是否能够写入指令缓存单元2中,则需要进行判断,在满足上述条件的情况下,才能写入。因此,在本实施例中,指令缓冲单元2还包括指针比较模块21,指针比较模块21用于比较输入的写指针和保证指针;在二者相等时输出控制信号,使得本取指周期取得的指令不能写入指令缓存单元2;而在写指针的值不同于保证指针的值时(此时,表示写入新指令的位置还没有达到上述保证指针指向的区域划分点)输出控制信号,使得本取指周期取得的指令写入指令缓存单中该写指针指向的存储位置。

[0029] 在本实施例中,上述保证指针在开始取值时,表现为读指针的一个拷贝,其数值就是读指针的值;但是随着指令的执行,可能会有多个指令虽然存在与指令流水线上,但还没有执行;由于保证指针是在其指向的指令被执行后才能增加或更新,所以保证指针的值将落后于上述读指针,其表示了已经输出的指令但尚未执行的、在时间上最先输出的指令在指令缓存单元2中的位置(通常来讲,此时该指令后面可能还排列有多个同样已输出但尚未执行的指令),但是,其指向的指令还没有执行,也就是不能确认没有数据冒险;当确认该指令确认没有数据冒险或数据冒险已消失的情况下(这种情况的结果就是该指令被执行),该保证指针更新,例如,按照系统规定的方式加一或减一,指向下一条已输出指令但是排列在上述被执行指令之后的那条指令。在本实施例中,为了实现上述性能,在指针产生逻辑单元1还包括保证指针产生单元11,保证指针产生单元11在系统开始取指时取得读指针的值作为保证指针的初始值,并在接收到来自存储器访问单元5的、表示该读指针指向的指令已执行的信号时,更新所述保证指针值。更新的方式与传统的读指针更新的方式是相同的,均遵从系统的设定,例如,更新的方向或步长。

[0030] 如果保证指针指向的指令存在数据冒险,也就是说,在执行该指令时,存储器访问单元5不能得到其涉及的所有数据,该指令就没有办法执行。此时,存储器访问单元5输出中断信号到指针产生逻辑单元1。在现有技术中,出现这种情况时,该指令占用硬件资源并等待数据冒险的消失,也就是等待这些数据的到位。但是,在本实施例中,如果出现这种情况,则暂停该线程的指令输出,放弃已经输出的本线程的指令,将后面的硬件资源释放,使得别的线程可以使用这些硬件资源。同时,指针产生逻辑单元1将读指针用保证指针的值替换。也就是说,此时,将保证指针的值拷贝到读指针中。这样,当恢复本线程输出指令时,读指针就指向当前的保证指针所指向的指令,由于该指令尚未被执行,所以在恢复本线程指令输出时,首先输出该指令并执行。

[0031] 总之,在本实施例中,在指令取消信号之前,指令会继续从读指针所指处发布,不

管读指针和保证指针是否相同。这是因为从指令发布到知道有没有数据阻碍会有好几个时钟周期，在这几个周期内，读指针和保证指针不会相同，但不表示当前线程应该等待。唯有取消信号发出，该线程才会等待（而同时读指针重回到保证指针的值）。

[0032] 同时，请参见图2，在本实施例的一种情况下，为了在存在数据冒险的情况下将已经输出的指令停止执行(cancel)，指针产生逻辑单元2还包括分别输出放弃当前指令执行的控制信号到所述指令发布单元和所述指令执行单元的停止信号产生单元13。该停止信号产生单元13是在上述出现了中断信号，表明数据冒险存在的情况下输出该控制信号的。这种设置进一步保证了对已输出指令的消除，即可以主动清除已发出的指令，为别的线程使用硬件资源提供更好的条件，避免了一个线程在等待由于某种原因迟到的数据时，对硬件资源的占用。当然，在一些情况下，也可以不管上述已输出在流水线上的指令，仅仅释放硬件资源，由接手控制该硬件资源的线程对其进行处理。这样可以减少本线程的操作，对当前线程而言，可以降低退出操作的复杂性。

[0033] 此外，在本实施例中，存储器访问单元5还与存储器6连接，存储器6是与外部存储器（图中未示出）连接的数据缓冲区域或数据缓冲单元；而在本实施例中，数据有效存在就是涉及的数据已经读入上述数据缓冲区域或数据缓冲单元并存储。

[0034] 请参见图3，在本实施例中，还涉及一种多线程处理器中的指令输出方法，包括如下步骤：

[0035] 步骤S11 产生读指针、写指针和保证指针：在本步骤中，如同传统的指令输出方法一样，产生读指针和写指针。一般来讲，这两个指针时随取指周期的进行而变化的，变化的步长是与系统相关的。在一个新的取指周期到来之际，这两个指针都会在原来的基础上变化设定的步长。当然，在本实施例中也会出现一些例外的情况，例如，如果上一个取指周期的取得指令并没有写入指令缓存单元中，则需要视系统设置情况判断是否再次取得存储在外部存储器中的指令。在传统的指令输出方法中，正如前面所述的一样，该读、写指针产生并被送到指令缓存单元，而指令缓存单元则依据该读指针输出指令以及依据该写指针将取得指令写入指令缓存单元中。如果该输出指令涉及的数据未准备好，该指令将在流水线上等待。而本实施例中涉及的指令输出方法则不然，其通过后面所述的步骤，避免了上述的指令由于数据不到位而出现的在流水线上等待的情况，从而使得流水线的硬件资源得到合理的利用。其中，在初始时，保证指针是读指针数值的一个拷贝，但是其更新的方法和读指针是不同的。读指针除了在后面的步骤中所述的被替换的情况下，都是按照取指周期及设定步长更新的。而保证指针一旦具有初始值，在未收到表示其指向的指令被执行的信号时，是不更新的；只有在其指向的指令被执行后才在当前的基础上按照设定步长更新。

[0036] 步骤S12收到中断信号否：在本步骤中，判断是否收到中断信号，如是，执行步骤S13；否则，执行步骤S16；在本实施例中，上述中断信号是存储器访问单元输出的、表示当前执行的指令所涉及的数据未取得的信号。通常来讲，该信号的出现表示数据冒险的存在。

[0037] 步骤S13暂停本线程指令输出：在本步骤中，由于读指针和保证指针不同，也就是说，已经输出的指令中至少有一条存在数据冒险的可能（通常是在未执行的指令中最先输出的那条指令），其数据还没有由外部存储器中送入，因此，该指令已经在等待中，在其后输出的指令也进入等待状态。整个硬件通道将等待该指令执行。所以，在本步骤中，停止本线程指令的输出，其目的是让出硬件资源给别的线程。

[0038] 步骤S14替换读指针为保证指针:在本步骤中,由于已经暂停本线程的指令输出,但是,已经输出的指令还有至少一条是没有执行的,为了保证本线程再次载入时由该条指令开始执行,在本步骤中,将读指针的值替换为保证指针的值,即保证本线程再次载入时开始执行的指令就是上次由于数据冒险而未执行的指令,确保了线程的正确执行。

[0039] 步骤S15是否收到第一设定信号:在本步骤中,判断是否收到第一设定信号,如是,表明前面所述的数据冒险已经消除,可以将本线程重新载入,跳转到步骤S11,重新开始执行本线程,值得一提的是,由此处跳转到步骤S11开始执行时,其读指针的值是步骤S14中替换过的读指针的值;否则,表明前面所述的数据冒险尚未消除,重复本步骤。在本步骤中,第一设定信号表示至少在上一个或多个取指周期中发现的已输出指令涉及数据在当时未有效存在的,而在本取指周期已经有效存在。也就是说,第一设定信号出现之前,一定出现过一个中断信号,该中断信号表示当时执行的指令由于其涉及的数据冒险没有成功执行;而第一设定信号的出现,表示这些数据现在已经有效存在,该指令可以继续执行了。于是,第一设定信号出现后,将本线程重新载入,由上次未能成功执行的指令开始执行。

[0040] 步骤S16保证指针和写指针相同否:在本步骤中,判断保证指针的值是否与写指针的相同,如相同,执行步骤S17;否则,执行步骤S18.

[0041] 步骤S17读得的指令不写入:在本步骤中,由于写指针指向保证指针指向的位置,所以由外部存储器取得的新的指令不能写入。一般来讲,不断地取得外部存储器存储的指令,同时不断地输出指令执行,构成程序执行的基本步骤。在现有技术中,实际上对新取得指令的写入也是有限定的,即不能覆盖尚未输出的指令所在的位置。但是,就本实施例而言,由于其要求较为特别,也就是已经输出但尚未执行的指令可能会被放弃,然后再重新输出。所以,其禁止写入的区域还需要包括这些已经输出但尚未执行的指令的存储位置。而在本实施例中,采用了保证指针来指示这一位置,划分可写入区域和禁止写入区域。

[0042] 步骤S18 读得的指令写入:在本步骤中,将由外部存储器取得的新的指令写入写指针指向的位置。请注意,在本步骤中,写指针指向的存储位置还没有到以保证指针为分界点的禁止写入区域,不会存在覆盖虽已输出但还未执行或尚未输出的指令的风险。所以,可以写入新指令。

[0043] 步骤S19 输出读指针指向的指令:在本步骤中,按照取指周期的设定,继续输出本线程指令,并在输出指令(通常是尚未执行的指令中的最先输出的指令,即排在未执行指令队列中最前面的指令)已执行时,更新保证指针值;保证指针的更新是按照系统的设定步长而递增或递减的。执行本步骤后,返回步骤S11;此时的写指针、读指针和保证指针值将被携带到步骤S11中。

[0044] 上述步骤S11-S19体现了一个当前执行线程取指周期中指令的输入和输出的步骤。当然,众所周知,处理器取指是多个取指周期连续执行而实现的。所以,按照传统的规则,在正常的取指周期完成后,上述几个指针的值被代入上述步骤S11,重新开始一个取指周期。其中,读指针和写指针可能会按照传统的规则变化,但是,保证指针的值将被代入步骤S11,不会变化(在出现第一设定信号的情况下,该值指的是在第一设定信号出现后经过变换的值)。

[0045] 在本实施例中,还涉及一种处理器,该处理器包括用于依次取得并执行指令的指令输出装置,该指令输出装置就是上述的指令输出装置。在本实施例,上述处理器可以是任

何设置有这种指令输出装置或取指装置的任何处理器。例如该处理器可以是CPU、GPU、CPU和GPU的混合体、流处理器、多核的并行处理器以及并行的、具有较强功能的嵌入式处理器中的一种或数种的组合,只要这种处理器是多线程处理器就可以使用本实施例中的这些装置或方法来实现不占用硬件资源(硬件资源被别的线程使用)等待数据的到来。

[0046] 基本上来说,在本实施例中,通过停止执行上载/存储的指令,并使其在指令缓存中等待,彻底地消除了当数据缓存中缺失相关数据时,指令在流水线上的停止和等待(在流水线上等待会占用硬件资源,并带来时序复杂、耗时等缺陷)。这样,当指令在上述指令缓存中等待数据由外部存储器中进入内部的数据缓存时,不会占用流水线或硬件资源。在这种情况下,允许其他线程使用该正在等待的指令曾经占用的硬件资源,从而在流水线上达到更好的负载平衡,且不存在不同的SMT线程之间转换的时延。而该等待的指令保存在指令缓存中,同时,与其同一线程的其他指令均不能发布(be issued)。当等待的数据由外部的主存储进入内部的数据缓存后,该等待的指令被重新发布(be reissued),且其所在的线程被如常地重新执行(resume)。

[0047] 以上所述实施例仅表达了本发明的几种实施方式,其描述较为具体和详细,但并不能因此而理解为对本发明专利范围的限制。应当指出的是,对于本领域的普通技术人员来说,在不脱离本发明构思的前提下,还可以做出若干变形和改进,这些都属于本发明的保护范围。因此,本发明专利的保护范围应以所附权利要求为准。

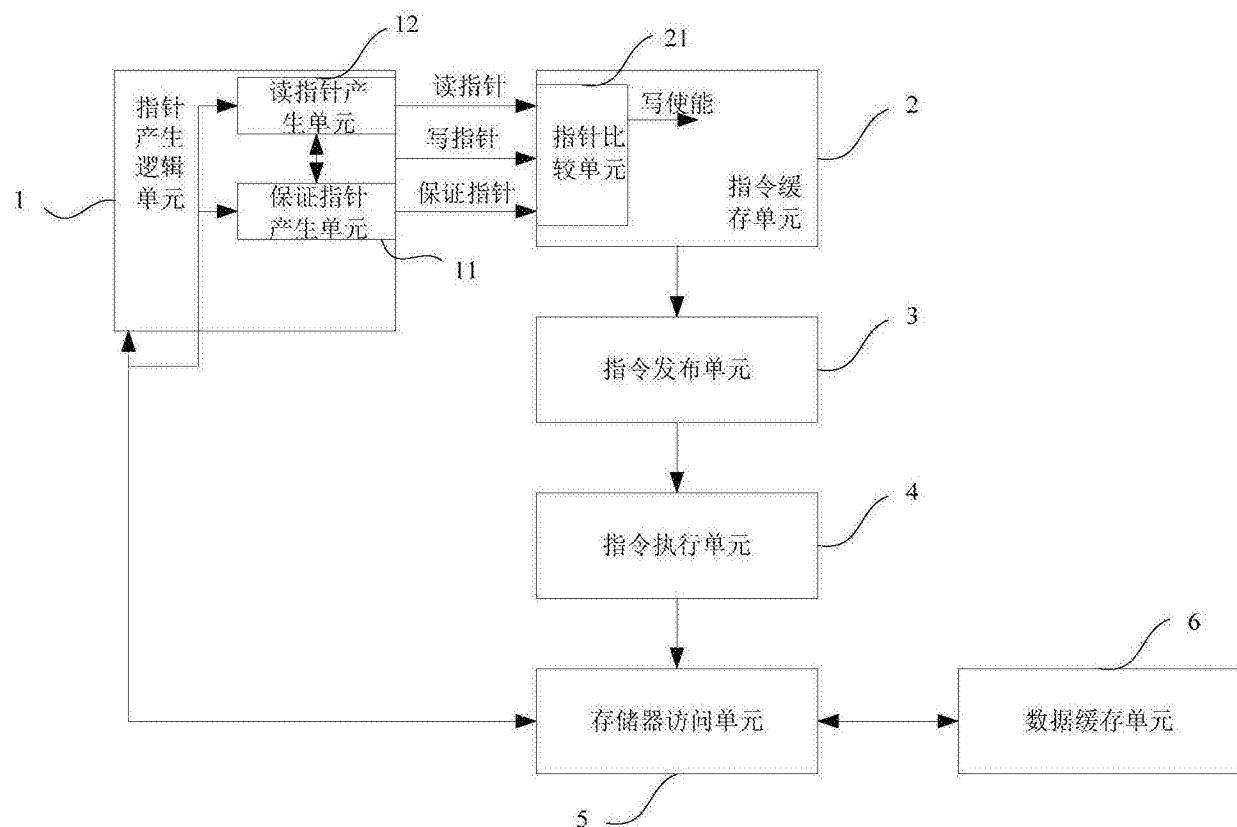


图1

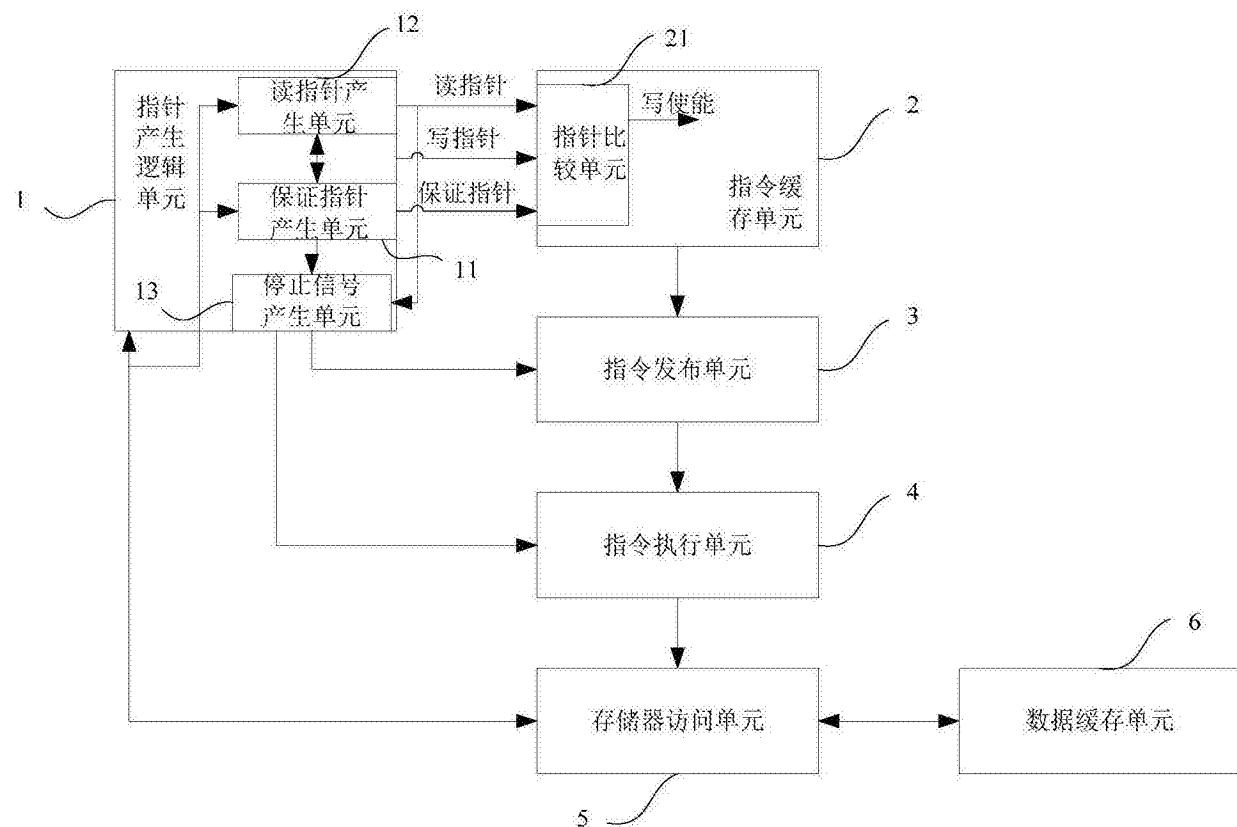


图2

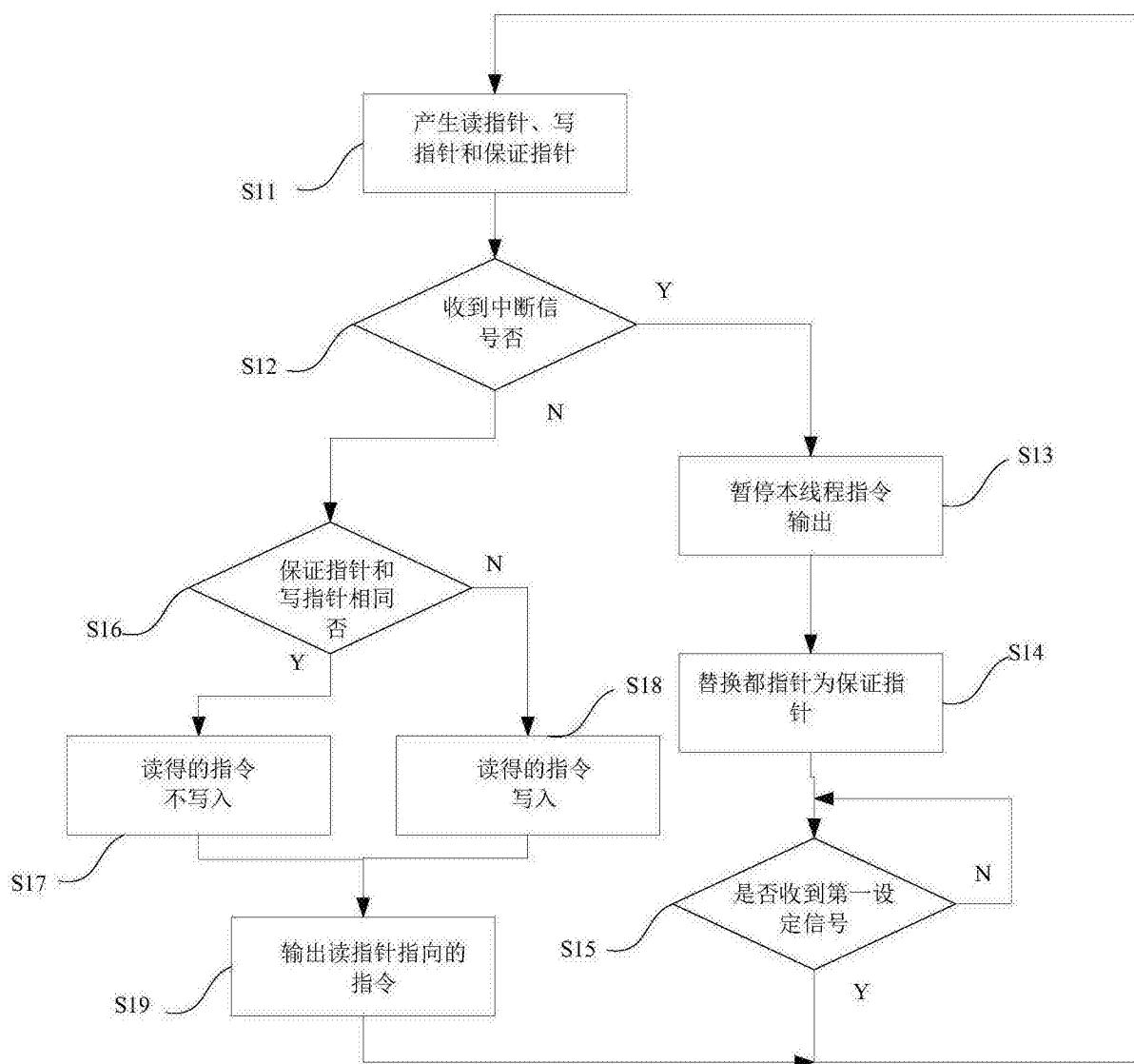


图3