



[12] 发明专利申请公开说明书

[21]申请号 95101692.X

[51]Int.Cl⁶

G06F 9/30

[43]公开日 1996 年 1 月 10 日

[22]申请日 95.1.16

[30]优先权

[32]94.3.31 [33]US[31]221,329

[71]申请人 国际商业机器公司

地址 美国纽约州

[72]发明人 宋承润

[74]专利代理机构 中国专利代理(香港)有限公司

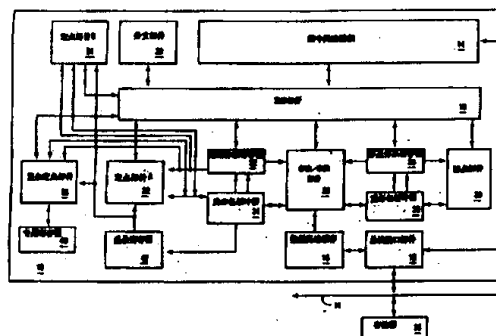
代理人 杜有文 马铁良

权利要求书 4 页 说明书 38 页 附图页数 21 页

[54]发明名称 处理系统和操作方法

[57]摘要

本发明提供了一个处理系统和操作方法，一个序列中的多条指令被调度到执行电路去执行。对于每条指令，在其执行之前首先判断该指令的执行是否会引引起一个异常。处理系统将完成一条指令以响应判断出该指令以及在该序列中该指令之前的每条指令均不引起一个异常，而不依赖于执行电路是否已经执行完毕每条不可能引起异常的指令。



权 利 要 求 书

1. 一个处理系统的操作方法,其特征在于包含下面的步骤:

将一个序列中的大量指令调度到执行电路中执行;

对于每条指令,在其执行之前判断所述指令的执行是否会引起一个异常;以及

完成一条指令,只要判断出所述指令以及在所述序列中所述指令之前的每条指令的执行均不引起一个异常,而不管所述执行电路是否已经执行完毕每条不可能引起异常的指令。

2. 根据权利要求 1 的方法,其特征在于所述的指令包含一条执行串行化指令,在所述序列中,该指令之前为一条特定的指令。

3. 根据权利要求 2 的方法,其特征在于所述的判断步骤包含下面的步骤:判断所述特定指令的执行不会引起一个异常。

4. 根据权利要求 3 的方法,其特征在于包含下述步骤:在完成所述特定指令后,于结束执行所述特定指令之前,执行所述执行串行化指令。

5. 根据权利要求 4 的方法,其特征在于在所述序列中,所述执行串行化指令之前为一条不同的指令,并且所述执行串行化指令的一个信息要依赖于所述不同指令的一个结果。

6. 根据权利要求 5 的方法,其特征在于所述执行步骤包含下面的步骤:进一步执行所述执行串行化指令,以响应所述不同指令的执行完毕。

7. 根据权利要求 4 的方法,其特征在于包含下面的步骤:在所

述特定指令完成之前执行所述执行串行化指令的第一阶段。

8. 根据权利要求 7 的方法, 其特征在于所述执行步骤包含下面的步骤: 在完成所述特定指令之后, 于结束执行所述特定指令之前, 执行所述执行串行化指令的第二阶段。

9. 根据权利要求 8 的方法, 其特征在于所述执行串行化指令是一条 Store 指令。

10. 根据权利要求 1 的方法, 其特征在于包含下面的步骤: 在一个排序缓冲器中保存一个标志, 以响应一条特定指令不可能引起一个异常。

11. 根据权利要求 10 的方法, 其特征在于所述保存步骤包含下面的步骤: 进一步保存所述标志, 以响应所述特定指令属于一类可被随机执行的指令。

12. 根据权利要求 10 的方法, 其特征在于包含下面的步骤: 在所述排序缓冲器中删除所述标志, 以响应所述特定指令的完成。

13. 根据权利要求 1 的方法, 其特征在于包含下面的步骤: 在一个排序缓冲器中保存一个标记, 以响应一条特定指令为一条执行于非精确的、不可恢复的浮点异常模式下的浮点指令, 即使所述浮点指令有可能引起一个异常。

14. 一个处理系统, 其特征在于包含:

执行指令的执行电路;

与所述执行电路耦合的调度电路, 该调度电路完成下面的工作:

将一个序列中的大量指令调度到所述执行电路去执行; 以及

对于每条指令, 在其执行之前判断所述指令的执行是否会引起一个异常;

与所述调度电路耦合的完成电路,用于完成一条指令,以响应判断出所述指令以及在所述序列中的所述指令之前的每条指令均不引起一个异常,而不管所述执行电路是否已经执行完毕不可能引起异常的每一条指令。

15. 根据权利要求 14 的系统,其特征在于所述的指令包含一条执行串行化指令,在所述序列中,该指令之前为一条特定指令。

16. 根据权利要求 15 的系统,其特征在于所述调度电路判断出所述特定指令的执行不会引起一个异常。

17. 根据权利要求 16 的系统,其特征在于所述执行电路在完成所述特定指令之后,于结束执行所述特定指令之前,执行所述执行串行化指令。

18. 根据权利要求 17 的系统,其特征在于在所述序列中,所述执行串行化指令之前为一条不同的指令,并且所述执行串行化指令的一个信息要依赖于所述不同指令的一个结果。

19. 根据权利要求 18 的系统,其特征在于所述执行电路进一步执行所述执行串行化指令,以响应所述不同指令的执行完毕。

20. 根据权利要求 17 的系统,其特征在于所述执行电路在所述特定指令完成之前执行所述执行串行化指令的第一阶段。

21. 根据权利要求 20 的系统,其特征在于所述执行电路在完成所述特定指令之后,于结束执行所述特定指令之前,执行所述执行串行化指令的第二阶段。

22. 根据权利要求 21 的系统,其特征在于所述执行串行化指令是一条 Store 指令。

23. 根据权利要求 14 的系统,其特征在于包含与所述调度电路

耦合的控制电路,用以在一个排序缓冲器中保存一个标志,以响应一条特定指令不可能引起一个异常。

24. 根据权利要求 23 的系统,其特征在于所述控制电路进一步保存所述标志,以响应所述特定指令属于一类可随机执行的指令。

25. 根据权利要求 23 的系统,其特征在于所述控制电路在所述排序缓冲器中删除所述标志,以响应所述特定指令的完成。

26. 根据权利要求 14 的系统,其特征在于包含与所述调度电路耦合的控制电路,用于在一个排序缓冲器中保存一个标记,以响应一条特定指令为一条执行于非精确的不可恢复的浮点异常模式下的浮点指令,即使所述浮点指令有可能引起一个异常。

说明书

处理系统和操作方法

本专利申请通常涉及到信息处理系统，特别涉及到一个处理系统和操作方法。

一个超标量处理系统包含多个执行单元用以同时执行多条指令。在某些处理系统中，指令可以不按照它们在指令流中的编程顺序执行。然而，一条或多条指令有可能受串行执行化的影响，此处的执行串行化指令只能按照其编程顺序执行。同时，在某些这样的处理系统的设计中，指令的结果要按照指令的编程顺序写入存贮单元(如体系寄存器)。采用以往的典型技术，这种按顺序的回写会对处理系统完成指令的时间产生不利的影响。此外，这种按顺序的回写还会对处理系统执行串行化指令的时间产生不利的影响。

因而，有必要研制这样的处理系统和操作方法：其中按顺序的回写能够对处理系统完成指令的时间以及处理系统执行串行化指令的时间产生较小的不利影响。

在一个处理系统和操作方法中，多条指令被依次地调度到执行电路中去执行。对于每条指令，系统都要在其执行之前先判断一下该指令的执行是否会产生异常。当系统判断出在指令序列中，某条指令的其前面各条指令的执行都不会产生异常时，即完成该指令而不管执行电路是否已经执行完不会产生异常的每条指令。

本发明的一个技术上的优点在于：按照顺序的回写能够对处理

系统完成指令的时间以及系统执行串行化指令的时间产生较小的不利影响。

参照附后的一些图表以及下面对它们所作的说明，可以更好地理解本发明的一个典型的实施例及其优点。其中：

图 1 是根据较佳实施例的用于处理信息的一个处理器的框图；

图 2 是图 1 中处理器的一个定点执行部件的框图；

图 3 是图 1 中处理器的一个定序部件的框图；

图 4 是图 3 中定序部件的一个排序缓冲器的概念性示意图。

图 5a-b 是图 1 中处理器所处理的指令的各个阶段的示意图；

图 6 是图 1 中处理器的一个排序缓冲器的概念性示意图；

图 7 是图 6 中四条指令的各个阶段的示意图；

图 8a-d 是图 1 中处理器的重命名缓冲器的概念性示意图；

图 9 是在不具备较佳实施例的显著特征的情况下，四条指令的各个阶段的示意图；

图 10a-d 是依据图 9 的一个处理器的重命名缓冲器的概念性示意图；

图 11 是图 1 中处理器的重命名缓冲器的另一实施例的框图，

图 12a-c 是图 1 中处理器的一个排序缓冲器的概念性示意图；

图 13 图 12a-c 中五条指令的各个阶段的示意图；

图 14a-f 是图 1 中处理器的重命名缓冲器的概念性示意图；

图 15 是在不具备较佳实施例的显著特征的情况下，图 12a-c 中五条指令的各个阶段的示意图；

图 16a-h 是在图 15 所示的指令处理的各个周期中，图 1 中处理器的重命名缓冲器的概念性示意图。

通过参照图 1-16h, 我们可以更好地来理解本发明的一个典型的实施例。附图中使用同类数码来标识相应的同类图表。

图 1 是依据较佳实施例的一个信息处理器 10 的结构图。在较佳实施例中, 处理器 10 是一个单片集成电路的超标量微处理器。如下所述, 它包括各种处理部件, 寄存器, 缓冲器, 存贮器等, 所有这些都由集成电路构成。再者, 在较佳实施例中, 处理器 10 根据精简指令集计算技术 (RISC) 进行操作。如图 1 所示, 系统总线 11 与处理器 10 的一个总线接口部件 (BIU) 12 相连。BIU12 控制处理器 10 和系统总线 11 之间的信息传输。

BIU 12 与处理器 10 的一个指令高速缓存 14 及一个数据高速缓存 16 相连。指令高速缓存 14 将指令输出到定序部件 18。定序部件 18 对来自指令高速缓存 14 的指令作出响应, 有选择地将指令输出到处理器 10 的其他执行电路。

除了定序部件 18 外, 在较佳实施例中, 处理器 10 的执行电路还包括多个执行部件, 即: 分支部件 20, 定点部件 A (FXUA) 22, 定点部件 B (FXUB) 24, 复杂定点部件 (complex fixed point unit - CFXU) 26, 存数/取数部件 (LSU) 28 和浮点部件 (FPU) 30。FXUA 22、FXUB 24、CFXU 26 和 LSU 28 从通用体系寄存器 (general purpose architectural register - GPRs) 32 和定点重命名缓冲器 34 读入它们的源操作数信息。此外, FXUA 22 和 FXUB 24 从进位寄存器 (CA) 42 处获得进位。FXUA 22、FXUB 24、CFXU 26 和 LSU 28 将它们操作的结果 (目的操作数信息) 输出到定点重命名缓冲器 34 的指定单元中存贮起来。再者, CFXU 26 从专用寄存器 (SPRs) 40 输入源操作数信息并向它输出目的操作数信息。

FPU 30 从浮点体系寄存器 (floating point architectural register - FPRs) 36 和浮点重命名缓冲器 38 读入它的源操作数信息, 并将它的操作结果 (目的操作数信息) 输出到浮点重命名缓冲器 38 的指定单元中存贮起来。

为响应取数指令, LSU 28 从数据高速缓存 16 中输入信息, 并将该信息复制到重命名缓冲器 34 和 38 的指定的一个中。如果数据高速缓存 16 中没有存贮该信息, 数据高速缓存 16 则通过 BIU 12 和系统总线 11 从连接在系统总线 11 上的系统存贮器 39 中读入该信息。此外, 数据高速缓存能够通过 BIU 12 和系统总线 11 将信息从数据高速缓存 16 传送到连在系统总线 11 上的系统存贮器 39 中。为响应存数指令, LSU 28 从 GPRs 32 和 FPRs 36 的指定的一个中读出信息; 并将该信息复制到数据高速缓存 16 中。

定序部件 18 从/向 GPRs 32 和 FPRs 36 输入/输出信息。分支部件 20 从定序部件 18 输入指令和指示处理器 10 当前状态的信号。为响应该指令和信号, 分支部件 20 向定序部件 18 输出指示适当内存地址的信号, 该地址所指向的内存空间存贮了一个将由处理器 10 来执行的指令序列。为响应这些从分支部件 20 发出的信号, 定序部件 18 从指令高速缓存 14 中读入所指向的指令序列。如果指令序列中的一条或多条指令没有存贮在指令高速缓存 14 中, 指令高速缓存 14 则 (通过 BIU 12 和系统总线 11) 从连接在系统总线上的系统存贮器 39 中读入该指令。

为响应从指令高速缓存 14 中读入的指令, 定序部件 18 有选择地将指令分派到相应的执行部件 20、22、24、26、28 或 30 中。每个执行部件执行某一特定指令类的一条或多条指令。例如, FXUA 22 和

FXUB 24 对源操作数执行第一类定点数学运算, 如加、减、与、或以及异或; CFXU 26 对源操作数执行第二类定点数学运算, 如定点乘法和除法; FPU 30 对源操作数执行浮点运算, 如浮点乘法和除法。

当信息存放在重命名缓冲器 34 的某一选定的单元里时, 该信息与指令指定的某一存贮单元 (例如, GPRs 32 的某一单元或 CA 寄存器 42) 相关联, 而所选定的重命名缓冲器正是为此指令而分配的。为响应从定序部件 18 中来的信号, 存放在重命名缓冲器 34 中指定单元里的信息被复制到与其相关的 GPRs 32 的某一单元 (或 CA 寄存器 42) 中。正如后面将结合图 6-10 所深入讨论的那样, 响应产生该信息的指令后完成, 定序部件 18 操纵了对存放在重命名缓冲器 34 中某一选定单元里的信息的这种复制过程。该复制过程被称作“回写”。

当信息存放在重命名缓冲器 38 中某一选定的单元里时, 该信息与 FPRs 36 中某一单元相关联。为响应从定序部件 18 中来的信号, 存放在重命名缓冲器 38 中某一选定单元里的信息被复制到与其相关的 FPRs 36 中的某一单元中。响应产生该信息的指令的完成, 定序部件 18 操纵了对存放在重命名缓冲器 38 中某一选定单元里的信息的这种复制过程。

通过在各个执行部件 20、22、24、26、28 和 30 中同时处理多条指令, 处理器 10 可以获得很高的性能。相应地, 每条指令都按照各阶段的序列来处理, 这些阶段都可以与其他指令的阶段并行地执行。这一技术被称作“流水线操作”。本发明的较佳实施例的一个显著特点是, 一条指令通常按六个阶段来处理, 即, 取指、解码、调度、执行、完成和回写。

在取指阶段, 定序部件 18 根据存放指令序列的一个或多个内存地址, 有选择地从指令高速缓存 14 中读入一条或多条指令, 这一点我们已经在上面的叙述中结合分支部件 20 和定序部件 18 做过详细的讨论。

在解码阶段, 定序部件 18 对取到的多至四条的指令进行解码。

在调度阶段, 定序部件 18 为每条被调度指令的结果 (目的操作数信息) 保留一个重命名缓冲器单元, 然后响应解码阶段的解码, 有选择地将经过解码的多至四条的指令调度到相应的执行部件 20、22、24、26、28 或 30 中。在调度阶段, 还将为每个所选定的执行部件提供相应指令的操作数信息。处理器 10 按照指令的编程顺序对它们进行调度。

在执行阶段, 执行部件执行调度给它们的相应指令, 并象前面所详细讨论过的那样, 将运算结果 (目的操作数信息) 输出到重命名缓冲器 34 和 38 中的所选择单元里。通过这种方式, 处理器 10 可以不按照指令的编程顺序来执行指令。

在完成阶段, 定序部件 18 指出指令已经执行完毕, 这将在下面的叙述中结合图 3-4 进行更为详细的讨论。处理器 10 按照指令的编程顺序完成对指令的执行过程。

在回写阶段, 定序部件 18 分别操纵从重命名缓冲器 34 和 38 到 GPRs 32 和 FPRs 36 的信息复制过程。在下面的叙述中, 我们将结合图 6-10, 对该过程作更为详细的讨论。同样是在某一特定指令的回写阶段, 处理器 10 为响应该指令, 将对其体系状态进行更新。处理器 10 按照指令的编程顺序完成各条指令的回写阶段。正如下面将结合图 6-10 所详细讨论的那样, 在某些特定的情形下, 处理器 10 将不

无益处地把指令的完成阶段和回写阶段合二为一。

在较佳实施例中，每条指令需要一个机器周期的时间来完成指令处理过程中的一个阶段。但是，某些指令(如由 CFXU 26 所执行的复杂定点指令)可能需要多于一个周期的时间。因此，相应于先前指令完成所需时间的变化，在特定指令的执行和完成阶段之间可能会产生一个可变的延迟。

图 2 是处理器 10 的执行部件 FXUA 22 的结构图。FXUA 22 包括两个备用站，如 50a 和 50b 所示。同样，分支部件 20、FXUB 24、CFXU 26、LSU 28 和 FPU 30 也各自拥有自己的备用站。为简明起见，下面我们仅以 FXUA 22 为例讨论其它执行部件的操作及其各自的备用站。

备用站 5a 和 5b 都能存放从定序部件 18 分派来的、要求 FXUA22 执行的某条指令的信息。每个备用站包含一个目的寄存器域、操作数 A 域、操作数 B 域和操作码域。此外，作为本发明的较佳实施例的一个显著特点，每个备用站另外还包含一个相应的允许执行(EOK)域。

在目的寄存器域，备用站至少为其相应的指令确定一个目的寄存器(如定序部件 18 所指定的)。类似地，在各自的操作数 A 和操作数 B 域，备用站为其相应的指令存放源操作数信息(自 GPRs 32、重命名缓冲器 34、FXUB 24、CFXU 26 或 LSU 28)。在操作码域，备用站根据其相应的指令存放一操作码(如定序部件 18 所指定的)，该操作码指出 FXUA 22 应对源操作数信息执行何种操作。

为响应从控制逻辑 56 来的信号，执行逻辑 54 从某一备用站的操作数 A 和操作数 B 域读入源操作数信息，并立即执行某一运算

(由备用站中存放的操作码指出)。该运算的结果信息从执行逻辑 54 输出到重命名缓冲器 34、FXUB 24、CFXU 26 和 LSU 28 中。该信息存放在重命名缓冲器 34 中某一选定的单元里。根据由多路转换器 58 的输出所确定的某一目的寄存器, 存放在选定重命名缓冲器里的信息将与 GPRs 32 中的某一单元相关联。

本发明的较佳实施例的一个显著特点是, 备用站还在它的 EOK 域存放 EOK 信息(如定序部件 18 所指定的)。该 EOK 信息有利于包含处理器 10 延迟一条串行指令的执行的某些情况。执行串行化是在诸如处理器 10 之类的多流水线、无序执行处理器中延迟指令执行过程的一种技术。处理器 10 延迟一条串行指令的执行的第一种情况是该指令必须按一定的顺序执行。处理器 10 延迟一条串行指令的执行的第二种情况是该指令必须等到其源操作数信息可用并且有效后方可开始执行。

就第一种情况而言(指令必须按一定的顺序执行), 处理器 10 通常按照随机的顺序执行指令, 为此, 指令就没有按照它在指令流中的程序顺序来执行。相应地, 指令的结果(目的操作数信息)也就没有必要按照指令的编程顺序出现。但是, 处理器 10 是按照指令的编程顺序往体系寄存器(即 GPRs 32 和 FPRs 36)中回写指令的结果的。因此, 处理器 10 中设置了相应的重命名缓冲器 34 和 38, 这些缓冲器用于临时存放指令的结果, 一直到某一合适的时间(即执行完没有异常情况的所有前驱指令)将存放的中间结果往体系寄存器中回写时为止。

不过, 作为较佳实施例中的一个例子, 某些指令在 SPRs 40 (图 1)上进行操作, 这时直接将指令的结果写到 SPRs 40 中, 中间不需要

经过重命名缓冲器。关于这种指令的一个例子是 Move To 指令, 在该指令中, CFXU 26 将信息从 GPRs 32 中的某一单元里移动到 SPRs 40 中的某一单元里。如图 1 所示, CFXU 26 是和 SPRs 40 相连的。Move To 指令在执行过程中刷新 SPRs 40 中的某一单元。作为另外一个例子, 一条存数指令的执行结果是将数据直接写到数据高速缓存 16 中的某一存贮单元里, 中间不需要再经过重命名缓冲器。

处理器 10 不按照随机顺序执行这种类型的指令(其中指令的结果不经过重命名缓冲器而直接写到某一体系寄存器或存贮单元里), 以便能够获得精确的中断和异常处理。此外, 因为指令的结果是直接写到体系寄存器或存贮单元里的, 中间无需经过重命名缓冲器暂存, 所以该指令在执行过程中没有回写阶段。相应地, 为了确保指令的完成阶段和回写阶段按顺序地完成, 处理器 10 延迟对该指令的执行过程, 一直等到其所有的前驱指令都执行完为止。

就第二种情况而言 (指令必须等到其所有的源操作数信息可用并且有效后方可开始执行), 参照图 2, 有一备用站在定序部件将指令调度时为尚未获得源操作数信息的相应指令临时保存信息。为响应自某一执行部件来的关于源操作数信息可用的信号, 该备用站读入并且保存该源操作数信息。备用站于某个适当的时刻将这些源操作数信息转送到执行逻辑 54 中。

在本发明的较佳实施例中, 大多数的指令指定一个或多个 GPRs 32 或 FPRs 36 作为其源操作数。相应地, 在本发明的较佳实施例中, 备用站包含用于转送从执行部件来的信息的电路。

但是, 在本发明的较佳实施例中, 备用站并不包含用于转送从其他类型的源操作数, 如 CA 寄存器 42 或 SPRs 40 来的信息的电路。

这是因为，这种电路的尺寸和花费跟需要无转送源操作数的指令的出现频度比较起来，设置这种电路是很不合算的。为此，处理器 10 至少将这种无转送源操作数类型的指令的执行过程延迟至所有的前驱指令执行完为止。在所有的前驱指令执行完以后，即从体系寄存器（如 SPRs 40）的指定单元中读出相应的无转送源操作数。关于这种类型的指令的例子有：(1) 读 CA 寄存器 42 的扩展算术运算指令，和 (2) 将信息从 SPRs 40 的某一单元传送至 GPRs 32 的某一单元的 Move From 指令。

综上所述，处理器 10 在两种情况下至少将一条串行指令的执行过程延迟至其所有的前驱指令执行完为止。第一种情况是指令不能随机执行。第二种情况是指令至少指定了一个处理器 10 没有为之提供转送电路的源操作数。通过定序部件 18 指定 EOK 信息，并将之存放在备用站里，可以比较好地适应这种情况。

如果某条指令受其执行串行化的影响，那么定序部件 18 会在该指令往 FXUA 22 中调度时将其 EOK 位（在存放于备用站的 EOK 域的 EOK 信息中）清为逻辑 0。通过将 EOK 位清为逻辑 0，定序部件 18 可以阻止 FXUA 22 执行该指令（即使该指令已经准备就绪也不例外）。因此，FXUA 22 只有在定序部件 18 通过线路 60 输出一个信号时，才能开始执行该指令串行化指令，这点将在下面做更进一步的讨论。

如果一条指令不受其执行串行化的影响，那么定序部件会在该指令往 FXUA 22 中调度时将其 EOK 位置为逻辑 1。通过将 EOK 位置为逻辑 1，只要指令的源操作数信息可用并且有效，定序部件 18 允许 FXUA 22 立即执行该指令。

图 3 是定序部件 18 的结构图。如上面所进一步讨论的那样, 在取指阶段, 定序部件 18 有选择地从指令高速缓存 14 中读入多至四条指令, 并将这些指令存放在指令缓冲器 70 中。在解码阶段, 解码逻辑 72 从指令缓冲器 70 中读入取到的多至四条的指令, 并对其解码。在调度阶段, 调度逻辑 74 有选择地将多至四条经过解码的指令分别调度到选定的 (相应于解码阶段的解码结果) 执行部件 20、22、24、26、28 或 30 中。

图 4 是本实施例的定序部件 18 的一个排序缓冲器 76 的概念性示意图。如图 4 所示, 排序缓冲器 76 共有 16 个单元, 分别按缓冲器号 0-15 标记。每个单元有五个域, 即“指令类型”域、“目的 GPR 数”域、“目的 FPR 数”域、“完成”域和“异常”域。另外, 指令类型域还有一个“执行部件”子域和一个“EOK”子域。

参照图 3, 当调度逻辑 74 往某个执行部件调度一条指令时, 定序部件 18 将该指令分配给排序缓冲器 76 中的某一相关单元。定序部件 18 本着先进先出的原则以循环方式把排序缓冲器 76 中的单元分配给所调度的指令, 即定序部件 18 首先分配 0 号单元, 接着按顺序分配单元 1-15, 然后回过来再分配 0 号单元。因为所调度的指令已经在排序缓冲器 76 中分配到了一个相关的单元, 调度逻辑 74 就把与该指令相关的信息输出存放 to 排序缓冲器 76 中相关单元的各个域和子域中。

例如, 在图 4 的 0 号单元中, 排序缓冲器 76 指出该指令已调度给 FXUA 22。此外, 单元 0 指出被调度的指令是串行化的, 其 $EOK = 0$, 处理器 10 至少将该指令的执行延迟至其所有的前驱指令执行完为止。再者, 在 1 号单元, 排序缓冲器 76 指出另一条后续指令

也是串行化的, 同样其 $EOK = 0$ 。

本发明的较佳实施例的另外一些显著特征是, 0 号单元进一步指出被调度的指令具有一个目的 GPR 寄存器 (从而“目的 GPR 数” = 1) 和零个目的 FPR 寄存器 (故“目的 FPR 数” = 0), 并且还没有执行完 (故“完成” = 0), 也没有引起什么异常 (故“异常” = 0)。

在执行部件执行一条分配给它的指令时, 该执行部件会对排序缓冲器 76 中与指令相关联的单元进行修改。特别是, 为响应指令执行完毕这一信号, 执行部件将修改相应单元的“完成”域 (即使“完成” = 1)。如果在执行指令的过程中, 执行部件遇到一个异常信号, 它将修改相应单元的“异常”域 (即使“异常” = 1)。

让我们回过来再看一看图 3, 排序缓冲器 76 的单元由定序部件 18 的完成逻辑 80 和异常逻辑 82 来读取。此外, 本发明的较佳实施例的一个显著特征是, 排序缓冲器 76 的单元还由定序部件 18 的执行串行化逻辑 84 来读取。异常逻辑 82 根据排序缓冲器 76 中的“异常”域来处理在执行指令的过程中所遇到的异常。

根据排序缓冲器 76 中的“完成”域和“异常”域, 完成逻辑 80 向调度逻辑 74、执行串行化逻辑 84 和排序缓冲器 76 中输出相应的信号。通过这些信号, 完成逻辑 80 按编程顺序指出所执行的指令已经完成。完成逻辑 80 指出一条指令已经完成, 如果该指令满足以下条件:

条件 1: 执行单元已经完成对该指令的执行 (从而使得排序缓冲器 76 中与该指令相关联的单元里“完成” = 1);

条件 2: 在处理该指令的任何阶段都没有遇到异常 (即在排序缓冲器 76 中与该指令相关联的单元里“异常” = 0);

条件 3: 任何前面已经调度过的指令都满足条件 1 和条件 2。

调度逻辑 74 根据排序缓冲器 76 中的信息决定增加需调度的指令的合适条数。

本发明的较佳实施例的一个显著特征是, 为响应从完成逻辑 80 来的信号, 执行串行化逻辑 84 通过线路 60 有选择地往 FXUA 22 输出一个信号。如果与调度到 FXUA 22 的指令相关的 EOK 位 (在排序缓冲器 76 中与该指令相关联的单元的“指令类型”域里) 已被清为逻辑 0, 那么, 为响应该被调度指令的所有前驱指令都已完成这一事件, 执行串行化逻辑 84 将通过线路 60 输出该信号。FXUA 22 只有在执行串行化逻辑 84 通过线路 60 发出该信号时才执行调度给它的指令; 当执行串行化逻辑 84 通过线路 60 输出该信号时, 上述调度的指令 (与其相关联的 EOK 位已清为逻辑 0) 将成为 FXUA 22 的备用站 50a-b 中等待最久的一条指令, 因为指令是按照它们的编程顺序“完成”的。

同样, 执行串行化逻辑 84 通过线路 86、88、90、92 和 94 向分别与其相连的执行部件 20、24、26、28 和 30 有选择地输出信号。

图 5a-b 是指令的各个阶段的示意图。参照图 5a, 一条执行串行化的指令 INST $n+1$ (例如一条 Move To 指令或一条 Move From 指令) 的执行 (周期 6) 一直要延迟到所有的前驱指令都完成为止 (例如在前驱指令 INST n 的周期 5 之后)。但是, 处理器 10 并不延迟位于执行串行化指令 INST $n+1$ 之后的指令 INST $n+2$ 的调度 (周期 3) 或执行 (周期 4) 过程。通过这种方式, 处理器 10 可以连续地调度位于一条执行串行化指令 (如 INST $n+1$) 之后的指令 (如 INST $n+2$)。并且, 处理器 10 的一个执行部件还可以不按顺序将指令从该

执行部件的备用站发送到其执行逻辑(如图2中的执行逻辑54),即使在执行部件的备用站中等待最久的指令可能受执行串行化的影响。

较佳实施例中的处理器10能够获得比其他方案更高的性能。如图5b所示,根据某一方案,响应对一条受执行串行化影响的指令INST $i+1$ 的解码(周期2),指令INST $i+2$ 的调度(周期8)过程必须延迟。在这种方案中,只有在所有前面已经调度的指令都已“完成”之后(例如在前面执行串行化指令INST $i+1$ 的周期7之后),才能开始指令INST $i+2$ 的调度(周期8)过程。这种方案的一个缺点是,对于任何位于某一执行串行化指令(如INST $i+1$)之后的指令(如INST $i+2$)来说,其执行过程必须延迟。

如前所述,处理器10可以连续地调度位于一条执行串行化指令之后的指令。一条Move From指令是一条执行串行化指令,因此,它将受益于本较佳实施例的执行串行化技术。此外,本较佳实施例的另外一个显著特征是,处理器10还包含用于将一条Move From指令的结果(目的操作数信息)在往GPRs 32的相应单元回写之前输送到某一选定的重命名缓冲器中存放起来的电路。

处理器10支持这种重命名,而不管Move From指令的目的寄存器是否是GPRs 32之一。通过这种方式,处理器10使用其备用站中已有的转送电路,连同用于搜寻重命名缓冲器的电路,以正确地将(Move From指令的)目的操作数信息与Move From指令之后调度的一条指令的某个源寄存器进行匹配。因此,其它位于Move From指令之后的指令可以在该指令调度完之后的任何时间内进行调度。对于Move From指令来说,如果不使用这种电路,在该指令完成之

前, 其它位于该指令之后的指令都将无法进行调度, 因为其它指令可能需要使用该指令的目的操作数信息(作为它们的源操作数信息)。

本较佳实施例的另外一个显著特征是, 处理器 10 通过使用一种便利的方式响应与取指和解码相关的异常 (“IFDRE”), 从而将指令的调度过程进行流水线式调度。如果异常条件完全可以从指令和处理器 10 的状态中检测到, 处理器 10 将在取指或解码阶段检测到一个与取指和解码相关的异常。这种与取指和解码相关的异常的例子有指令访问缺页、指令访问违反存贮保护、违犯特权指令和非法指令。

较为有利的是, 定序部件 18 根据对这种与取指或解码相关的异常的检测将引起 IFDRE 的指令调度到某个执行部件的一个备用站中, 并将其 EOK 位清为逻辑 0 (如前面结合图 2-4 所详细讨论的那样)。此外, 定序部件 18 还通过将排序缓冲器 76 中与该引起 IFDRE 指令相关联的单元里的“异常”域设置为 1 (同时保持“完成”域为 0) 的方式来保存 IFDRE 条件的指示。这种指示指出该指令是一条引起 IFDRE 的指令。

如前结合图 2-4 所述, 通过将 EOK 位清为逻辑 0, 定序部件 18 可以阻止执行部件执行已经调度给它的指令, 即使指令已经准备就绪也不例外。因此, 执行部件只有在定序部件 18 通过线路 60、86、88、90、92 或 94 中一条与其相连的线路输出一个信号时, 才能执行该指令。

但是, 在引起 IFDRE 的指令的所有前驱指令都已完成之后, 定序部件 18 将直接处理这一与取指或解码相关的异常, 而不是通过相应相连的线路 60、86、88、90、92 或 94 中的一条向执行部件发出一个

信号。因此,引起 IFDRE 的指令从不由执行部件来执行。根据排序缓冲器 76 中与引起 IFDRE 指令的相关联的单元“异常”=1 且“完成”=0,定序部件 18 即认为该指令是一条引起 IFDRE 的指令。

在这种方式中,处理器 10 使用调度逻辑 74(图 3)来流水化调度和加速指令的调度过程。调度逻辑在一超标量处理器中是速度方面极为关键的电路。因此,本较佳实施例中的处理器 10 能够获得比其他设计方案要高的性能。根据其它的方案,处理器从不调度引起 IFDRE 的指令。这类替代方案的调度过程通常都很复杂并且速度很慢,因为对每条指令来说,在决定是否调度它之前,处理器必须检测所有与取指和解码相关的异常情况。例如在这种方案中,处理器必须根据第 n 条指令或前面 $n-1$ 条前驱指令是否有一与取指或解码相关的异常情况来决定是否调度第 n 条指令。

比较起来,在本较佳实施例中,调度逻辑 74(图 3)能够独立地来处理与取指或解码相关的异常情况。即使最后已经决定将指令调度到执行部件,定序部件 18 还要判断该指令是否存在一 IFDRE 情况。如果定序部件 18 断定该指令中存在 IFDRE 情况,它将输出一个关于 IFDRE 的信号,以禁止执行部件对于该指令的执行。具体来说,一旦定序部件 18 断定存在一个有关该指令的 IFDRE 情况,它将在实际调度过程中通过下述手段输出一个这样的信号:(1)将排序缓冲器 76 中与该引起 IFDRE 的指令的相关的单元里的“异常”域置为 1(同时保持“完成”=0);(2)将该引起 IFDRE 的指令的备用站中的 EOK 位清为逻辑 0。

这一点是很有好处的,因为在决定调度某条指令之后,如果再想挽回这一决定所造成的影响并撤销对该指令的调度,通常是不现实

的。另外，在调度这个周期里，决定调度该指令之后还会执行一些附加的操作。因此，处理器 10 在决定是否调度某条指令之前勿需去检测任何异常情况。这对于定序部件 18 希望在处理器 10 的单一周期中同时调度多条指令来说是特别有用的。

图 6 是排序缓冲器 76 的概念性示意图，其中给出了四条指令，这四条指令在处理器 10 的同一周期里完成了指令的执行阶段（“完成”=1）。图 7 是图 6 中四条指令的各个阶段的示意图。图 8a-d 是处理器 10 中重命名缓冲器 34 的概念性示意图。

参照图 6-8，本较佳实施例的一个显著特征是，回写和指令的完成是相互独立的，故由处理器 10 所处理的指令的回写阶段和完成阶段是可以相互分开的。通过这种把回写和完成分开的方式，处理器 10 在重命名缓冲器和体系寄存器之间使用较少的回写端口即可获得很高的执行效率。例如，如图 1 所示，本较佳实施例的处理器 10 在重命名缓冲器 34 和 GPRs 32 之间及重命名缓冲器 38 和 FPRs 36 之间各包含两个回写端口。由于使用了较少的回写端口，这样就缩小了重命名缓冲器 34 和 38 及体系寄存器 32 和 36 的物理尺寸。此外，完成逻辑 80 (图 3) 更为流畅，从而处理器 10 可以更快地判定一条特定的指令是否能在当前指令周期“完成”。

在本较佳实施例中，处理器 10 每个周期可以“完成”多达四条指令，并且每条指令可以拥有多达两个目的操作数。因此，如果处理器 10 不支持回写和完成的分离，为能在某一周期同时完成四条指令（如果每条指令有两个目的操作数），它必须（在重命名缓冲器 34 和 GPRs 32 之间）设置八个回写端口。这是因为一条指令的“完成”需要为该指令的每个目的操作数设置一个回写端口，这些操作数需从重

命名缓冲器中复制到某一与其关联的体系寄存器中。

在具有较少的回写端口的情况下，当需要在同一周期考虑更多指令的回写时，回写端口的有效性验证问题就变得更为复杂。这是因为，在某一周期里，对某一指令来说，一个回写端口的有效性依赖于在本周期或先行周期中用于前驱指令的回写端口的个数。

将回写和完成分开之后，处理器 10 中的完成逻辑 80 (图 3) 就变得更加流畅了。这是因为，一条指令的“完成”依赖于下面的条件：

条件 1: 执行部件已完成了该指令的执行过程；

条件 2: 在处理该指令的每个阶段都没有遇到任何异常；以及

条件 3: 所有前面已调度的指令都满足条件 1 和条件 2。

在完成和回写分开的情况下，于某一周期中，如果有回写端口可供使用，则处理器 10 将一条已完成的指令的目的操作数从重命名缓冲器复制到某一体系寄存器中保存起来。如果在该周期中没有可以使用的回写端口，那么处理器 10 将推迟这一复制过程，直至在稍后的某个周期里有一回写端口可用时为止。

参照图 6，排序缓冲器 76 中保存了在处理器 10 的同一周期里执行完成的四条指令的一些相关信息。图 7 给出了分别与图 6 中排序缓冲器 7、8、9 和 10 相关联的四条指令 INST X、INST X + 1、INST X + 2 和 INST X + 3 的各个阶段。从图 6 中可以看出，指令 INST X 和指令 INST X + 1 都只有一个目的操作数（“目的 GPR 数” = 1），而指令 INST X + 2 和指令 INST X + 3 各拥有两个目的操作数（“目的 GPR 数” = 2）。如图 7 所示，指令 INST X、INST X + 1、INST X + 2 和 INST X + 3 都在周期 4 结束时完成其执行过程。

图 8a - d 是处理器 10 中重命名缓冲器 34 的概念性示意图。为

清楚起见, 下面的讨论仅以定点重命名缓冲器 34 的操作为例, 对于浮点重命名缓冲器 38 的情况可以以此类推。如图 8a - d 所示, 重命名缓冲器 34 包括 12 个单元, 分别按 0 到 11 编号。定序部件 18 本着先进先出的原则以循环方式为已调度的指令分配重命名缓冲器的各个单元, 即首先分配 0 号单元, 接着按顺序分配 1 到 11 号单元, 然后回起来再分配 0 号单元。

参照图 8a - d, 重命名缓冲器中的单元 2 和单元 3 已分别分配给指令 INST X 和 INST X + 1, 用以存放目的操作数信息。因为指令 INST X + 2 有两个目的操作数, 所以给它分配了 4 和 5 两个单元用以存放目的操作数信息。类似地, 给指令 INST X + 3 分配了 6 和 7 两个单元。

图 8a 指出在图 7 的周期 4 和周期 5 开始时分配指针 80、回写指针 82 和完成指针 84 的状态。处理器 10 设置这些指针的目的是为了更好地控制重命名缓冲器 34 的读写过程。分配指针 80 用以指示一个缓冲单元是否已分配给某一特定的指令。如图 8a 所示, 分配指针 80 指向第 8 号单元, 因此 8 号单元即是下一个将要分配给某一指令的缓冲单元。

本较佳实施例的一个显著特征是, 处理器 10 还维护回写指针 82 用以指示一个重命名缓冲单元 (前面已分配给某一指令) 是否可以再分配给另外一条指令。如图 8a 所示, 回写指针 82 指向单元 2, 因此处理器 10 下一个将从单元 2 中将目的操作数信息 (存放在图 8a 中重命名缓冲器的“信息”域里) 复制到 GPRs 32 的某一单元里 (如图 8a 中重命名缓冲器的“寄存器号”域所指出的)。

相应地, 处理器 10 将指令的结果 (目的操作数信息) 从重命名缓

冲器复制到体系寄存器之后,即将回写指针 82 进行移动(越过前面已经分配给某条指令的一个命名缓冲单元)。通过这种方式,处理器 10 保留一个已分配的重命名缓冲单元用以存放相应指令的结果(目的操作数信息),直到它将该结果复制到某一体系寄存器里为止。

另外,处理器 10 还维持完成指针 84 用以指示(对先前已经分配给某一指令的一个重命名缓冲单元)相应的指令是否满足下述条件:

条件 1: 相应的执行部件已经完成该指令的执行过程;

条件 2: 在处理该指令的各个阶段都没有遇到任何异常;以及

条件 3: 所有先前已调度的指令都满足条件 1 和条件 2。

如图 8a 所示,完成指针 84 指向重命名缓冲器的单元 2,因此单元 2 即是下一个可以满足条件 1、2 和 3 的重命名缓冲单元。本较佳实施例的一个显著特点是,处理器 10 维护完成指针 84 时,不受指令的结果是否已从重命名缓冲器复制到体系寄存器这一条件的制约。

因此,“重命名项”可以定义为从完成指针 84 所指的缓冲器单元以及该重命名缓冲单元之前由分配指针 80 所指的另一重命名缓冲器单元。“回写项”可以定义为从回写指针 82 所指的缓冲器单元以及该重命名缓冲单元之前完成指针 84 所指的另一重命名缓冲器单元。回写项存放那些已经“完成”但其结果尚未从重命名缓冲器复制到体系寄存器(例如由于通向体系寄存器的写端口无一可用)的指令的结果。

从概念上讲,回写项位于重命名项和体系寄存器之间。非常有用的是,如果一条指令在完成阶段发现有一回写端口可用,其结果可以越过回写项而直接写入体系寄存器。另外,如重命名项一样,如果一个执行部件正在执行一条指定了某一体系寄存器的指令,并且该寄

寄存器与相应回写项中的信息有关联, 则处理器 10 将操纵回写项往相应的执行部件中输出该信息。

例如, 图 8b 给出了在图 7 的周期 6 开始时分配指针 80、回写指针 82 和完成指针 84 的状态。如图 8b 所示, 分配指针 80 仍未变化, 因为处理器 10 尚未调度新的指令。但完成指针已从单元 2 往前移动到了单元 8, 这表明在周期 5 中完成了总共有六个目的操作数的四条指令 INST X、INST X + 1、INST X + 2 和 INST X + 3。

另外, 在图 8b 中回写指针 82 从单元 2 往前移动到了单元 4, 这表明在周期 5 中已为指令 INST X 和 INST X + 1 完成了目的操作数信息的回写过程。图 7 中“完成/回写”表示指令 INST X 和 INST X + 1 的完成和回写过程都在周期 5 中进行。通过这种方式, 指令 INST X 和指令 INST X + 1 的结果(在重命名缓冲器的 2 号和 3 号单元里)可以越过回写项而直接写入 GPRs 32, 因为在周期 5 开始时有两个回写端口可以使用。在图 8b 中, 回写指针 82 没有移动超过重命名缓冲单元 4, 因为在周期 5 中两个回写端口都要用于指令 INST X 和 INST X + 1 的结果的回写过程。

图 8c 给出了图 7 中分配指针 80、回写指针 82 和完成指针 84 在周期 7 开始时的状态。如图 7 和 8c 所示, 在周期 6 中, 两个回写端口都用来回写指令 INST X + 2 的两个结果。因此, 回写指针 82 从单元 4 移到了单元 6。分配指针 80 仍保持不动, 因为处理器 10 尚没有调度新的指令。同样, 完成指针 84 也没有变化, 因为处理器 10 此间没有完成另外的指令。

图 8d 给出了图 7 中分配指针 80、回写指针 82 和完成指针 84 在周期 8 开始时的状态。如图 7 和 8d 所示, 在周期 7 中, 两个回写端口

都用来回写指令 $\text{INST } X + 3$ 的两个结果。因此，回写指针 82 从单元 6 移到了单元 8。分配指针 80 仍保持不动，因为处理器 10 尚没有调度新的指令。同样，完成指针 84 也没有变化，因为处理器 10 尚没有完成新的指令。

图 9 是在不具备本较佳实施例的显著特征的情况下，四条指令的各个阶段的示意图。其中处理器 10 分开处理完成和回写阶段，指令 $\text{INST } Y$ 、 $\text{INST } Y + 1$ 、 $\text{INST } Y + 2$ 和 $\text{INST } Y + 3$ 分别与图 6 中排序缓冲单元 7、8、9 和 10 相关联。

图 10a - d 是依据图 9 的一个处理器的重命名缓冲器的概念性示意图。具体来说，图 10a 至图 10d 分别给出了图 9 中分配指针 90 和完成指针 94 在周期 4 和 5 开始时、在周期 6 开始时、在周期 7 开始时及在周期 8 开始时的状态。

如图 10a - d 所示，在不具备本较佳实施例的显著特征且处理器 10 分开处理完成和回写阶段的情况下，一条指令（拥有一个已分配给它的重命名缓冲器）只有在其结果确实已从相应的重命名缓冲器转贮到某一体系寄存器后才算完成。比较起来，在具备本较佳实施例的显著特征且处理器 10 分开处理完成和回写阶段的情况下，处理器 10 另外还维护一个回写指针 82 用以指示一个重命名缓冲单元（先前已分配某一指令）是否可以再分配给另外一条指令。此外，处理器 10“完成”一条指令时并不考虑该指令的结果是否确实已从相应的重命名缓冲器转贮到某一体系寄存器。因此，处理器 10 仅在一个周期中就可“完成”多至四条指令，即使每条指令可能有两个目的操作数，以及较少目的操作数在该周期内转贮到 GPRs 32 中。

图 11 是重命名缓冲器 34 的一个可选实施例的结构图，其中“回

写项”与“重命名项”分开存放在缓冲器 110 中,而“重命名项”存放在缓冲器 112 中。在某一特定的周期中,为响应某条指令(为该指令分配一“重命名项”)的完成,缓冲器 112 将分配给该指令的“重命名项”中的信息转贮到缓冲器 110 的某一“回写项”中。在某一特定的周期中,为响应该周期中通向 GPRs 32 的两个回写端口中某一端口的可用性,缓冲器 110 将“回写项”中的信息转送到 GPRs 32 中的某一单元里存放起来。

如图 11 所示,缓冲器 112 可以通过八个端口中的任一个向缓冲器 110 输出信息,这样,缓冲器 112 能在处理器 10 的任何一个周期中从最多可达八个的“重命名项”向缓冲器 110 输送信息。因此,处理器 10 在某一周期中能同时完成最多可达四条的指令,即使每条指令都可能有两个目的操作数以及只有较少的目的操作数在该周期内转贮到 GPRs 32 中。比较有利的是,如果在完成阶段有一回写端口可用,结果(目的操作数信息)可以越过缓冲器 110 的“回写项”而直接写到 GPRs 32 中。如图 11 所示,处理器 10 通过适当地操纵多路转换器 113a 和 113b 来实现这一越过。这两个多路转换器按图 11 中的方式与缓冲器 110 和 112 及 GPRs 32 相连。

图 11 中重命名缓冲器 34 的另一实施例的一个优点是在完成了(仍在回写之前)先前与缓冲器 112 中某一重命名项相关的某一指令(先前已将该重命名项分配给该指令)之后,该重命名项可以再分配给另外一条指令,因为在先前关联的指令完成时,重命名项中的信息即刻便转贮到缓冲器 110 中的回写项中了。另一方面,在图 11 中,如果某一执行部件执行一条指定了某一体系寄存器的指令,则多路转换器 114 会适时地从缓冲器 110 向该执行部件输出选定的与相应体

系寄存器相关的信息。虽然本较佳实施例中处理器 10 在重命名缓冲器和体系寄存器之间设置了两个回写端口，但对一个具体的实施例来说，回写端口的合适数量其实是能使得重命名项和回写项变满从而延迟指令调度的一个似然函数。

如前面结合图 2-5 所仔细讨论的那样，处理器 10 在两种情况下至少将一条串行化指令的执行过程延迟至其所有的前驱指令都已执行完为止。第一种情况是指令不能按照随机方式来执行。第二种情况是指令至少指定了一个处理器 10 没有为之提供转送电路的源操作数。

关于第一种情况，有些指令对专用体系寄存器 SPRs 40 (图 1) 进行操作，指令的结果直接写入 SPRs 40，中间勿需再经重命名缓冲器缓存。关于这种指令的一个例子是一条 Move To 指令，该指令将信息从 GPRs 32 中的某一单元转送到 SPRs 40 中的某一单元。如图 1 所示，该 Move To 指令由 CFXU 26 来执行，它在执行时即时更新 SPRs 40 中的某一单元。类似地，一条 Store 指令在执行时即时更新数据高速缓存 16 中的某一存储单元。处理器 10 并不试探地去执行这种类型的指令（这类指令结果不需经过重命名缓冲器的暂存而直接写至体系寄存器或内存单元），以便能够实现精确的中断和异常。因此，为确保按顺序完成和回写，处理器 10 将延迟 Move To 指令或 Store 指令的执行，直到其所有的先行指令完成时为止。

如果与调度给某个执行部件的指令相关联的 EOK 位（在排序缓冲器 76 中与该指令相关联的单元的“指令类型”域）清成了逻辑 0，那么在该指令的所有先行指令都已完成后执行串行化逻辑 84 将通过与该执行部件相连的相应线路 60、86、88、90、92 或 94 输出一个

信号。执行部件只有在执行串行化逻辑 84 输出该信号时才执行该指令。

图 12a - c 是排序缓冲器 76 的概念性示意图。图 13 是图 12a - c 中五条指令的各个阶段的示意图。图 14a - f 是处理器 10 中重命名缓冲器 34 的概念性示意图。

图 12a 给出了图 13 中周期 4 开始时排序缓冲器 76 的状态。因此, 在图 12a 中, 排序缓冲器 76 为图 13 中周期 3 期间所调度的四条指令存贮了信息。图 13 中给出了 INST a、INST a + 1、INST a + 2 和 INST a + 3 四条指令的各个阶段, 这四条指令分别与图 12a 中的排序缓冲器单元 3、4、5 和 6 相关联。

如图 12a 所示, 指令 INST a 被调度给 FXUA 22, 它有两个目的操作数 (“目的 GPR 数” = 2)。指令 INST a + 1 被调度给 FXUB 24, 它有一个目的操作数 (“目的 GPR 数” = 1)。指令 INST a + 2 是一条存数指令, 它被调度给 LSU 28 并且没有目的操作数 (“目的 GPR 数” = 0); 另外, 一旦指令 INST a + 2 被串行化执行, 排序缓冲器单元 5 (与指令 INST a + 2 相关联) 的 EOK 位即被清零。指令 INST a + 3 被调度给 CFXU 26, 它有一个目的操作数 (“目的 GPR 数” = 1)。

，本较佳实施例的一个显著特征是由定序部件 18 来判断一条指令的执行是否会引起某种异常。定序部件 18 在指令执行之前做出这一判断。如果一条指令不可能引起异常, 定序部件 18 不无益处地置与该指令相关的排序缓冲器单元的 “完成” 为 1 (在调度该指令时), 而不管处理器 10 是否确实已完成了该指令的执行。

另外, 在这种情况下, 处理器 10 完成所有的先行指令后即 “完成” 该指令, 而不管它是否确实已执行完这一不可能引起异常的指

令。因此,在判定一条指令及其编程序列中所有的先行指令都不会引起异常后,处理器 10 即“完成”该指令,而不管它是否已执行完这些不可能引起异常的指令。所以,这种指令的完成阶段完全有可能先于其执行阶段;处理器 10 通过这种方式来支持这类指令的“提前完成”。

通过支持“提前完成”,处理器 10 能够更快地来执行其后的执行串行化指令(如 Move To 指令或 Store 指令),该指令的结果无需经过中间重命名缓冲器的缓存而直接写入某一体系寄存器或存储单元。这一点并不会降低处理器 10 捕获精确中断和异常的能力。这是因为所有的先行指令要么(1)已经没有异常地执行完毕,要么(2)将要没有异常地执行完。处理器 10 使用这种技术继续确保按顺序的完成和顺序的回写。

例如,在图 12a 中,指令 INST $a+1$ 和 INST $a+3$ 不可能引起某一异常,故而排序缓冲单元 4 和 6 中的“完成”皆为 1。图 14a 给出了图 13 中周期 4 开始时重命名缓冲器 34 的分配指针(AL) 80、回写指针(WB) 82 和完成指针(CO) 84 的状态。故图 14a 对应于图 12a。如图 14a 所示, WB 82 和 CO 84 指向单元 2, AL 80 指向单元 6。单元 2 和 3 分配给了指令 INST a 。单元 4 和 5 分别分配给指令 INST $a+1$ 和 INST $a+3$ 。值得注意的是,这里没有给指令 INST $a+2$ 分配重命名缓冲单元,因为指令 INST $a+2$ 没有目的操作数。在单元 2-5 中,“信息有效”=0,说明这些缓冲单元的“信息”域没有存放有效数据。在周期 4 开始时(图 13)“信息有效”=0 是因为指令 INST a 、INST $a+1$ 、INST $a+2$ 和 INST $a+3$ 还没有执行完毕。

在重命名缓冲单元 4 中,“进位有效”=1 说明 INST $a+1$ 修改了

CA 寄存器 42。值得注意的是, 如前所述, 指令 INST $a+1$ 不会引起某一异常, 故排序缓冲单元 4 (图 12a) 中的“完成” = 1。因此, 处理器 10 支持修改 CA 寄存器 42 的指令的“提前完成”。

如图 14a - g 所示, CA 位信息暂时存放在重命名缓冲器 34 中。但是, 本较佳实施例中的处理器 10 并没有包含用于将这种 CA 位信息从重命名缓冲器 34 传送到执行部件的电路。如果一条指令的源操作数信息包括 CA 寄存器 42, 那么该指令具有一个依赖于一条修改 CA 寄存器 42 的先行指令(如果有的话)的结果的源操作数信息。

图 12b 给出了图 13 中周期 5 开始时排序缓冲器 76 的状态, 其中图 13 中周期 4 期间调度的指令 INST $a+4$ 的信息存放在排序缓冲器 76 中。图 13 给出了指令 INST $a+4$ 的各个阶段, 该指令与排序缓冲单元 7 相关联。如图 12b 所示, 指令 INST $a+4$ 调度给了 FXUA 22 并且有一个目的操作数 (“目的 GPR 数” = 1); 另外, 排序缓冲单元 7 (与指令 INST $a+4$ 相关联) 置其 EOK = 0 以响应指令 INST $a+4$ 被执行串行化。

另外, 因为指令 INST a 在周期 4 中执行完毕, 所以图 12b 中排序缓冲单元 3 的“完成” = 1。参见图 13, 指令 INST $a+2$ 是一条 Store 指令。在周期 4 中, LSU 28 完成了该指令的第一个执行阶段 EXEC A。在 EXEC A 阶段, LSU 28 翻译该指令并对其进行存储保护检查。因此, 图 12b 中排序缓冲单元 5 的“完成” = 1。

图 14b 给出了图 13 中周期 5 开始时重命名缓冲器 34 的分配指针 (AL) 80、回写指针 (WB) 82 和完成指针 (CO) 84 的状态。故图 14b 对应于图 12b。如图 14b 所示, WB 82 和 CO 84 继续指向单元 2, 而 AL 80 已从单元 6 向前移动到了单元 7。因此, 单元 6 也就分配给了

指令 INST $a + 4$ 。

在重命名缓冲单元 6 中,“信息有效” = 0 说明该单元的“信息”域尚未存放有效数据。在周期 5 (图 13) 开始时,单元 6 中的“信息有效” = 0 是因为指令 INST $a + 4$ 还没有执行完。另一方面,因图 13 中指令 INST a 和 INST $a + 1$ 已在周期 4 中执行完,重命名缓冲单元 2 和 3 中“信息有效” = 1,这表明单元 2 和 3 的“信息”域里存放了有效数据(在图 14b 的“信息”域中表示为“DATA”)。

在重命名缓冲单元 6 中,“进位有效” = 1 说明 INST $a + 4$ 修改了 CA 寄存器 42。此外,指令 INST $a + 4$ 的源操作数信息包含 CA 寄存器 42,从而 INST $a + 4$ 是一条执行串行化指令,该执行串行化指令具有一个依赖于修改了 CA 寄存器 42 的先行指令 INST $a + 1$ 的结果的源操作数信息。虽然指令 INST $a + 3$ 并没有修改 CA 寄存器 42,但它同样具有一个依赖于先行指令 INST $a + 1$ 结果的源操作数信息,如图 13 中的箭头 100 所指。

图 12c 给出了图 13 中周期 6 开始时排序缓冲器 76 的状态。图 14c 给出了图 13 中周期 6 开始时重命名缓冲器 34 中分配指针(AL) 80、回写指针(WB) 82 和完成指针(CO) 84 的状态。故图 14c 对应于图 12c。

如图 12c 所示,响应图 13 中指令 INST a 、INST $a + 1$ 、INST $a + 2$ 和 INST $a + 3$ 在周期 5 中“完成”,排序缓冲单元 3、4、5 和 6 中的信息已被删除。值得注意的是,如图 13 中由“提前完成/执行”所示,在周期 5 中,指令 INST $a + 1$ 的“提前完成”要先于它的结束执行;指令 INST $a + 1$ 是需要处理器 10 用多于一个周期的时间来执行的一类指令中的一员。

参看图 14c, 处理器 10 为响应所有先行指令 (如 INST a、INST a + 1 和 INST a + 2) 的完成, 将 CO 84 从先前分配给某一指令 (如 INST a + 3) 的重命名缓冲单元向前移动一个单元 (如在周期 6 开始之前从单元 5 向前移动), 而不管处理器 10 是否确实已执行完该指令 (如 INST a + 3 继续在周期 6 - 8 中执行)。

此外, 在周期 5 中, 执行串行化逻辑 84 (图 3) 在指令 INST a 和 INST a + 1 完成时通过与 LSU 28 相连的线路 92 输出一个信号; 这是由于指令 INST a + 2 (已在周期 3 调度给 LSU 28) 在排序缓冲单元 5 中相关联的 EOK 位已被清为逻辑 0。

在周期 5 中, 存数指令 a + 2 因其已经无异常地执行完阶段 EXEC A (周期 4) 且其先行指令 INST a 和 INST a + 1 也已完成, 其自身也可“完成”。在执行的第二阶段 EXEC B, LSU 28 才确实将信息写入数据高速缓存 16 的某一存储单元。LSU 28 只有在执行串行化逻辑 84 通过线路 92 输出相应的信号后才执行 Store 指令 INST a + 2 的执行阶段 EXEC B。

如图 14c 所示, AL 80 继续指向重命名缓冲单元 7, CO 84 已因先行指令 INST a、INST a + 1 和 INST a + 3 在周期 5 中“完成”而从单元 2 向前移动到了单元 6。另外, 如图 13 所示, 指令 INST a 的两个结果已在周期 5 中从重命名缓冲单元 2 和 3 复制到了 GPRs 32 (图 1)。因此, 如图 14c 所示, WB 82 已从单元 2 向前移动指向了单元 4; 并且, 因指令 INST a 的两个结果已在周期 5 从单元 2 和 3 复制到了 GPRs 32 (图 1), 故此处删去了单元 2 和 3 中保存的信息。

在图 14c 中, 重命名缓冲单元 4 的“信息有效” = 1 以响应图 13 中指令 INST a + 1 在周期 5 中执行完毕。这表明单元 4 的“信息”域和

“进位”域中都存放了有效数据（在图14c中分别以“信息”域里的“DATA”和“进位”域里的“CA”来表示）。因为 $INST\ a+4$ 是一条执行串行化指令，该指令具有一个依赖于修改 CA 寄存器 42 的指令 $INST\ a+1$ 的结果的源操作数信息，并且重命名缓冲器 34 中的“进位”域尚未向前传送，所以处理器 10 延迟执行指令 $INST\ a+4$ ，直到在周期 6 中指令 $INST\ a+1$ 的回写阶段 (WBACK) 其“进位”结果已从重命名缓冲单元 4 复制到 GPRs 32 (图 1) 时为止。

作为对比，指令 $INST\ a+3$ 也有一个依赖于指令 $INST\ a+1$ 的 DATA 结果的源操作数信息，但不同的是重命名缓冲器 34 中的该“信息”域已传送了。因此，为响应重命名缓冲单元 4 中的“信息有效”= 1 (表明指令 $INST\ a+1$ 在周期 5 中执行完毕)，处理器 10 在指令 $INST\ a+1$ 的回写阶段完成之前于周期 6 中开始指令 $INST\ a+3$ 的执行过程。

图 14d 给出了图 13 中周期 7 开始时重命名缓冲器 34 中分配指针 (AL) 80、回写指针 (WB) 82 和完成指针 (CO) 84 的状态。如图 14d 所示，AL 80 继续指向单元 7，CO 84 继续指向单元 6。如图 13 所示，指令 $INST\ a+1$ 的结果已在周期 6 期间从单元 4 复制到了 GPRs 32 (图 1)。因此，如图 14d 所示，WB 82 已从单元 4 向前移动指向了单元 5；此外，因指令 $INST\ a+1$ 的结果已在周期 6 期间从单元 4 复制到了 GPRs 32 (图 1)，故此处删去了单元 4 中保存的信息。

在本较佳实施例中，对于一条具有一个依赖于某个尚未向前传送的操作数 (如 CA 位信息的目的地址是 CA 寄存器 42) 的源操作数信息的执行串行化指令 (如 $INST\ a+4$)，处理器 10 在重命名缓冲器 34 包含以下信息时即执行该执行串行化指令 (如 $INST\ a+4$)：

(1) 所有的先行指令都已“完成”；并且

(2) 对任何已“完成”的先行指令来说，如果其回写阶段尚未结束，则不修改上述未传送的操作数。

因此，在图 14d 中，CO 84 指向单元 6 表明对指令 INST $a+4$ 来说其所有的先行指令都已“完成”。此外，WB 82 指向单元 5 表明虽然已“完成”的先行指令 INST $a+3$ 的回写阶段尚未结束，但指令 INST $a+3$ 并没有修改任何将要传送到 CA 寄存器 42 的未传送信息（因为单元 5 中的“进位有效”= 0）。于是，处理器 10 在周期 7 中执行指令 INST $a+4$ 。

在一个可选的实施例中，对于一条具有一个依赖于某个尚未向前传送的操作数（如 CA 位信息的目的地是 CA 寄存器 42）的源操作数信息的执行串行化指令（如 INST $a+4$ ），处理器 10 在重命名缓冲器 34 包含以下信息时即执行该指令（如 INST $a+4$ ）：

(1) 所有的先行指令都已“完成”；并且

(2) 所有已“完成”的先行指令的回写阶段都已结束，从而 WB 82 和 CO 84 指向同一个重命名缓冲单元。

图 14e 给出了图 13 中周期 8 开始时重命名缓冲器 34 中分配指针 (AL) 80、回写指针 (WB) 82 和完成指针 (CO) 84 的状态。如图 14e 所示，AL 80 继续指向单元 7，WB 82 继续指向单元 5。在周期 7 中，指令 INST $a+4$ 结束执行要先于其“完成”，如图 13 中“执行/完成”所示。因此，如图 14e 所示，CO 84 已从单元 6 向前移动指向了单元 7。

比较有利的是，处理器 10 为响应所有先行指令（如 INST a 、INST $a+1$ 、INST $a+2$ 和 INST $a+3$ ）的完成，将 CO 84 从先前分配

给某一指令（如 INST $a + 4$ ）的重命名缓冲单元向前移动一个单元（如在周期 8 开始之前从单元 6 向前移动），而不管处理器 10 是否确实已执行完所有的先行指令（如 INST $a + 3$ 继续在周期 8 中执行）。

在图 14e 中，重命名缓冲单元 6 中“信息有效”= 1 以响应图 13 中周期 7 期间指令 INST $a + 4$ 结束执行。这表明单元 6 的“信息”域和“进位”域中都存放了有效数据（在图 14e 中分别以“信息”域里的“DATA”和“进位”域里的“CA”来表示）。

图 14f 给出了图 13 中周期 9 开始时重命名缓冲器 34 中分配指针 (AL) 80、回写指针 (WB) 82 和完成指针 (CO) 84 的状态。如图 14f 所示，AL 80 和 CO 84 继续指向单元 7，WB 82 继续指向单元 5。在图 14f 中，单元 5 里“信息有效”= 1 以响应指令 INST $a + 3$ 于周期 8 结束执行。这表明单元 5 的“信息”域里存放了有效数据（在图 14e 中以“信息”域里的“DATA”来表示）。

图 14g 给出了图 13 中周期 10 开始时重命名缓冲器 34 中分配指针 (AL) 80、回写指针 (WB) 82 和完成指针 (CO) 84 的状态。如图 14g 所示，AL 80 和 CO 84 继续指向单元 7。如图 13 所示，指令 INST $a + 3$ 和 INST $a + 4$ 的结果在周期 9 期间分别从单元 5 和 6 复制到 GPRs 32 中（图 1）。因此，如图 14g 所示，WB 82 已从单元 5 向前移动指向了单元 7；此外，因指令 INST $a + 3$ 和 INST $a + 4$ 的结果已在周期 9 从单元 5 和 6 复制到了 GPRs 32（图 1），故此处已删去了单元 5 和 6 中的信息。

图 15 是某一不具备本较佳实施例的“提前完成”特征的另一实施例中图 12a - c 中五条指令的各个阶段的示意图。图 16a - h 是图 15 中所示指令处理的各个周期里处理器 10 的重命名缓冲器 34 的概

念性示意图。如图 15 所示,在不具备本较佳实施例的“提前完成”特征的情况下,指令 INST $a+4$ 的执行要延迟到周期 10,而其结果的完成/回写阶段则要等到周期 11。这是由于指令 INST $a+4$ 是一条执行串行化指令,必须要等到其所有的先行指令完成时方可开始执行。在不具备本较佳实施例的“提前完成”特征的情况下,指令 INST $a+3$ 要等到周期 9 才能完成,故指令 INST $a+4$ 的执行必须得等到周期 10。

而比较起来,在图 13 中,指令 INST $a+3$ 于周期 5“提前完成”,指令 INST $a+4$ 于周期 7 执行。指令 INST $a+4$ 是一条具有一个依赖于其先行指令 INST $a+1$ 的结果的源操作数信息的执行串行化指令,其中指令 INST $a+1$ 修改将要送往 CA 寄存器 42 的未传送 CA 位信息。但是,在图 13 中,指令 INST $a+4$ 的执行发生在周期 7,其结果的完成/回写发生在周期 9。因此,在图 13 中,在具有本较佳实施例的“提前完成”特征的情况下,指令 INST $a+4$ 的执行、完成和回写阶段要比图 15 中没有“提前完成”特征的情况下发生的早。

处理器 10 将一条指定了未传送源操作数(如 SPRs 40 和 CA 寄存器 42)的指令的执行过程至少延迟到其所有的先行指令完成时为止。这是因为,如果该指令指定了未传送源操作数,则它的执行必须要受体系寄存器中的这些源操作数信息可用并且有效的影响。关于这种指令的例子有 (1) 一条读 CA 寄存器 42 的扩展算术指令和 (2) 一条将信息从 SPRs 40 的某个单元传送到 GPRs 32 的某个单元的 Move From 指令。

即使所有的先行指令都已“完成”,对某条先行指令来说也有可能它是“提前完成”的,从而其完成阶段先于其执行阶段。如上面结合

图 1-5 所详细讨论的那样, 处理器 10 包含用于将信息从 GPRs 32、FPRs 36、重命名缓冲器 34 和 38 及控制寄存器传送到执行部件的传送电路。如果一条指令具有一个依赖于某条“提前完成”的先行指令的结果的源操作数信息, 并且该源操作数信息有处理器 10 的传送电路的支持, 那么处理器 10 将延迟该指令的执行, 直到它结束了前述“提前完成”指令的执行阶段为止 (此时该指令的结果已出现在某一重命名缓冲单元中并且有效)。

比较起来, 如果该源操作数信息没有处理器 10 的这种传送电路的支持, 那么处理器 10 将延迟该指令的执行直到它结束了前述“提前完成”指令的回写阶段时为止 (此时“提前完成”指令的结果在体系寄存器中出现并且有效)。如果一条“提前完成”的指令尚未结束执行阶段, 那么该指令的结果 (目的操作数信息) 尚未出现。在这种情况下, 处理器 10 相应地延迟该指令的回写阶段直至它结束执行阶段时为止。

如果一条指令受提前完成的影响, 那么定序部件 18 在调度该指令到与其相关联的执行部件时输出一个指示信号。接到该信号后, 相关联的执行部件不再去修改排序缓冲器 76 中分配给该指令的单元。这是因为, 一旦完成了与排序缓冲单元相关联的指令, 该排序缓冲单元须再分配给另外一条指令, 从而该单元中的信息不再有效。

因此, 如果一条指令受提前完成的影响, 那么在该指令的执行部件结束其执行阶段之前, 分配给该指令的排序缓冲单元可能已经无效。在本较佳实施例中, 于处理器 10 的某一个周期, 定序部件 18 检查具有有效信息的排序缓冲单元中资格最老的四个, 这样就有四条候选指令可以在一个周期内同时完成。

在本较佳实施例中，一条 Move From 指令从某一 SPR 输入信息，一个 SPR 仅能由一条 Move To 指令来修改。比较有利的是，处理器 10 在所有先行指令完成后即执行一条 Move From 指令，即使其中一条或多条先行指令是“提前完成”的。这一点是可能的，因为处理器 10 并不“提前完成”一条不能被随机地执行的指令（如一条 Move To 指令或一条 Store 指令）。如果所有的先行指令都已“完成”，那么处理器 10 确实已结束了所有的先行 Move To 指令的执行阶段。因此，处理器 10 在所有的先行指令完成之后即可开始执行一条 Move From 指令，因为在这种情况下处理器 10 实际上已结束了所有的先行 Move To 指令的执行阶段。

本较佳实施例的另外一个显著特征是，处理器 10 支持一种非精确的不可恢复的浮点异常模式。当在这种模式下运行时，对于引起异常情况的浮点指令处理器 10 以不精确的方式来报告浮点异常情况，处理器 10 在处理这种异常时，允许其自身处于下述状态，即它已完成除引起该异常的浮点指令之外的一定数量的指令。当在这种模式下运行时，对于那些不太关心浮点指令引起异常时中断程序执行的精确位置的指令序列来说，处理器 10 可以获得更高的执行效率。

当在精确的浮点异常模式下运行时，即使所有的先行指令都已完成，处理器 10 也不在结束一条浮点指令的执行阶段之前“提前完成”该指令。

比较起来，当在非精确的不可恢复的浮点异常模式下运行时，处理器 10 在所有的先行指令都已完成后，于一条浮点指令结束执行阶段之前，“提前完成”该指令。在非精确的不可恢复的浮点异常模式下，“提前完成”的浮点指令与其他类型的“提前完成”的指令有所不同。

同, 因为即使一条浮点指令的执行可能会引起一个浮点异常情况, 处理器 10 还是“提前完成”该指令。每个排序缓冲单元都指出与其相关联的指令是否是一条浮点指令。

在一条浮点指令结束其执行和完成阶段之后, 其结果暂存到浮点重命名缓冲器 38 中, 并一直要存到所有前面的浮点指令的结果都已复制到 FPRs 36 时为止。通过这种方式, 处理器 10 按照浮点指令的编程顺序来处理其相应的“回写”阶段。

重命名缓冲器 38 暂存以后将复制到 FPRs 36 的浮点指令的结果。另外, 重命名缓冲器 38 还存放用于适时修改某一体系浮点状态和控制寄存器 (FPSCR) (未给出) 的状态信息。在从重命名缓冲器 38 往 FPRs 36 和 FPSCR 中复制结果期间的检测到一个浮点异常 (由被设置的 FPSCR 中的一个 FEX 位指出) 时, 处理器 10 报告该异常情况。

当在非精确的不可恢复的浮点异常模式下运行时, 对于引起该异常情况的浮点指令处理器 10 以不精确的方式来报告浮点异常情况。在这种情况下, 比较有利的是, 本较佳实施例中的处理器 10 按下述方式报告一个异常。

首先, 处理器 10 在检测到浮点异常的那个周期停止指令的完成。具体在那一条指令处理器 10 停止了完成并不重要, 只要处理器 10 的状态精确反映了所有先于处理器 10 停止完成阶段时的所停止那一指令的已完成指令的效果即可。

第二, 处理器 10 结束所有“提前完成”的先行指令的执行阶段。此外, 处理器 10 还将这些指令的结果复制到体系寄存器中, 从而使重命名缓冲器 34 和 38 中的所有回写项都变成空。

第三, 处理器 10 报告该浮点异常。在非精确的不可恢复的浮点异常模式下, “提前完成”的浮点指令与其它类型的“提前完成”的指令有所不同, 因为即使一条浮点指令的执行可能会引起一个浮点异常情况, 处理器 10 还是“提前完成”该指令。因此, 没有由其他“提前完成”或确实已完成的指令所引起的其他异常产生。

在检测出一个由非浮点指令引起的异常后, 处理器 10 延迟报告这一非浮点异常, 直到它结束了所有“提前完成”的先行浮点指令的执行阶段而没有遇到任何浮点异常时为止。在这种情况下, 比较有利的是, 在本较佳实施例中的处理器 10 按下述方式报告一个异常:

首先, 处理器 10 在检测到非浮点异常的那个周期停止指令的完成。处理器 10 在引起这一非浮点异常的指令处停止指令的完成, 所以该指令尚未完成。

第二, 处理器 10 结束所有“提前完成”的先行指令的执行。这决定处理器 10 是否能够不遇任何浮点异常地结束所有“提前完成”的先行浮点指令。另外, 处理器 10 将这些指令的结果复制到体系寄存器中, 从而使得重命名缓冲器 34 和 38 中的所有回写项都变成空。通过这种方式, 处理器 10 在报告这一异常时仍能维持一种稳定的状态。此外, 在检测出有异常后, 处理器 10 保持稳定的处理器状态一直等待直到其他的回写缓冲器都变成空时为止。

第三, 如果处理器 10 没有检测到浮点异常, 则报告精确的非浮点异常。相反, 如果处理器 10 检测到了一个浮点异常, 则报告非精确的不可恢复的浮点异常。

上面我们仔细描述了本发明的一个示范实施例及其优点。但是应当注意的是这些描述仅仅作为例子而非限制。在不脱离本发明的

外延、范围和精神的情况下，该示范实施例中可以有各种变化、替换和改造。本发明的外延、范围和精神不应受该示范实施例的限制，而应当结合下述权利要求及其等价物来定义。

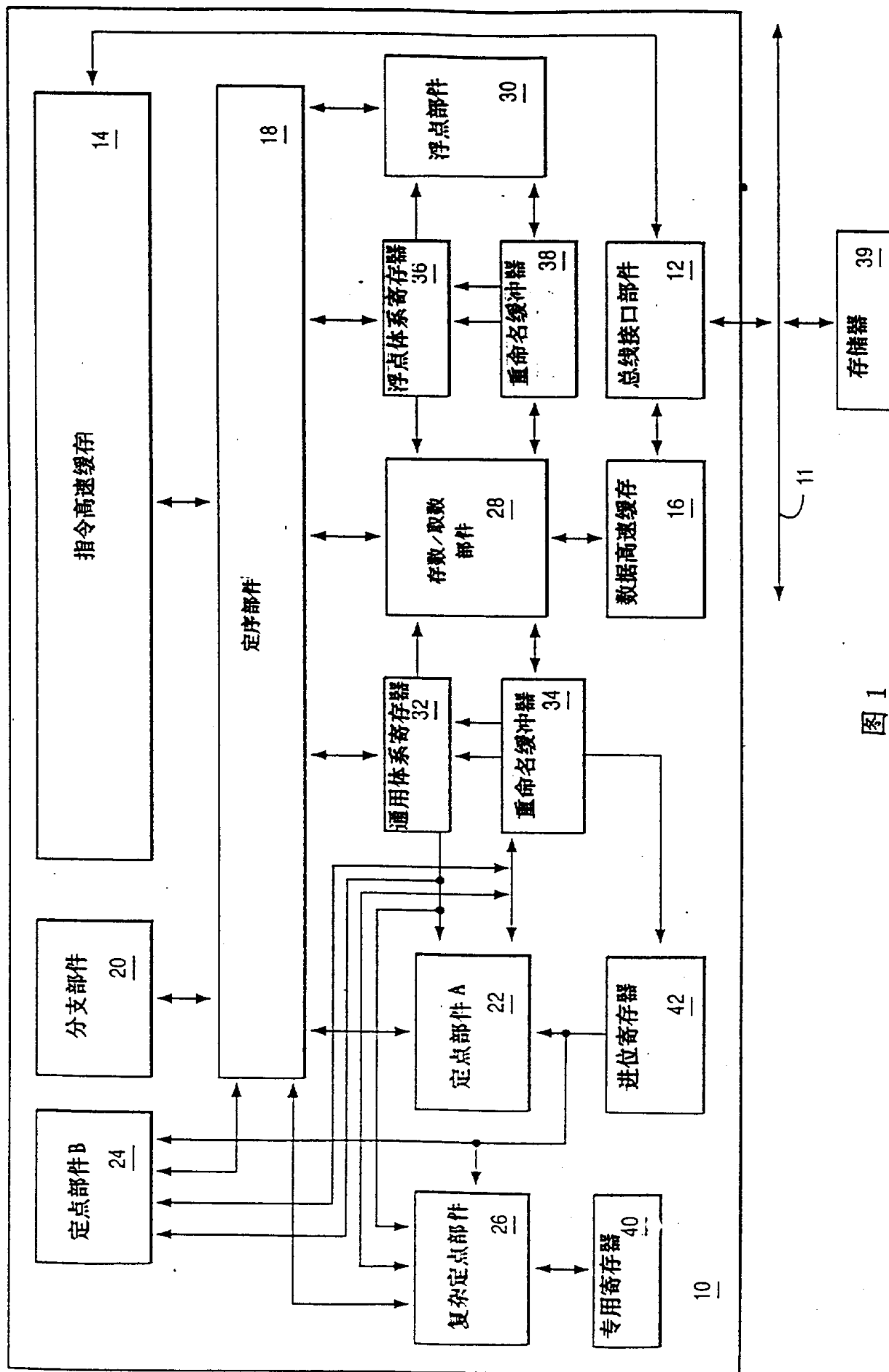
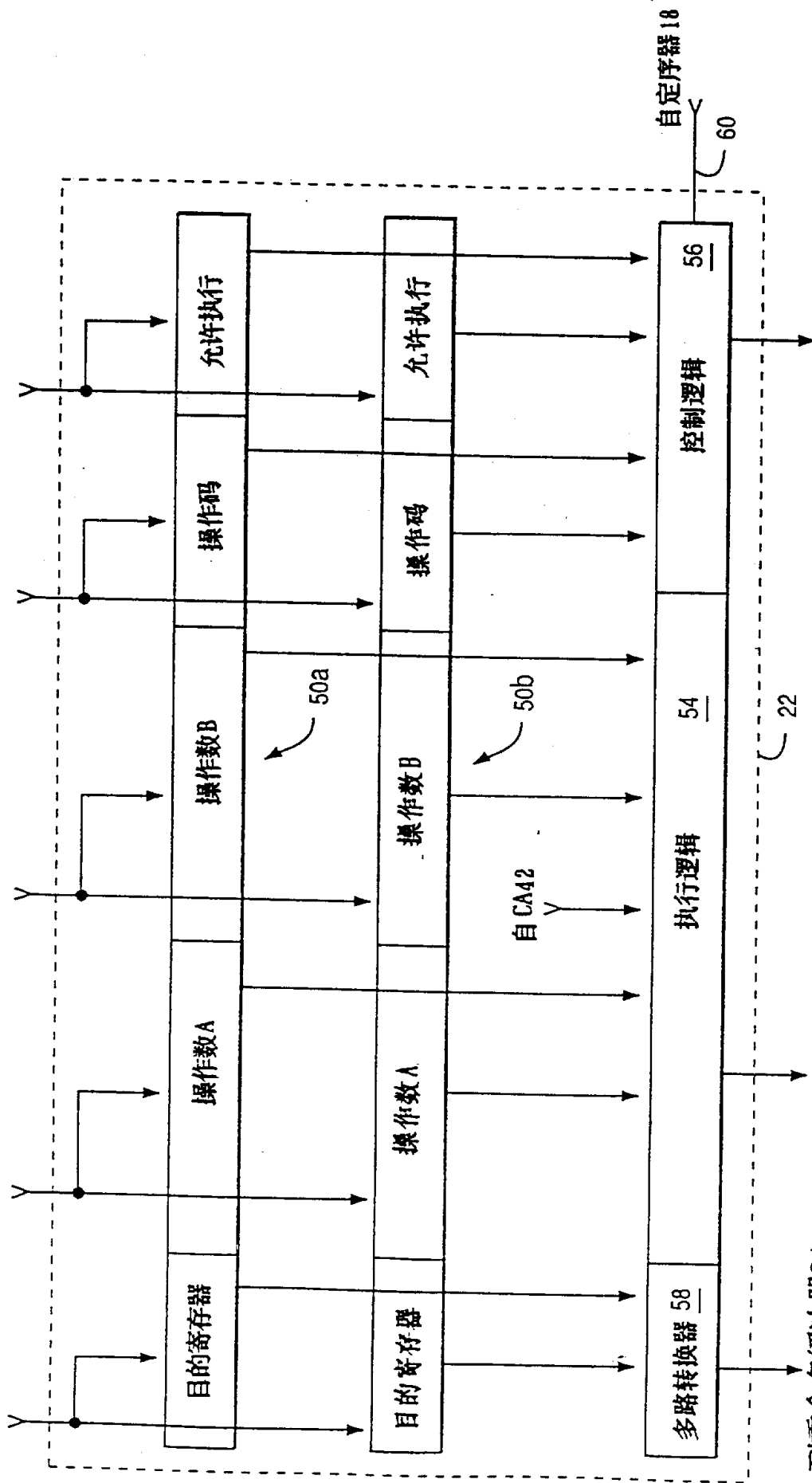


图 1

自GPRs32, 重命名缓冲器34, 自GPRs32, 重命名缓冲器34,
 FXUB24, CFXU26, LSU28 FXUB24, CFXU26, LSU28

自定序器18 自定序器18



到重命名缓冲器34, FXUB 24, CFXU 26, LSU 28

到定序器18

图 2

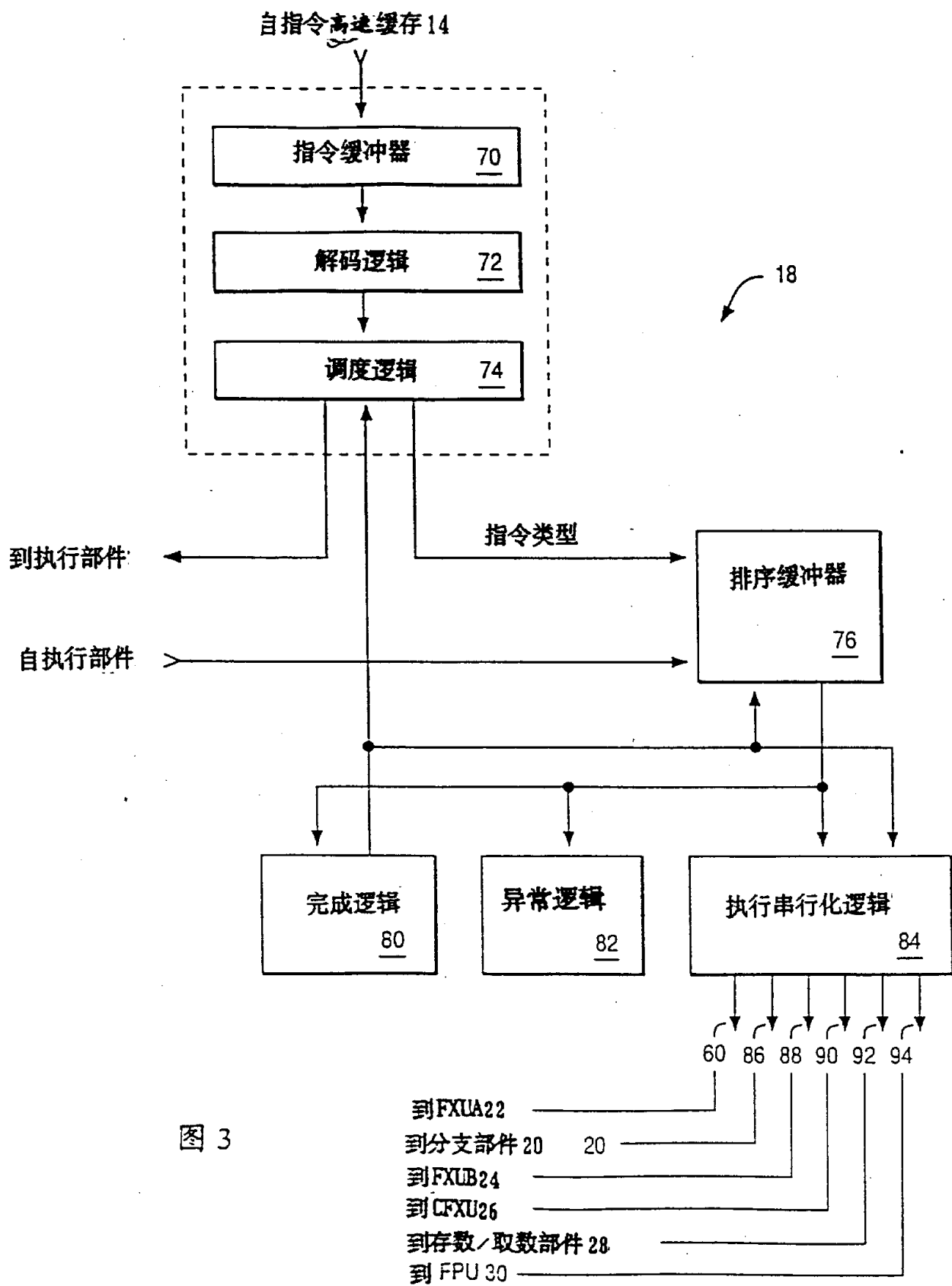


图 3

76

缓冲器号	指令类型		目的GPR数	目的FPR数	完成	异常
	执行部件	允许执行				
0	22	0	1	0	0	0
1	22	0	1	0	0	0
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						

图 4

76

缓冲器号	指令类型		目的GPR数	目的FPR数	完成	异常
	执行部件	允许执行				
0						
1						
2						
3						
4						
5						
6						
7	24	1	1	0	1	0
8	22	1	1	0	1	0
9	28	1	2	0	1	0
10	28	1	2	0	1	0
11						
12						
13						
14						
15						

图 6

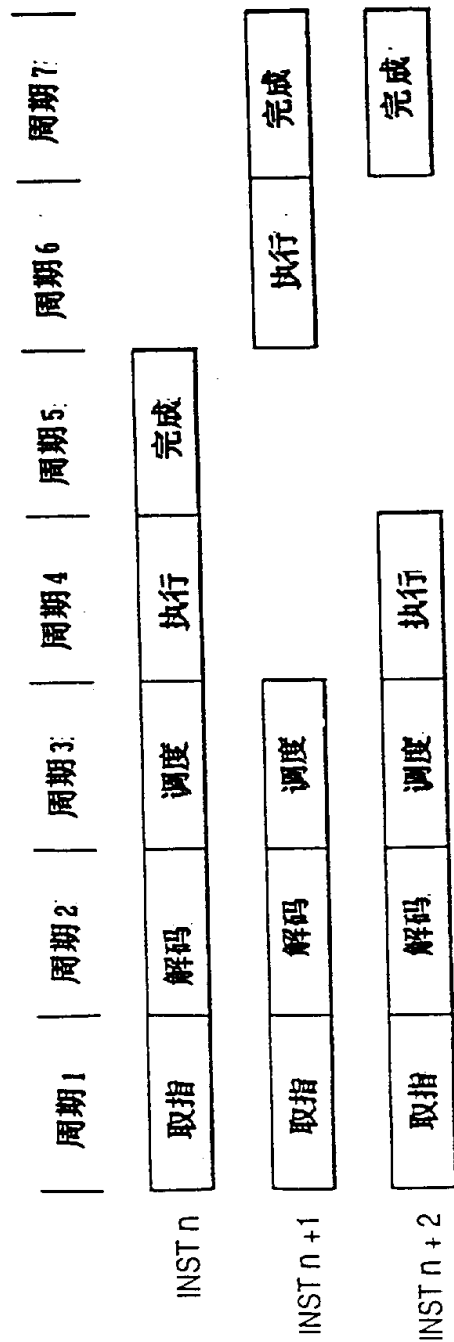


图 5a

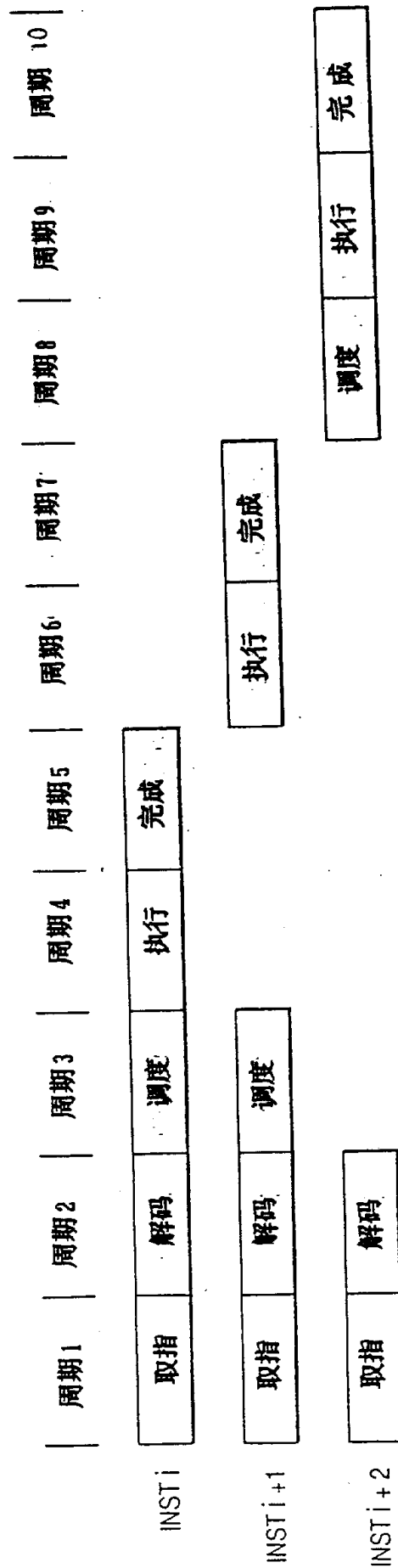


图 5b

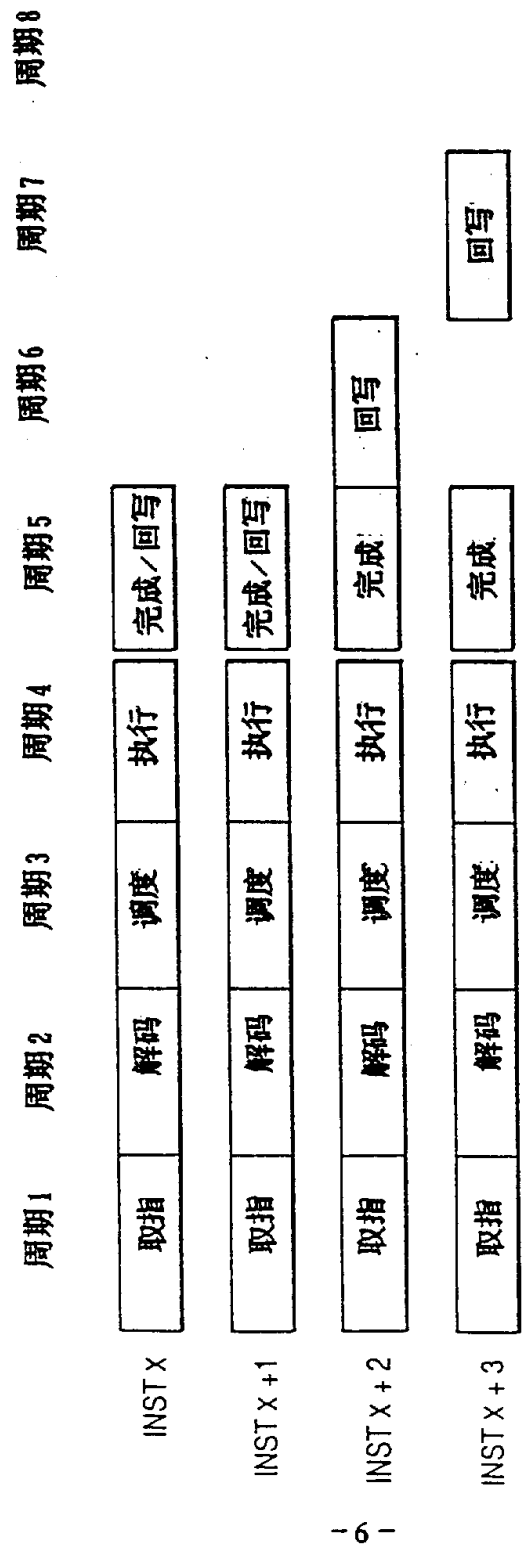


图 7

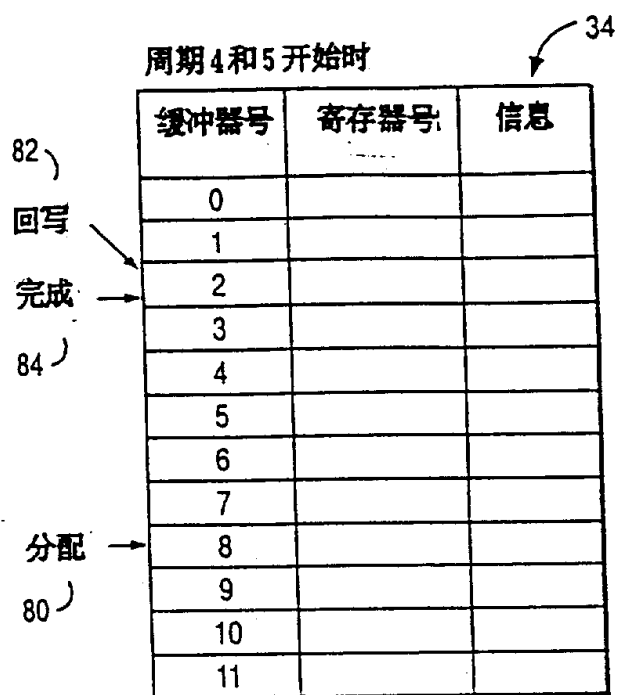


图 8a

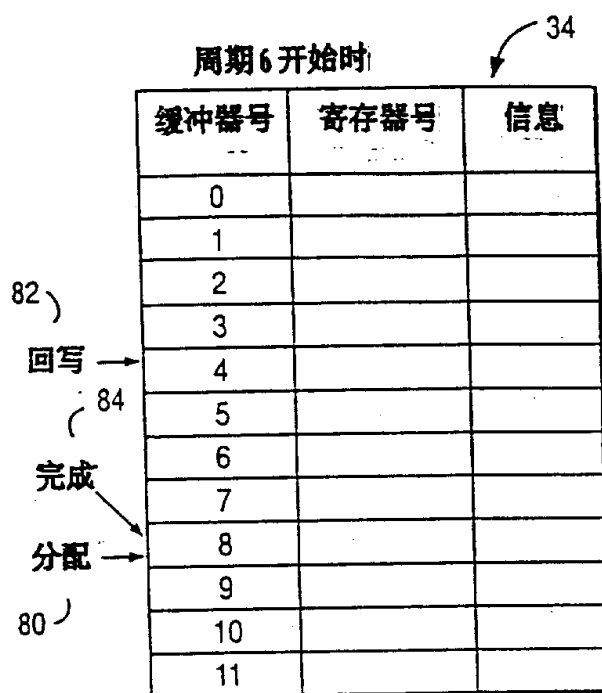


图 8b

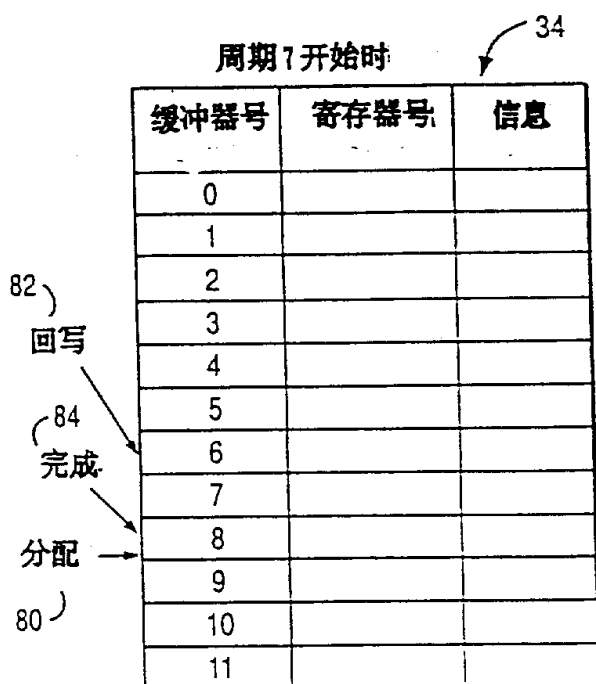


图 8c

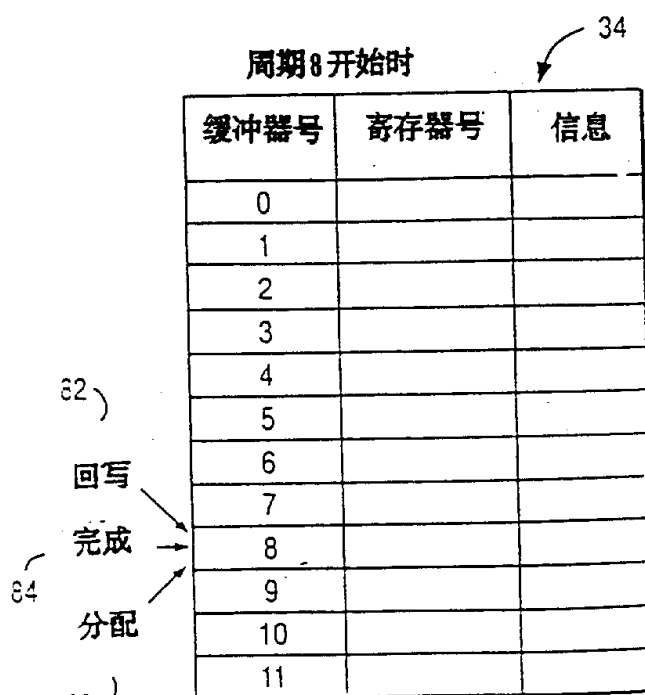


图 8d

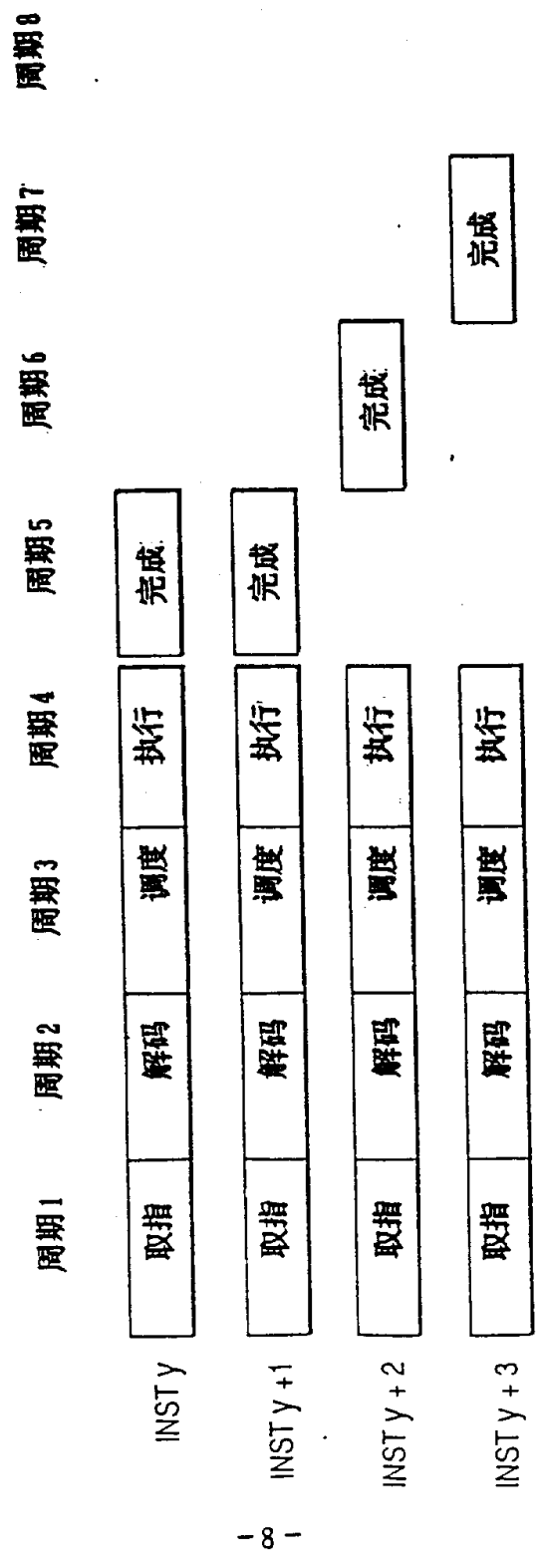


图 9

周期4和5开始时

缓冲器号	寄存器号	信息
0		
1		
完成 → 2		
94) 3		
4		
5		
6		
分配 → 7		
90) 8		
9		
10		
11		

图10a

周期6开始时

缓冲器号	寄存器号	信息
0		
1		
2		
94) 完成 → 3		
4		
5		
6		
7		
分配 → 8		
90) 9		
10		
11		

图10b

周期7开始时

缓冲器号	寄存器号	信息
0		
1		
2		
3		
4		
94) 完成 → 5		
6		
分配 → 7		
90) 8		
9		
10		
11		

图10c

周期8开始时

缓冲器号	寄存器号	信息
0		
1		
2		
3		
4		
5		
6		
7		
完成 → 8		
94) 分配 → 9		
10		
11		

图10d

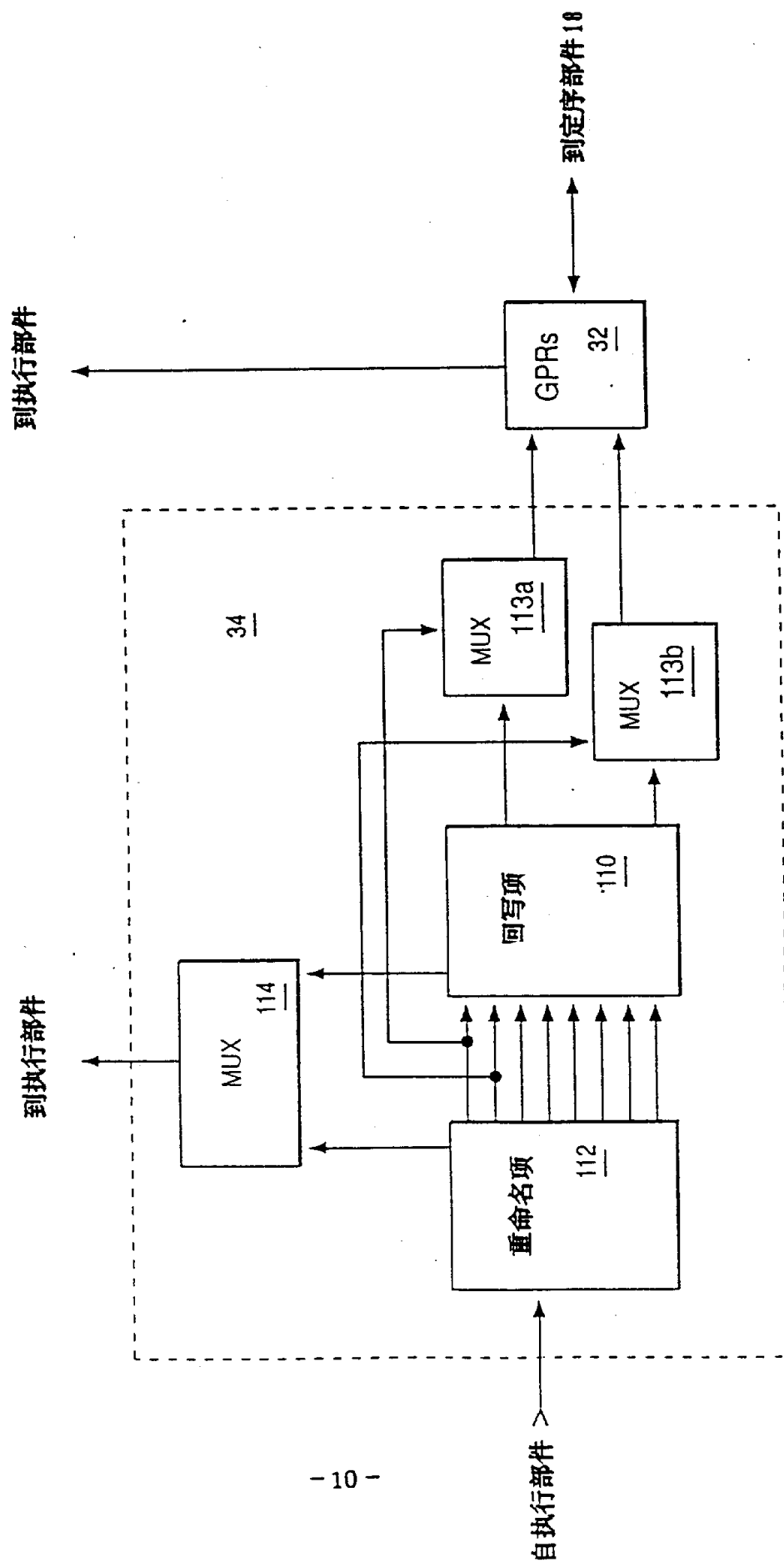


图 11

76

缓冲器号	指令类型		目的 GPR 数	目的 FPR 数	完成	异常
	执行单元	EOK				
0						
1						
2						
3	22	1	2	0	0	0
4	24	1	1	0	1	0
5	28	0	0	0	0	0
6	26	1	1	0	1	0
7						
8						
9						
10						
11						
12						
13						
14						
15						

图 12a

76

缓冲器号	指令类型		目的 GPR 数	目的 FPR 数	完成	异常
	执行单元	EOK				
0						
1						
2						
3	22	1	2	0	1	0
4	24	1	1	0	1	0
5	28	0	0	0	1	0
6	26	1	1	0	1	0
7	22	0	1	0	0	0
8						
9						
10						
11						
12						
13						
14						
15						

图12b

76

缓冲器号	指令类型		目的 GPR 数	目的 FPR 数	完成	异常
	执行单元	EOK				
0	22	1	1	0	1	0
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						

图12c

周期3 周期4 周期5 周期6 周期7 周期8 周期9 周期10 周期11 周期12

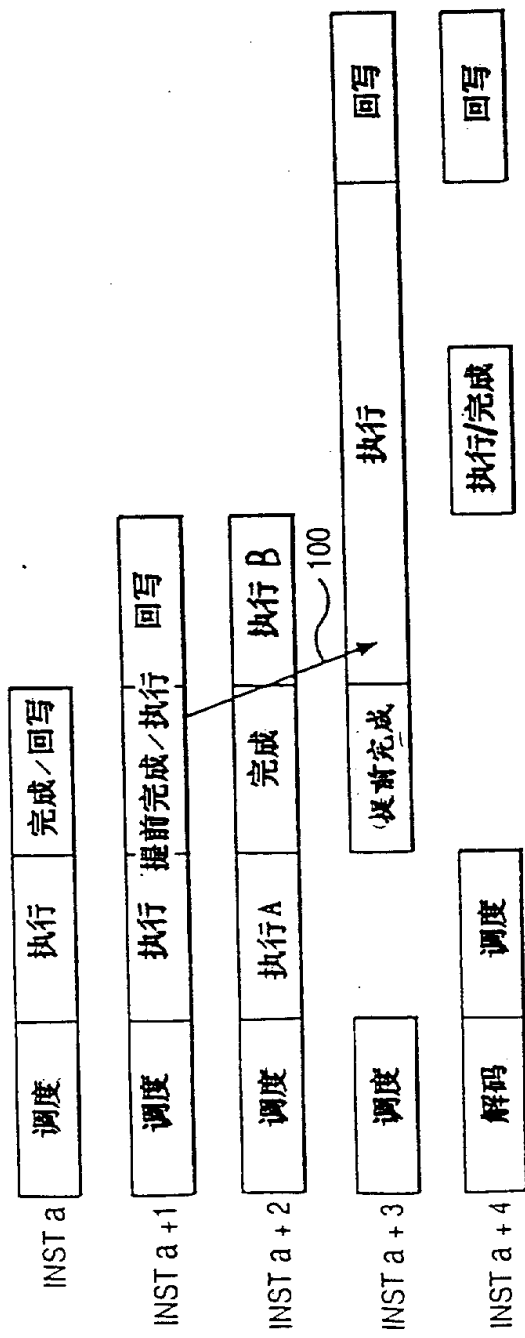


图 13

周期4开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
CO, WB	2		0		0
82	3		0		0
	4		0		1
	5		0		0
AL	6				
80	7				
	8				
	9				
	10				
	11				

图14a

周期5开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
CO, WB	2	DATA	1		0
82	3	DATA	1		0
	4		0		1
	5		0		0
	6		0		1
AL	7				
80	8				
	9				
	10				
	11				

图14b

周期6开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2					
3					
82 WB →	4	DATA	1	CA	1
5			0		0
84 CO →	6		0		1
AL →	7				
80	8				
9					
10					
11					

图14c

周期7开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2					
3					
4					
82 WB →	5		0		0
84 CO →	6		0		1
AL →	7				
80	8				
9					
10					
11					

图14d

周期8开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2					
3					
4					
82 WB →	5		0		0
84 CO, AL →	6	DATA	1	CA	1
80	7				
	8				
	9				
	10				
	11				

图14e

周期9开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2					
3					
4					
82 WB →	5	DATA	1		0
80 AL, CO →	6	DATA	1	CA	1
84	7				
	8				
	9				
	10				
	11				

图14f

周期10开始时

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					

34

FIG. 14g

80 (84

AL, CO, WB → 82

周期 12

周期 11

周期 10

周期 9

周期 8

周期 7

周期 6

周期 5

周期 4

周期 3

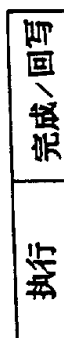
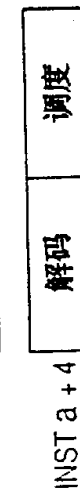
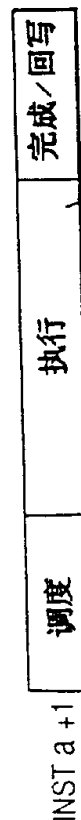
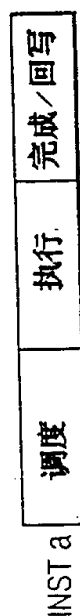


图 15

周期4开始时

34

(84
CO, WB →
82)

AL →
80)

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2			0		0
3			0		0
4			0		1
5			0		0
6					
7					
8					
9					
10					
11					

图 16a

周期5开始时

34

(84
CO, WB →
82)

AL →
80)

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2		DATA	1		0
3		DATA	1		0
4			0		1
5			0		0
6			0		1
7					
8					
9					
10					
11					

图 16b

周期6开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2					
3					
84 — CO, WB →	4	DATA	1	CA	1
	5		0		0
	6		0		1
AL →	7				
80)	8				
	9				
	10				
	11				

图16c

周期7和8开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2					
3					
84 — CO, WB →	4				
	5		0		0
	6		0		1
AL →	7				
80)	8				
	9				
	10				
	11				

图16d

周期9开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2					
3					
4					
84 CO, WB →	5	DATA	1		0
82	6		0		1
AL →	7				
80	8				
	9				
	10				
	11				

图16e

周期10开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2					
3					
4					
5					
84 CO, WB →	6		0		1
82	7				
AL →	8				
80	9				
	10				
	11				

图16f

周期11开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2					
3					
4					
5					
84 CO, WB →	82 AL →	DATA	1	CA	1
80					

图16g

周期12开始时

34

缓冲器号	寄存器号	信息	信息有效	进位	进位有效
0					
1					
2					
3					
4					
5					
80					
84 AL CO WB →	82				

图16h