Tomasulo 算法与记分牌调度算法研究

王 磊

(陕西广播电视大学 计算机与信息管理教学部,陕西 西安 710119)

摘 要: 为消除数据竞争,可通过旁路控制机构或其它技术手段来解决,但仍然不能从根本上消除数据竞争。针对数据相关的不可避免性,采用流水线的动态调度算法来解决数据竞争问题,常用的动态调度算法是 Tomasulo 算法和记分牌调度算法,分析了 Tomasulo 算法和记分牌调度算法的基本思想和算法实现,对他们的异同进行了分析说明,实验证明,动态调度算法具有很好的消除数据竞争效果。

关键词: 数据竞争;动态调度;Tomasulo 算法;记分牌调度算法

中图分类号:TP368.1 文献标识码:A 文章编号:1003-7241(2013)06-0023-04

The Tomasulo Algorithm and Scoreboard Scheduling Algorithm

WANG Lei

(Department of Computer and Information Management, Shaanxi Radio and TV University, xi' an 710119 China)

Abstract: Eliminate data competition may be solved through the bypass control agencies or other technical means, but it still can not fundamentally eliminate data competition. The problem of data race may be solved by the pipeline dynamic scheduling algorithm for the inevitability of related data. The common dynamic scheduling algorithm is the Tomasulo algorithm and the scoreboard scheduling algorithm. The basic ideas and algorithm codes of Tomasulo algorithm and scoreboard scheduling algorithm are analyzed and their similarities and differences are also analyzed. Experiments show that the dynamic scheduling has fine effect for the elimaton of data competition.

Key words: data competition; dynamic Scheduling; tomasulo algorithm; the scoreboard scheduling algorithm

1 引言

为满足处理器性能增长的要求,提高处理器性能,要求处理器具有每个周期能发射执行多条指令的能力,超标量微处理器可以同时发射执行多条指令,但也同时增加了相关性机率。可以通过旁路控制机构,编译技术调度指令序列等手段分开具有相关性的指令,来使竞争数目和性能损失减小到最低程度,但不可避免的数据相关使得数据竞争不可能完全消除。文献[1][2]分析了数据、结构、控制相关性等问题,提出了解决方法,并提到了动态转移预测和重排序缓冲机制[2]。针对超标量体系中遇到的数据相关和结构相关的问题,讨论了流水线的静态调度思想和动态调度思想,常用的数据流调度方法是 Tomasulo 算法,介绍了 Tomasulo 算法和记分牌方法的动态调度的基本思想,对算法进行了比较分析。

2 数据相关和控制相关

程序中的相关(Correlation)是指在相近指令之间存在有某种关系,且会影响到指令流水线的执行,分为数据相关和控制相关。数据相关指导在执行本条指令的过程中,如果用到的指令、操作数、变址偏移量等是前面指令的执行结果,则必须等待前面的指令执行完成,并把结果写到主存或通用寄存器中之后,本条指令才能开始执行。控制相关是指由条件分支指令、转子程序指令、中断等引起的相关。

数据相关分为三类:RAW 相关、WAR 相关、WAW 相关。RAW 相关(真相关)指若两条指令之间存在真相关,则两条指令不能同时或完全重叠执行。WAR 相

收稿日期:2012-11-14

关(反相关)与 WAW 相关(输出相关)两种相关的产生都是由于指令的乱序执行所引起的。在数据相关中,真相关是在一条指令的源操作数依赖于前一条指令的结果的条件下发生的,真相关的解决必须有操作数的流动,出现相关的两条指令不能同时或完全重叠执行。而对于反相关和输出相关,相当于是由于资源问题所引发的相关,其实质是由于使用了共同的寄存器资源保存了不同的变量值所引发的,若采用更名的方法用两个不同的寄存器来保存这两个变量,则出现相关的两条指令就可并发执行, 不会引起性能损失,因此反相关和输出相关又称为伪相关(false dependence),可通过更名解决[1]。

在超标量体系结构中,数据相关的解决可通过静态调度方法和动态调度方法。静态调度由编译器将相关的指令调度开来以减少相关的发生和相关所造成的性能损失,而动态调度由硬件重新安排指令的执行顺序, 允许指令乱序执行, 以减少相关所引起的性能损失。常用的数据流调度方法是 Tomasulo 算法[1]。

3 静态调度和动态调度基本思想

5 段流水线有一个很大的局限性是按序流出和按序执行。如果相近的指令存在相关,就很可能会导致冲突,引起停顿,后面所有的指令也都停止了前进。为解决此问题,将 5 段流水线的译码(ID)段细分为两个段:[3][4]

(1) 流出:指令译码并检查是否存在结构冲突。若不存在结构冲突,就将指令流出。

(2) 读操作数:等待数据冲突消失(若存在),然后读操作数,并开始执行。

这样修改后的指令流出仍然是按序流出,但在读操作数段可能停顿或互相跨越,因而进入执行段时可能乱序执行。动态调度思想是对于一条会产生异常的指令来说,只有当处理机确切地知道该指令将被执行时,才允许它产生异常。

静态调度可在编译过程中减少相关的产生,而动态调度可根据处理器的动态信息发掘出更多的 ILP(Instruction Level Parallel)。动态调度的优势是能够在数据相关时,避免暂停流水线,特点就是流水线的乱序执行,指令的发射是乱序的,指令的完成也有可能是乱序的。由于流水线允许多条指令在同一时刻执行所以流水线结构需要改变,功能单元也需要改变。

4 记分牌动态调度方法思想

记分牌算法把简单流水线的译码段 ID 分解为:流出和读操作数。在没有结构冲突时,尽可能早地执行没有数据冲突的指令,实现每个时钟周期执行一条指令。如果某条指令被暂停,而后面的指令与流水线中正在执行或被暂停的指令都不相关,是这些指令可以跨越它,继续流出和执行下去。

每条指令的执行过程分为 4 段:流出、读操作数、执行和写结果。[3][5]

(1) 流出:如果当前流出指令所需的功能部件空闲,并且所有其它正在执行的指令的目的寄存器与该指令的不同,记分牌就向功能部件流出该指令,并修改记分牌内部的记录表。如果存在结构相关和 WAW 冲突,该条指令就不流出,从而消除了 WAW 冲突。

(2) 读操作数:记分牌监测操作数的可用性,一旦数据可用,就告之功能部件从寄存器中读出源操作数并开始执行。从而消除了 RAW 冲突,并可能导致指令乱序执行。

(3) 执行:取到操作数后,功能部件开始执行。当结果产生后,就告之记分牌它已经完成执行,该步相当于标准流水线中的执行段(EX)。

(4) 写结果:记分牌知道执行部件完成了执行,就检测是否存在 WAR 冲突。如果不存在,或者已有的 WAR 冲突已消失,记分牌就告之功能部件把结果写入目的寄存器,并释放该指令使用的所有资源。

记分牌中记录的信息由以下 3 部分构成。

(1) 指令状态表:记录正在执行的各条指令已经进入到了哪一段。

(2) 功能部件状态表:记录各个功能部件的状态。

(3) 结果寄存器状态表 Result:每个寄存器在该表中有一项,用于指出哪个功能部件将把结果写入该寄存器。

5 Tomasulo 算法思想和算法实现

5.1 算法基本思想[3][5]

寄存器换名是通过保留站和流出逻辑来共同完成,当指令流出时,如果其操作数还没有计算出来,则该指令中相应的寄存器换名将产生这个操作数的保留站

的标识。因此,指令流出到保留站后,其操作数寄存器或者换成了数据本身,或换成了保留站的标识,和寄存器无关。后面指令对该寄存器的写入操作就不会产生 WAR 冲突。

指令执行步骤可分为 3 步:

(1) 流出。从指令队列的头部出一条指令。若该指令的操作所需要的保留站有空闲,则把该指令送到保留站(r)。如果其操作数在寄存器中已经就绪,将这些操作数送入保留站 r,否则将产生的该操作数的保留站的标识送入保留站 r。此后,若被记录的保留站完成计算,它将直接把数据送给保留站 r。此过程实际上是进行了寄存器换名和对操作数进行缓冲,从而消除 WAR 冲突。

(2) 执行。或某个操作数还没有被计算出来,本保留站将监视 CDB,等待所的计算结果,且被放到 CDB 上,本保留站将立即获得数据。当两个操作数就绪后,本保留站就用相应的功能部件开始执行指令规定的操作。Load 和 store 指令的执行需要两个步骤:计算有效地址和把有效地址放入 load 和 store 缓冲器。Load 缓冲器中的 load 指令的执行条件是存储器部件就绪。Store 缓冲器中的 store 指令在执行前等到要存入存在器的数据到达。

(3) 写结果。功能部件计算完毕后,就将计算结果放到 CDB 上,所有等待该计算结果的寄存器和保留站都同时从 CDB 上获得所需要的数据。Store 指令在这一步完成对存储器的写入:当写入地址和数据备齐时,将它们送给存储器部件,store 指令完成。

5.2 算法实现[3][6]

给出 Tomasulo 算法中指令进入各阶段的条件以及在各阶段进行的操作数和状态表内容修改。各符号的意义如下:

r:分配给当前指令的保留站或者缓冲器单元。

rd:目的寄存器编号。

imm:按符号位扩展后的立即数。

Result:浮点部件或 load 缓冲器返回的结果。

Regs[]:寄存器组。

rs、rt:操作数寄存器编号。

RS:保留站。

Qi:寄存器状态表。

Op:当前指令的操作码。

对于 load 指令来说,rt 是保存所取数据的寄存器号;对于 store 指令来说,rt 是保存所要存储的数据的寄存器号。与 rs 对应的保留站字段是 Vj、Qj;与 rt 对应的保留站字段是 Vk、Qk。

(1) 指令流出

①浮点运算指令

进入条件:有空闲保留站(r)

操作和状态表内容修改:

if(Qi[rs]≠0){RS[r].Qj ← Qi[rs]}

Else {RS[r].Vj ← Regs[rs];RS[r].Qj ← 0};

if(Qi[rt]≠0){RS[r].Qk ← Qi[rt]}

else{RS[r].Vk ← Regs[rt];RS[r].Qk ← 0};

RS[r].Busy ← yes;

RS[r].Op ← Op;

Qi[rd] ← r;

②load 和 store 指令

进入条件:缓冲器有空元单元(r)

操作和状态表内容修改:

if(Qi[rs]≠0){RS[r].Qj ← Qi[rs]}

else{RS[r].Vj ← Regs[rs];RS[r].Qj ← 0};

RS[r].Busy ← yes;

RS[r].A ← Imm;

对于 load 指令:

Qi[rt] ← r;

对于 store 指令:

If(Qi[rt]≠0){RS[r].Qk ← Regs[rt];RS[r].Qk ← 0};

(2) 执行:

① 浮点运算指令

进入条件:(RS[r].Qj=0)且(RS[r].Qk=0);

操作和状态表内容修改:进入计算,产生结果。

② load 和 store 指令

进入条件:(RS[r].Qj=0)且 r 成为 load/store 缓冲队列的头部。

操作和状态表内容修改:

RS[r].A ← RS[r].Vj+RS[r].A;

对于 load 指令,在完成有效地址计算后,还要进行:

从 Mem[RS[r].A]读取数据;

(3) 写结果:

① 浮点运算指令和 load 指令

进入条件:保留站 r 执行结束,且 CDB 就绪。

操作和状态表内容修改:

$\forall x(if(Q[x])=x)\{Reg[x]\leftarrow result;Q[x]\leftarrow 0\};$
$\forall x(if(RS[x])Qj=r)\{RS[x]Vj\leftarrow result;RS[x]Qj\leftarrow 0\});$
$\forall x(if(RS[x]Qk=r)\{RS[x]Vk\leftarrow result;RS[x]Qk\leftarrow 0\});RS[r]Busy\leftarrow no;$

② store 指令。

进入条件:保留站 r 执行结束,且 RS[r].Qk=0

操作和状态表内容修改:

$$Mem[RS[r].A]\leftarrow RS[r].Vk$$
$$RS[r].Busy\leftarrow no;$$

如果能够准确地预测分支,采用 Tomasulo 算法将获得很高的性能。

6 Tomasulo 算法和记分牌算法的异同

(1) 相同之处[7]:

两个算法消除 RAW 竞争的思想相同,Tomasulo 算法采用了记分牌方法的动态调度的基本思想,多条指令处于发射状态,等待条件满足,可以不按顺序执行。

(2) 不同之处[8]:

Tomasulo 方法通过寄存器换名过程可消除 WAR 和 WAW 竞争。记分牌方法可检测 WAR 和 WAW 竞争,一旦检测到存在 WAR 和 WAW 竞争,通过插入停顿周期来解决此竞争。因此,记分牌方法不能消除 WAR 和 WAW 竞争。

7 结束语

在超标量体系结构中,存在着数据相关和控制相关问题,数据相关的解决可通过静态调度方法和动态调度方法。静态调度由编译器将相关的指令调度开来以减少相关的发生和相关所造成的性能损失,而动态调度由硬件重新安排指令的执行顺序,允许指令乱序执行,以减少相关所引起的性能损失。常用的数据流调度方法是 Tomasulo 算法。讲解了 Tomasulo 算法和记分牌方法的动态调度的基本思想,对算法进行了比较分析。

参考文献:

[1] 莫壮坚,李振.超标量微处理器研究[J].海南师范学院学报(自然科学版).2004,17(4):347-351.

[2] 邓正宏,康慕宁,罗旻.超标量微处理器研究与应用[J].微电子学与计算机.2004,21(9):59-63.

[3] 张晨曦.计算机系统结构教程[M].北京:清华大学出版社.2009.

[4] 宋辉.量子计算机体系结构及模拟技术的研究与实现[D].中国人民解放军国防科学技术大学.2003.

[5] 沈立.动态 VLIW 体系结构关键技术研究与实现[D].国防科学技术大学.2003.

[6] 王蓉晖.指[9]令级并行性开发关键技术的研究与实现[D].国防科学技术大学.2002.

[7] WILLIAMSTALLINGS.COMPUTER ORGANIZATION AND ARCHITECTURE:Design for Performance (FourthEdition).北京:清华大学出版社,2002.

[8] 周敏,付慧生,李雪峰.基于流水线的 RISC 微处理器设计[J].大众科技;2006,10(5):55-57.

作者简介:王磊(1984-),男,助教,硕士研究生,主要研究方向为并行计算。

(上接第 22 页)

了地域的限制,当生产设备出现故障时,可通过远程诊断技术,及时找到系统问题,避免了技术人员两地奔波,只要拥有权限,技术人员就可以通过互联网访问远程监控系统,对生产设备进行维护。

参考文献:

[1] 许家忠,尤波,胡海燕,王雄健.内加热固化环氧玻璃钢管道制造系统[J].材料科学与工艺,2007,15(1):102-106.

[2] 李律松,马传宝,李婷.Visual C# + SQL Server 数据库开发与实例[M].北京:清华大学出版社,2006.

[3] 李涛,刘凯奎,王永蛟.VISUAL C+++SQL SERVER 数据库应用系统开发与实例[M].北京:清华大学出版社,2006.

作者简介:许家忠(1977-),男,教授,硕士生导师,研究方向:复合材料壳体高效成型及机电控制方向。