Software Engineering and Applications 软件工程与应用, 2016, 5(4), 243-253
Published Online August 2016 in Hans. http://www.hanspub.org/journal/sea
http://dx.doi.org/10.12677/sea.2016.54028

Hans汉斯

The Research and Implementation of Pipeline Data Dependency in RISC Microprocessor Design

Yanfei Guo[1], Lihua Song[2], Yin Zhang[2]

[1]ChemChina Petrochemical Co., Ltd., Beijing
[2]School of Computer, North China University of Technology, Beijing
Email: guoyanfei@petro.chemchina.com, slh2g@126.com

Received: Aug. 8th, 2016; accepted: Aug. 27th, 2016; published: Aug. 30th, 2016

Abstract

Data dependency in the processor pipeline usually affects the correctness of the results of the instruction execution, and limits the performance of the processor. In this paper, we first analyze the nature and causes of pipeline data dependency, and put forward the basic principle of data dependency. Then combining this principle with the advantages that both of the suspended pipeline method and the data push forward method, we propose a new solution "a minimum instruction cycle writing back method". The method is flexible and fully saves the consumption of hardware resources. Without loss of generality, the method also can be used for solving the pipeline data dependency problem of the embedded RISC microprocessor.

Keywords

Data Hazard, RISC, CPU

RISC微处理器设计中流水线数据相关问题研究与实现

郭艳飞[1]，宋丽华[2]，张　胤[2]

[1]中国化工油气股份有限公司，北京

文章引用: 郭艳飞, 宋丽华, 张胤. RISC 微处理器设计中流水线数据相关问题研究与实现[J]. 软件工程与应用, 2016, 5(4): 243-253. http://dx.doi.org/10.12677/sea.2016.54028

[2]北方工业大学计算机学院，北京
Email: guoyanfei@petro.chemchina.com, slh2g@126.com

收稿日期：2016年8月8日；录用日期：2016年8月27日；发布日期：2016年8月30日

摘　要

处理器流水线中的数据相关问题，通常会影响指令执行结果的正确性，并限制处理器性能的发挥。本文首先对流水线数据相关问题的本质及成因进行剖析，并提出解决数据相关的基本原理。依据此原理，并结合两种常用解决方法，流水线暂停法和数据前推法的优点，提出新的解决方案——“最小指令周期写回法”，该方法既灵活又充分节省了硬件资源的消耗，不失一般性，该方法可为解决用于嵌入式设备的RISC微处理器流水线中的数据相关提供参考。

关键词

数据相关，RISC，CPU

1. 引言

流水线技术因其能够以相对较小的硬件资源消耗换取较高的运行效率已经被广泛应用于多种类型的微处理器中。然而，在流水线技术为微处理器降低成本并提升性能的同时，采用此项技术同样会引发一些不容忽视的问题。数据相关(Data Hazard)问题就是采用流水线技术所造成的主要问题之一，特别是随着流水线深度的增加以及流水线各级内部逻辑复杂度的增加，数据相关问题也将显得更加突出和严重，能否以及如何解决数据相关将直接决定微处理器是否能产生正确的运算结果，并在很大程度上影响其运算性能。

高性能微处理器大都采用 Tomasulo 动态指令调度算法[1]来解决数据相关问题，而对于专为嵌入式系统而设计的 RISC 微处理器来说，采用 Tomasulo 算法来解决数据相关的现实意义并不大[2]。一方面，Tomasulo 算法并不完全适用于解决单指令流处理器的数据相关问题，另一方面，嵌入式系统及专用设备的微处理器往往在硬件资源的消耗方面有相对较高的要求，因此人们也很难为其提供大量的额外片上资源用于实现一个复杂的指令调度算法。通常，人们会采用流水线暂停法或数据前推法来解决此类微处理器的数据相关问题，但这两种解决方案其各自的缺点同样非常明显，并会从不同的方面影响处理器的性能，使其无法满足人们的实际需求。

为了能更好地解决 RISC 微处理器流水线数据相关问题，本文将对数据相关的形成原因及其本质进行讨论，并比较两种常用解决方案，即流水线暂停法及数据前推法的优劣。通过对此两种方案的分析与综合，我们将提出一种新的用于解决 RISC 微处理器流水线数据相关问题的技术——最小指令周期写回法，并以一款 32 位 RISC 微处理器 Rewrite1060525 的体系结构为例，讨论此种解决方案的特点及其实现。

2. 数据相关

在对 RISC 微处理器流水线数据相关问题进行讨论之前，我们以常见的 RISC 微处理器体系结构为例，对存在的问题进行简要分析。

图 1 是一个典型的 RISC 微处理器的概要模型，其采用了五级流水线结构，分别为 IF 取指令级、ID 译码级、EX 执行级、MEM 存储器访问级和 WB 写回级，指令与数据分开存储[3]。所有的指令都将从 IF

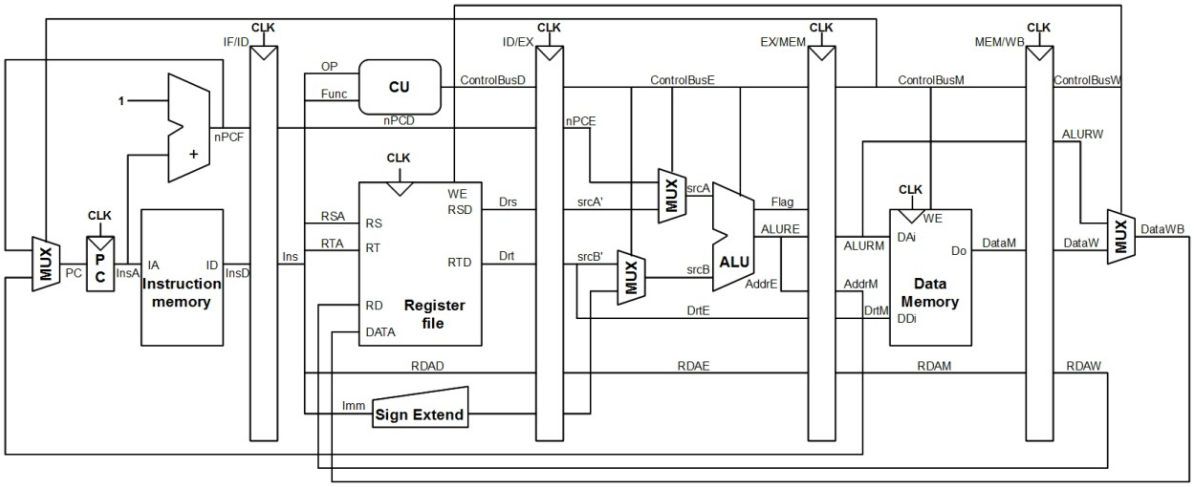

Figure 1. RISC processor with pipeline structure
图 1. 典型的 RISC 微处理器体系结构

级的指令存储器中取出并依次进入整条流水线的各级。值得注意的是，在该流水线的写回级与译码级之间存在一条反馈路径，也就是说，图中所示的通用寄存器堆既要在译码级被读取，又要在写回级被写入。实际上，正是由于这条反馈路径的存在，使得整条流水线不再保持绝对意义上的线性特征[4]，从而造成了数据相关问题。

从指令执行的角度看，假设有连续两条指令在上述 RISC 处理器中执行：

第 k 条指令完成将寄存器 R5 的值与 R2 的值相加，结果写入 R3 的操作，其后一条指令则完成寄存器 R1 与 R3 (R5 与 R2 之和)相减，结果写入 R9 的操作。如图 2 所示，根据其流水线结构可得知，第 k 条指令将在第五个时钟周期将指令执行的结果写入 R3，而第 k + 1 条指令却在第 3 个时钟周期就已经进入流水线的译码级，并需要从寄存器堆中取出 R3 的值。显然，在不做任何处理的情况下，第 k + 1 条指令无法从 R3 取出正确的操作数，从而使整个程序无法得出正确的执行结果。此时所出现的问题就是“先写后读”(RAW)相关。

当然，实际应用中所遇到的数据相关问题除先写后读相关外，还有“先读后写”(WAR)相关和“先写后写”(WAW)相关。而对于多数 RISC 处理器来说，因其流水线结构相对简单，且流水线中各级的操作均在一个时钟周期内完成，也就避免了指令在流水线中的乱序流动，从而不会出现 WAR 和 WAW 相关。本文的后续内容也将重点对 RISC 处理器中的 RAW 相关问题展开讨论。事实上，WAR 及 WAW 相关的解决方案均可由 RAW 相关的解决方案衍生而出。

3. RISC 处理器中数据相关问题的常用解决方案

目前最常见的用于解决数据相关问题，特别是 RISC 微处理器中 RAW 相关的方法有两种，其中一种是“流水线暂停”法，另一种是“数据前推” [5]。两种方法各有各的优点，但其各自的缺点同样非常明显，特别是当这两种方案用于解决嵌入式设备处理器的数据相关问题时，这些缺点将显得更为棘手。

3.1. 流水线暂停法

流水线暂停法是避免数据相关的最简单，也是最常用方法，其核心思想可以简述为，若流水线中的某一条指令与当前正要被执行的指令存在数据相关时，延迟当前指令的执行，直到相关被解除之后，该指令方可继续在流水线中执行。采用硬件方式来实现时则需要向处理器中添加专用的冲突检测，及流水线暂停部件。流水线暂停的过程如图 3 所示。

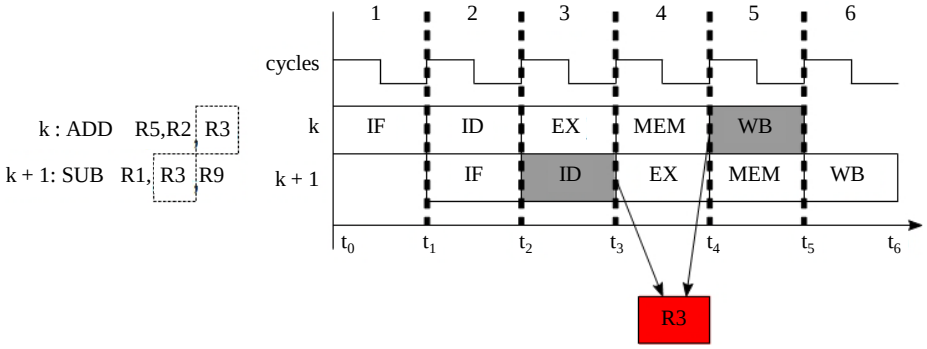

Figure 2. Two instructions with data hazard
图 2. 两条数据相关的指令

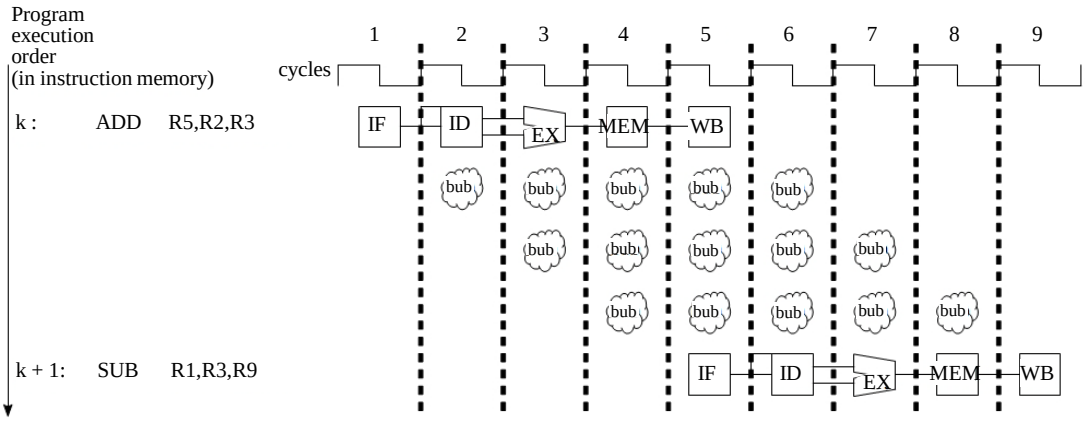

Figure 3. Abstract pipeline diagram to illustrating stall to solve hazards
图 3. 流水线暂停法

流水线暂停法的优点是实现简单，特别是当采用软件的方法来实现时，完全不需要硬件上做额外开销，同时，使用流水线暂停法来解决数据相关也并不会使流水线的控制别的更加复杂。这种方法的缺点则是流水线的吞吐率和效率低，毕竟，大量的处理器时钟周期都被 NOP 指令所占用。

3.2. 数据前推法

数据前推法的本质是，将流水线中已得出但尚未被写回的运算结果通过所建立的专用数据路径，直接反馈到所需要这个运算结果的功能段中。其基本原理是数据重定向，适用于解决采用顺序流动方式的处理机中所出现的 RAW 相关问题。图 4 表示了数据前推法解决数据相关的基本过程。

数据前推法必须通过硬件的方式来实现，除去专用数据路径外，其核心部件是冲突检测单元，通常是将流水线 EX 级中两个操作数 SrcA、SrcB 的源寄存器编号与 MEM 级和 WB 级中目的寄存器的编号进行比较，如果二者相等，并且寄存器堆写使能信号有效，则说明发生了 RAW 相关，需要进行数据前推。其算法描述如下所示：

if ((EX.rs! = 0) AND (EX.rs == MEM. rd) AND MEM.registerWE) *then*
SrcA = MEM.alur;
else if ((EX.rs! = 0) AND (EX.rs == WB.rd) AND WB.registerWE) *then*
SrcA = WB.alur;
else SrcA = E.SrcA;

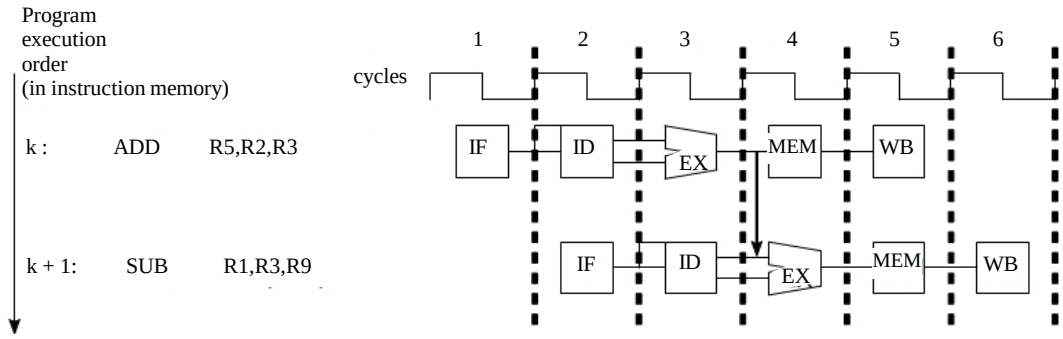

Figure 4. Abstract pipeline diagram to illustrating forwarding
图 4. 数据前推法

上述算法描述了采用数据前推法解决 RAW 相关时，ALU 部件 A 操作数的前推逻辑，B 操作数与其类似。

数据前推法的优点是解决数据相关问题，特别是对于 ALU 指令产生的 RAW 相关问题时不会影响整条流水线的效率。但实现数据前推法需要较大的硬件资源开销，另外还要注意的是，仅使用数据前推法是无法解决 RISC 处理器中某些数据相关问题的，例如由 Load 指令及其后续指令所产生的 RAW 相关等，若出现此类相关则必须暂停整条流水线，因此，依靠数据前推法来解决 RISC 处理器中的数据相关问题会使得对流水线的控制变得相对复杂。

3.3. 两种常用方法的总结

从上文的描述和对比中不难看出，实现流水线暂停法来解决数据相关，实际上是以处理器时钟周期的消耗来抵消硬件资源开销的过程，是一种以时间换空间的极端策略。相反，实现数据前推法来解决数据相关则是以较大的硬件资源开销来消除处理器时钟周期损失的过程，也就是以空间换时间的极端方案。尽管这两种方法都能有效地解决数据相关冲突，但对于那些专为嵌入式设备设计的 RISC 微处理器来说，因其本身受到较多的时空因素的制约，所以这两种方案的实现往往会在相当大的程度上受到限制。为此，人们有必要找到一种折衷方案，来解决那些专用设备上的 RICS 微处理器中的数据相关问题。

4. 造成流水线数据相关问题的本质原因

要找到一种能更好地解决 RISC 处理器中数据相关问题的方法，必须首先了解数据相关问题的本质及其成因。解决此问题可以从两个角度入手，一是从指令的角度，二是从处理机执行的角度。

从指令的角度来看，发生数据相关的两条指令，它们的源操作数和目的操作数之间一定存在某种关联[4]，图 5 抽象地描述了这种关联。

如图 5 所示，S 和 D 分别表示指令的源操作数和目的操作数。假设按原定指令执行次序，指令 i 应限于指令 j 执行，那么采用顺序流动方式的处理机则会出现图 5 所示的“先写后读”相关。而出于某种原因，流水线中的指令出现乱序流动，j 指令的执行超越了 i 指令，此时除“先写后读”相关外，还可能出现图 5 所示的“先读后写”相关和图 5 所示的“写—写”相关。发生此三种相关时，流水线中的指令 i 和指令 j 之间的关系可用下列关系式表达：

$$\begin{aligned} &\text{RAW hazard}: D(i) \cap S(j) \neq \varnothing \\ &\text{WAR hazard}: S(i) \cap D(j) \neq \varnothing \\ &\text{WAW hazard}: D(i) \cap D(j) \neq \varnothing \end{aligned} \quad \text{(公式 1)}$$

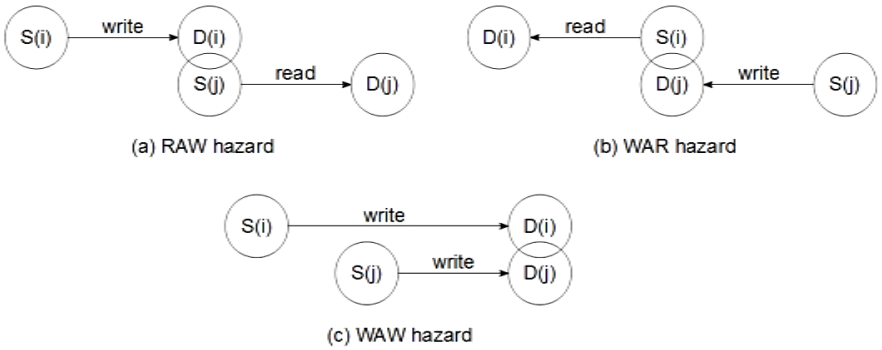

Figure 5. The relationship between instructions that might lead to data hazards
图 5. 存在数据相关问题的指令

图 5 描述了发生数据相关的必要条件，也就述说，当流水线中的两条指令 i 和 j 发生数据相关冲突时，这两条指令一定满足(公式 1)三个关系式之一。但满足(公式 1)所示关系的指令并不一定会发生冲突，因为数据相关问题的出现还与处理机流水线的结构，以及指令在流水线中实际的执行情况密切相关。

从处理机实际执行指令的角度看，由于受到流水线结构的制约，通常情况下，一条指令运算所得出的结果往往不能立即被写回到这条指令的目标存储单元中，而是要随着时钟节拍流动一段时间，直到被流水线中的写回部件完成写回操作，此时这条指令完整的指令周期正式结束。然而，由于流水线中的指令并行执行的特点，后续指令的读取操作数的操作往往又是先于前序指令的写回操作，特别是在指令之间的启动距离相对较短的情况下，这种读操作与写操作之间的时间差就很有可能导致数据相关问题的出现。

假设 i 和 j 是采用顺序流动方式的流水线中的两条指令，i 于 j 前执行，T_{read} 和 T_{write} 分别表示指令读取操作数以及写回运算结果的时刻，结合可得出以下结论：

$$\text{当：}\ \mathrm{D(i)}\cap\mathrm{S(j)}=\mathrm{P},\mathrm{P}\neq\varnothing$$

$$\text{并且：}\ T_{\text{iwrite}}(\mathrm{P})=T_{\text{jread}}(\mathrm{P})+\Delta t,\Delta t>0 \tag{4.2}$$

时，流水线中将发生 RAW 相关。同理可得 WAR 相关发生的条件为：

$$\mathrm{S(i)}\cap\mathrm{D(j)}=\mathrm{Q},\mathrm{Q}\neq\varnothing\ \text{且}\ T_{\text{iread}}(\mathrm{Q})=T_{\text{jwrite}}(\mathrm{Q})+\Delta t,\Delta t>0 \tag{4.3}$$

WAW 相关发生的条件为：

$$\mathrm{D(i)}\cap\mathrm{D(j)}=\mathrm{R},\mathrm{R}\neq\varnothing\ \text{且}\ T_{\text{iwrite}}(\mathrm{R})=T_{\text{jwrite}}(\mathrm{R})+\Delta t,\Delta t>0 \tag{4.4}$$

以上 3 点结论可以看作是出现数据相关问题的根本原因，避免发生数据相关也应从削弱这三个条件入手。在实际应用中，通常程序员是无法修改已进入指令存储器中指令的顺序的，因此，避免数据相关的关键就在于如何削弱 Δt 所产生的影响。

实际上，无论是流水线暂停法还是数据前推法，其根本目标都是减小，甚至消除指令间访问操作数的时间差 Δt。以最常出现的 RAW 相关为例，数据前推法实际上是通过建立专用数据路径，提前返回运算结果，从而减小 Δt。此过程可作如下描述：

建立专用路径，提前 i 指令运算结果的反馈时间：

$$T_{\text{iwrite}}(\mathrm{P})'=T_{\text{iwrite}}(\mathrm{P})-\Delta t',\Delta t'>0$$

则有：$T_{\text{iwrite}}(\mathrm{P})'+\Delta t'=T_{\text{jread}}(\mathrm{P})+\Delta t$，即

$$T_{\text{iwrite}}(\text{P})' = T_{\text{jread}}(\text{P}) + (\Delta t - \Delta t')$$

如果采用流水线暂停法，实际上是通过推迟后续指令访问操作数的时间，从而减小 Δt。过程描述为：推迟 j 指令访问操作数的时刻：

$$T_{\text{jread}}(\text{P})' = T_{\text{jread}}(\text{P}) + \Delta t'', \Delta t'' > 0$$

则有：$T_{\text{iwrite}}(\text{P}) = \left(T_{\text{jread}}(\text{P})' - \Delta t''\right) + \Delta t$，即

$$T_{\text{iwrite}}(\text{P}) = T_{\text{jread}}(\text{P})' + (\Delta t - \Delta t'')$$

上面两个过程抽象地描述了流水线暂停法和数据前推法解决数据相关问题的实质，从中可知，当 Δt 足够小，以至于 $T_{\text{iwrite}}(\text{P}) \le T_{\text{jread}}(\text{P})$ 时，即使指令间存在 $\text{D(i)} \cap \text{S(j)} \neq \varnothing$ 这样的关系，也不会发生 RAW 相关。我们还应注意到，无论是流水线暂停法还是数据前推法，都是“单方面”地进行延迟后续指令或前推前序指令的操作，而正是这种单方面的策略使得两种方法走向了时空转换的两个极端，并制约了这两种方法在实际应用中的具体实现。

5. 最小指令周期写回法

上文中已经对数据相关问题的本质、成因，及解决问题的关键进行了分析和总结，同时也比较了两种常用的解决方案，流水线暂停法和数据前推法的特点，并对这两种方法出现较大的时间、空间代价的原因作了简要分析。综合已有结论，并结合两种常用方案的优点，我们提出一种用于避免数据相关问题的折衷方案——最小指令周期写回法，以为解决嵌入式设备中 RISC 微处理器的数据相关冲突提供新的思路。

这种方法具有实现方式灵活简单，硬件资源消耗小等特点，并非常适用于解决 RISC 处理器流水线中的 RAW 相关问题。

5.1. 最小指令周期写回法的基本原理及实现原则

与流水线暂停法和数据前推法相同，最小指令周期写回法解决数据相关问题的入手点也是在于减小前后两条指令之间访问操作数时所存在的时间差 Δt。仍以 RAW 相关为例，其产生 Δt 的原因是指令的运算结果产生后不能被立即被下一条指令所使用，而是要按时钟节拍在流水线中继续流动，直到被写回部件写回到目标寄存器中为止。运算结果在流水线中流动的时间越长，可能造成的数据相关也就越严重。实际上，对于运算指令来说，其正确的运算结果和连带的状态产生后，这条指令主要任务基本上已经完成，而在被写回部件写回之前，这些结果和状态在流水线中的流动并不会引起处理机状态的变化。最小指令周期写回法就是依据类似的思想，以“结果得出即写回”为原则，尽可能缩短每条需要进行结果反馈的指令的指令周期，避免运算结果在流水线中不必要的流动，从而实现 $T_{\text{iwrite}}(\text{P})' = T_{\text{iwrite}}(\text{P}) - \Delta t'$ 这样的效果。但仅仅将指令周期控制到最短也只能削弱数据相关问题的严重性，减小其出现概率，要完全消除数据相关冲突，必须在提前结束前序指令的指令周期同时，延迟后续指令访问操作数的时间，也就是要实现 $T_{\text{jread}}(\text{P})' = T_{\text{jread}}(\text{P}) + \Delta t''$ 的效果，此时的 $\Delta t''$ 将会明显小于仅采用流水线暂停法时因暂停流水线而消耗掉的处理器时钟周期数。这就是最小指令周期写回法的基本原理，从中我们不难发现，此种方法结合了流水线暂停法和数据前推法两者的特点。

设计和实现最小指令周期写回法，并使其发挥真正的功效，必须遵循以下 4 点原则：

1) 通过在产生反馈结果的部件(如 ALU、Data Memory 等)与通用寄存器堆之间建立专用数据路径，

实现结果得出即写回的效果。

2) 对于有必要做延迟处理的指令，通过软件的方式推迟其访问源操作数的时间。

3) 对流水线中用于写回运算结果的路径进行集中控制。

4) 其余由于流水线结构造成的 WAR、WAW 相关，以及因采用本方法而造成的 WAW 相关，采用软件方式实现的流水线暂停法进行处理。

在实际操作中应注意，由于进行结果反馈的指令其最小指令周期可能不同，如 Load 指令和 ALU 指令，因此可能造成 WAW 相关。但对于 RISC 处理器来说，绝大多数非控制类指令都为 ALU 指令，Load 指令出现的频率相当小，所以即使采用以软件方式实现流水线暂停法来解决这种 WAW 相关，处理器在性能上的损失也非常小。同时，通常 RICS 微处理器流水线中的指令在访问源操作数之前不会出现乱序流动的情况，因而也就不会出现 WAR 相关，即便某些 RISC 处理器中出现 WAR 相关，也可以依据上述 4 点原则，同 RAW，WAW 相关进行统一处理。

5.2. 最小指令周期写回法的硬件设计

依据最小指令周期写回法的基本原理和实现原则可知，在硬件层面上，实现此种方法包含两大要点，一是建立专用数据路径，二是对这些路径进行集中控制，因此必须根据指令实际的执行情况设计一个专用的数据路径控制器，其结构如图 6 所示。

数据路径控制器将对所有运算指令及存储器访问指令的执行结果的写回进行集中控制，除接收指令运算结果及其自身所需要的控制信号外，还需接收流水线执行级中正在进行运算的指令的标识。控制器内部包含数据通路控制逻辑以及一条由若干个寄存器组成的寄存器队列。此寄存器队列用于接收指令标识，并在统一的处理器时钟周期的驱动下形成一条流水线，其中的每一个寄存器可以称为指令标识寄存器。进入这个队列的指令标识信号的流动将与其对应指令的执行时序同步，也就是说，当某一条指令的指令标识进入队列中相应指令标识寄存器的同时，这条指令的执行结果已生成并稳定，此时，数据通路控制逻辑开放该指令执行结果的写回路径，完成写回操作。

图 7 简要描述了写回 Load 指令执行结果的时序，数据路径控制器在第二个时钟周期接收到 Load 指令标识信号“Loadmarker”，随即，该信号按时钟节拍在寄存器队列中流动，直到第四个时钟周期，“Loadmarker”信号进入指令标志寄存器 LoadmarkReg 中。而在第四个时钟周期，数据存储器中的数据刚好被取出，此时数据通路控制逻辑发送“writebackLR”信号，开放数据存储器到通用寄存器堆之间的专用路径，完成执行结果的写回。

整个控制过程可作如下描述：

if (insmarkReg == insmark) *then*

writeback insResult;

else block data path;

当流水线的执行级中包含多个运算时，对每个运算部件产生结果的写回路径做相同控制，即可实现“结果得出即写回”的效果。同时，配合流水线暂停法对后续指令访问源操作数的延迟，则可以避免数据相关冲突。

5.3. 最小指令周期写回法的实际应用

为了验证最小指令周期写回法的可行性及其在实际应用中的效果，我们使用该方法来解决处理器 Rewrite1060525 流水线中的数据相关。Rewrite1060525 中流水线的结构如图 8 中所示。

Rewrite1060525 是一款用于 SOPC 平台教学实验的 32 位 RISC 微处理器，采用指令与数据分开存储

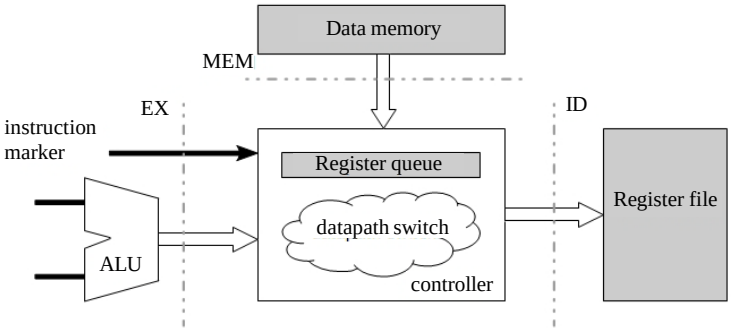

Figure 6. Structure of the data path controller
图 6. 数据路径控制器

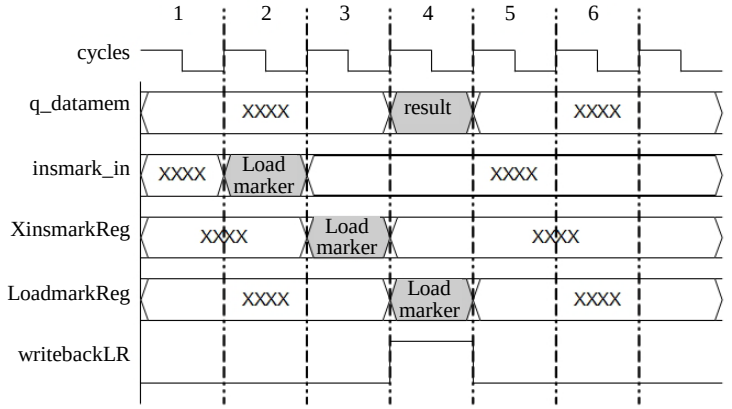

Figure 7. Write back the result of load instruction
图 7. Load 指令执行时序

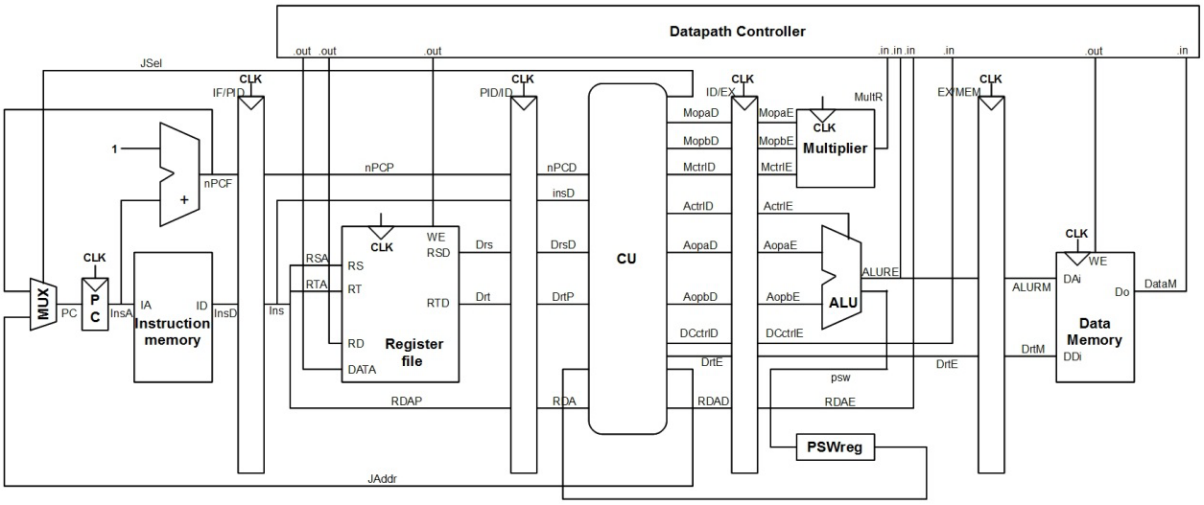

Figure 8. Structure of Rewrite1060525
图 8. Rewrite1060525 体系结构

的哈佛结构，具有 5 级流水线结构。这五级依次为 IF 取指令级，用于将指令从指令存储器中取出；PID 指令预处理级，用于读取指令所需源操作数，并分析出运算指令的目标存储单元的地址信息；ID 译码级，对指令进行译码，同时处理所有控制类指令；EX 执行级，执行指令所需完成的运算操作，运算部件除算

Table 1. Comparison of the number of stall cycles under worst case
表 1. 最坏情况下流水线暂停占用的时钟周期数比较

	使用最小指令周期写回法	不使用最小指令周期写回法
ALU 指令数据相关引起的流水线暂停时钟周期	2	4
乘法指令数据相关引起的流水线暂停时钟周期	6	8
Load 指令数据相关引起的流水线暂停时钟周期	4	6

Table 2. Consumption of hardware resources (without on chip RAM)
表 2. 硬件资源消耗比较(无片上 RAM 情况下)

	Total logic elements	Total registers
Rewrite1060525	8091	2877
Datapath controller	226	40

数逻辑运算器外还包含一个带有 4 级流水线结构的 32 位整数乘法器；MEM 存储器访问级，完成对数据存储器的访问。由于使用了最小指令周期写回法，流水线中不再设置写回级，所有执行结果的写回操作由数据路径控制器集中控制。

Rewrite1060525 将采用 Verilog-HDL 进行描述，并由 QuartusII 进行综合。硬件环境选用 Cyclone II：EP2C35F672C8。实验将在采用和不采用最小指令周期写回法的情况下分别进行，对比由 ALU 指令、乘法指令和 Load 指令引起的最坏情况下的数据相关导致流水线暂停所占用的时钟周期数的差异。具体数据如表 1 所示。

从中可以看出，因为使用了最小指令周期写回法，由 ALU 指令引起数据相关时，流水线暂停所占用的时钟周期减少了 50%；乘法指令引起数据相关时，流水线暂停所占用的时钟周期数减少了 25%；Load 指令则减少了 33.3%。表 2 给出了硬件资源消耗的情况。

用于实现最小指令周期写回法的部件数据路径控制器，在逻辑单元的消耗中只占 Rewrite1060525 整体的 2.8%，寄存器资源的消耗只占 1.4%。通过上面数据的对比可以看出，最小指令周期写回法一方面有效地解决数据相关问题，另一方面又充分节省了硬件资源的消耗。

6. 结论

本文分析并总结了 RISC 微处理器流水线中数据相关问题的成因及其基本解决思路，对比了常用的两种解决方法，流水线暂停法和数据前推法的优劣，提出并设计了新的解决方案“最小指令周期写回法”。此方法具有实现灵活，硬件资源消耗小，对处理器性能影响小的特点，适用于解决嵌入式设备 RISC 微处理器流水线中的数据相关问题。

基金项目

北方工业大学大学生科技活动资助项目。

参考文献 (References)

[1] 李山山, 刘敬晗. 利用 Tomasulo 算法处理数据相关的流水线 CPU 设计[J]. 实验室研究与探索, 2014, 33(12): 90-95.

[2] Lu, J.J., Zhou, X.F. and Wang, J.Y. (2007) A Novel Dynamic Scheduling Algorithm of Data Hazard for Embedded Processor. *Proceedings of the 7th IEEE International Conference on ASIC*, Shanghai, 22-25 October 2007, 28-31.

[3] Harris, D.M., Harris, S. L. 数字设计和计算机体系结构(英文版) [M]. 北京: 机械工业出版社, 2008: 401-406.

[4] 郑纬民, 汤志忠. 计算机系统结构(第二版) [M]. 北京: 清华大学出版社, 2014.

[5] 东野长磊. 基于现场可编程门阵列的 RISC 处理器设计[J]. 计算机工程, 2011, 37(11): 241-243.