基于 RISC-V GCC 编译器的指令延迟调度

余晓江　罗欣
（西华师范大学　四川省南充市　637000）

摘　要：本文通过对 RISC-V GCC 编译器的改进和优化，为进一步研究和优化 RSIC-V GCC 奠定了编译器基础，更为快速研究新型的 RISC-V 指令集架构处理器起着重要的推动作用。在深入分析 RISC-V GCC 中表调度算法得出编译器可以计算出指令停顿的位置和拍数，指令延迟调度技术是在编译器中静态调度指令达到动态调度的效果以避免流水线冒险和特殊时序带来的问题。验证了使用指令延迟调度技术的编译器的性能和正确性，并在火苗原型系统和超导模拟器上运行测试通过。

关键词：RISC-V 指令集；编译器；延迟调度；指令调度

1 引言

2014 年加州大学伯克利分校（University of Cal-I fornia at Berkeley, 以下简称 UCB）的研究人员 Krste Asa-novic、Andrew Waterman、Yunsup Lee 设计并发布了 RISC-V 指令集架构[1]。RISC-V 基于精简指令集计算（RISC）原理构建的开放 RISC-V 指令集架构（RISC-V instruction set architecture 简称 RISC-V ISA）[2]。RISC-V 指令集架构采用的是精简指令集，同时有着非常良好的可扩展性，并且 RISC-V 生态系统迅速的发展，促进了这种新型的 RISC-V ISA 在各大处理器设计中得到应用。RISC-V 开源自由协议受到个人、科研团队以及商业公司的青睐，各大公司纷纷加入 RISC-V 联盟，如蜂鸟、中科院计算所、英伟达、华为、谷歌、阿里巴巴等等[3][4][5]。

大多数早期超标量处理器[6]都是顺序执行，RISC-V 处理器设计初期均采用单发射顺序执行的流水线工艺[7]。

流水线技术之所以能提高性能，其本质是利用了时间上的并行性，让原本应该先后执行的指令在时间上一定程度的并行起来，然而这样也会带来一些冲突和矛盾，进而可能引发错误，这种情况就被称为流水线冒险。数据冒险是流水线冒险中最常见的一种，数据冒险是指令所需要的数据矛盾，如果一条指令需要某个数据而该数据正在被之前的指令操作，那这条指令就无法执行，就导致了数据冒险。针对流水线冒险其中一种解决方案是使用流水线停顿来延迟下一条指令。从而避免流水线冒险带来的程序执行错误。

在传统处理器设计中，流水线借助于一个执行单元队列，让暂未满足各类资源的指令在此等待一定周期，或者采用流水线停顿技术 (stall) 让整个流水线上相关的资源均陷于一定周期的等待状态，直至问题解除[8][9]。等待的周期主要取决于两个因素：前驱指令的执行开销、以及当前指令所需硬件资源。而这些恰恰是编译器静态指令调度策略中的核心影响因素。因此，如果借助于静态调度策略，为当前指令去权衡计算它需要等待的时间，适当调整指令的位置，在指令排布过程中拉长它和前驱的距离，我们就能保证它在正确的时间完成译码以及到达执行功能部件。

综上所述，虽然成熟的处理器大部分使用动态调度来避免流水线冒险，但是基于 RISC-V 指令集的处理器目前尚未成熟，都处在研究设计阶段，所以使用 RISC-V ISA 的处理器研发时，为缩短研发周期和研发成本。静态调度是解决流水线冒险最有效的方法。本文提出一种面向 RISC-V 处理器的指令调度技术——指令延迟调度技术，用于向 RISC-V 处理器研究提供一种实用、有效的软件解决方案。在 RISC-V 处理器研发处于萌芽期的大背景下，编译器延迟调度技术是一种实用、有效的途径，使编译器的研发不受硬件时序控制设计与验证的约束，直接在具有基本执行功能的 FPGA 或者软件模拟器上进行研发，从而使软件研发与处理器研发同步前行。

2 指令延迟调度技术

本文提出一种基于 RISC-V 处理器的编译器调度技术——指令延迟调度技术，它通过一个保守的资源开销评估模型对当前指令的正确发射时间进行评估。通过指令的静态调度对流水线上的指令到达执行部件的时刻进行干涉，从而有效避免由于指令提前到达执行部件所引发的流水线冒险。指令延迟调度技术是一种实用、有效的软件解决方案，使核心软件可以与处理器设计同期同步开展。优化后的表调度分析算法能够准确地记录每一个基本块中指令流中需要插入延迟指令的位置和数量。通过本地延迟指令生成算法，生成占用拍数但又不做任何操作的 NOP 指令[10]。指令延迟调度技术根据表调度算法计算出的数据对表调度之后的指令流进行再次调度，使每一拍都有指令发射，从而避免流水线冒险带来的执行错误。经过指令延迟调度的程序在处理器上不需要进行动态调度即可正确高效的执行。为了不影响编译器的功能模块和方便进行对比测试。将指令延迟调度封装在编译选项 -fsched-delay 中，在编译时，通过使用编译选项 -fsched-delay 来控制编译器使用指令延迟调度技术。

（1）表调度分析算法。要对表调度后的 RTL 进行指令延迟调度，需要优化表调度的分析算法。通过指令调度数据流 DUMP 算法的输出数据流，直观的显示了指令信息。RTL 中间表示阶段所有指令是由一个结构体存储即 INSN。要对指令进行延迟调度，首先需要了解表调度的调度原理，如图 1 所示。指令 INSN 已提交到计划中，从“Ready”列表移动到“Scheduled”列表。当发生这种情况时，“Pending”列表中的 INSN 满足其依赖关系并移动到“Ready”列表或“Queued”集合，具体取决于是否已经有足够的时间使其准备就绪；随着时间的推移，准备就绪的 INSN 从“Queued”移动到“Ready”列表；“Pending”列表（P）是未安排的 INSN_FORW_DEPS 中的 INSN，即准备好，排队和挂起的 INSN。“Queued”集合（Q）由变量 insn_queue 实现。“Ready”列表（R）由变量 ready 和 n_ready 实现。“Scheduled”列表（S）是由此传递构

●基金项目：西华师范大学英才科研基金项目（No. 17YC163）。

```
15--> b  0: i1095 a5=[sp+0x4]                :alu
16--> b  0: i 534 s1=s11                     :alu
 Advancing clock by 1 cycle[s] to 18
18--> b  0: i 781 a2=a5                      :alu
```

（a）RISC-V GCC编译器指令调度数据流片段

```
15--> b  0: i1095 a5=[sp+0x4]                :alu
16--> b  0: i 534 s1=s11                     :alu
17--> b  0: i 1342 nop                       :nothing
18--> b  0: i 781 a2=a5                      :alu
```

（b）指令延迟调度优化的RISC-V GCC编译器指令调度数据流片段

图 1：编译器指令调度数据流对比

```
lw   a5,4(sp)
mv   s1,s11
mv   a2,a5
```

（a）RISC-V GCC编译器编译出的汇编码片段

```
lw   a5,4(sp)
mv   s1,s11
nop
mv   a2,a5
```

（b）指令延迟调度优化的RISC-V GCC编译器编译出的汇编码片段

图 2：编译器编译出的汇编码对比

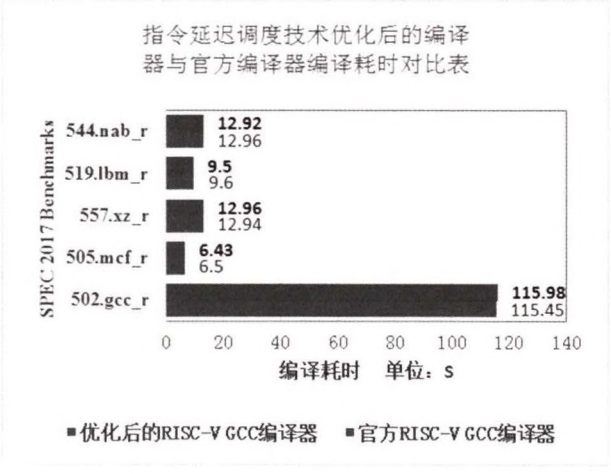

图 3：指令延迟调度技术优化后的编译器与官方编译器编译 SPEC2017 Benchmarks 的耗时对比图

建的新 INSN 链。当选择最好的调度 INSN 时，转换（R-> S）在 schedule_block 的调度循环中实现。当 INSN 从就绪列表移动到调度列表时，转换（P-> R 和 P-> Q）在 schedule_insn 中实现。随着时间的推移或引入停顿，转换（Q-> R）在 queue_to_insn 中实现。为了减小对编译器的影响，对表调度分析算法进行优化，计算出延迟调度的相关信息并记录保存，传递到指令延迟调度中。

（2）延迟指令生成算法。由于指令流里的指令是由源码翻译过来的，从一开始转换到 RTL 时，就生成了双向链表，经过表调度时，指令队列进行指令调度后依然是一个双向链表。那么在进行延迟调度时，要在双向链表中插入延迟指令 NOP。就要对双向链表进行维护。每一条指令都有一个前驱和后继，当前指令、前驱指令和后继指令都是相互依赖的。当 insn 需要延迟时，首先就要断开双向链表 prev 和 insn，将 nop 指令放在 prev 和 insn 之间。把 insn 的前驱设置为 nop，prev 的后继修改为 nop，并把 nop 的前驱设置为 prev，后继为 insn。根据机器模型获取 nop 指令的 RTL 结构，设置 nop 指令所属的基本块为 insn 的基本块。

（3）指令延迟调度算法。指令延迟调度算法根据表调度分析算法记录并传递来的数据使用延迟指令生成算法在需要延迟的指令前插入足够的空操作指令 NOP 来占用无指令的拍数。为尽量避免对编译器其他功能的影响，我们选择在编译器表调度完成之后，RTL 翻译成汇编码之前，进行指令延迟调度。首先判断是否在编译时使用了 -fsched-delay 编译选项，以下所有操作都是在使用 -fsched-delay编译选项的情况下实现。接着判断当前是否属于最后一次调度，即前面说的重载后调度（after reload）。当两个条件同时满足时，对当前基本快中已经调度好的指令流进行遍历，寻找需要延迟的指令。当遍历到 INSN 的编号 INSN_UID 为表调度分析算法传来数据中的 INSN_UID 时，即这条 INSN 需要在发射前插入 NOP 延迟，根据数据中记录的延迟拍数循环插入 NOP 指令。调用延迟指令生成算法进行指令生成。最后对指令在调度过程中的 LUID 进行维护，以保证调度过程中每一条指令在调度过程中都有唯一的识别 ID。

3 实验结果与分析

（1）指令延迟调度技术是通过静态调度中使用停顿避免流水线冒险带来的执行错误，经过应用该技术的 RISC-V GCC 编译器编译程序时，使用 -fsched-verbose=n 编译选项可输出指令调度过程，能够直观的看到指令延迟调度的结果，如图 1 所示。

另一方面，指令延迟调度技术在静态调度中完成了空拍指令的填充，生成的汇编码中会明确显示 NOP 在相应的位置，如图 2 所示。

经过如图 1 和图 2 的试验验证，指令延迟调度技术能够在指定拍数处填充 NOP 指令，并且正确的生成对应的汇编码，来延迟下一条指令。编译器的性能是在对编译器改进后的综合评估，主要通过编译时间来进行比较，使 SPEC2017 作为测试用例，测试了使用指令延迟调度技术的 RISC-V GCC 编译器编译时间。实验编译器运行环境为：32 核 Intel Xeon E7-4809 主频 2.00GHz 的 CPU，64 位 ubuntu14.04 系统，一级指令 / 数据 Cache 各 32K，二级 Cache256K。实验结果如图 3 所示。

图 3 中分别使用指令延迟调度技术优化后的 RISC-V GCC 编译器和官方 RISC-V GCC 编译器在相同条件下编译 SPEC2017 中使用 C 语言编写的 Benchmarks，由图可见，优化后的编译器和原编译器编译相同 Benchmarks 耗时相差不超过 1s，实验证明，指令延迟调度技术对 RISC-V GCC 的性能影响很小，达到了设计初期最小化编译器影响的要求。为了验证优化后的编译器编译出的程序可以正确的在 RISC-V 处理器上运行，以 stream 作为测试程序，分别在超导模拟器和火苗原型系统上测试通过，能够正确的执行完成。超导 v1 版本模拟器：位模式 32 位，译指队列大小 151，一级 Cache128byte，二级 Cache32K，寄存器个数 16。火苗原型系统：4 核 Labeled RISC-V 系统，100 MHz rocket 核心， 16KB L1 Icache,

混合型养老社区综合服务平台之智能终端系统的设计

袁赛杰　胡泽斐　丛玉华　谢玲　周怡诚
（南京理工大学紫金学院计算机学院　江苏省南京市　210023）

摘　要：本文着眼于社会养老问题，针对混合型养老社区综合服务平台设计其智能终端系统。智能终端系统包括助老终端和数据监测两个子系统，助老终端负责为老年人提供娱乐和护理，包含人机交互和运动控制两部分内容。数据监测负责采集室内环境状态和老年人身体状态，包含环境监测和体征监测两部分内容。系统基于嵌入式、传感器、C语言等技术，通过传感器进行数据采集，主控器进行数据处理分析、指令生成，人机接口设备进行数据显示和提醒。

关键词：混合养老社区；助老终端；数据监测；传感器；主控器

1 引言

随着老龄化问题的日益严重，一系列养老问题也引起了人们的关注：家庭支持资源不足、老年人的生活和医疗护理不足、护理的设施不足、老龄人的情感需求也在不断增加，面对这些难点，如何提高老龄人的生活质量就显得愈加重要了。为此中共中央强调要以家庭为基础建立多层次的护理服务系统，积极开展老龄化行动，推动养老的多样化。“家庭养老”和“社区养老”等实施表明，社区养老将是未来的主要趋势。混合型养老社区可以帮助老年人护理机构和社区大大提高管理效率[1]。混合养老社区中老年人生活系统的概念和技术的介入将使未来的养老模式更加人性化。养老社区将来可以使用智能设备实时监控老年人的生活和健康状况，并做到信息同步，家庭成员可以远程了解老年人的信息状况，家庭成员也可以通过云平台发布需求，服务公司可以根据需求为老年人提供现场服务[2]。本文将设计智能终端系统即为养老社区综合服务系统的智能终端部分。

2 系统总体方案

2.1 功能分析

智能终端系统要求能够在老年人的日常生活中起到陪护和与老年人相关状态的监测功能，因此系统应包含两个子系统：助老终端系统和数据监测系统。系统功能如图1所示。

助老终端系统负责老年人的居家娱乐和简单护理，娱乐部分可满足老年人精神方面需求，通过在助老终端上设计人机交互模块来

16KB L1 Dcache，2MB L2 cache，16GB DDR4 SO-DIMMs，Linux 4.6.2内核。

4 结束语

指令调度在编译器中是非常关键的环节，指令延迟调度技术是RISC-V GCC编译器在RTL一级的指令调度优化，在RISC-V处理器研发过程中作为编译器辅助有着重要作用，尤其是在超导计算机研究中作为编译器优化研究意义重大。本文通过对RISC-V GCC编译器的改进和优化，为进一步研究和优化RSIC-V GCC奠定了编译器基础，更为快速研究新型的RISC-V指令集架构处理器起着重要的推动作用。

参考文献

[1]A Waterman,Y Lee,DA Patterson, et al.The RISC-V Instuction Set Manual,Volume I:User-Level ISA [J].Eecs Depart-ment,2011,7(9):475.

[2]Krste Asanovic, David Patterson.The Case for Open Instruction Sets[J].MICROPROCESSOR report,2014(8):1-7.

[3] 胡振波．手把手教你设计CPU RISC-V处理器篇[M]．北京：人民邮电出版社，2018:54-60.

[4] 胡振波.RISC-V的爆发，是中国芯片产业的一次机遇！[J]．单片机与嵌入式系统应用，2019,19(07):1-3.

[5] 倪光南．迎接开源芯片新潮流[J]．信息安全与通信保密，2019(02):11-13.

[6] 邓凯伟．超标量处理器高效发射队列的设计及与实现[C]．中国计算机学会．第十八届计算机工程与工艺年会暨第四届微处理器技术论坛论文集．中国计算机学会：中国计算机学会计算机工程与工艺专业委员会，2014:177-181.

[7] 任永青．逻辑核动态可重构的众核处理器体系结构[D]．中国科学技术大学，2010.

[8] 贾琳，樊晓桠．32位RISC微处理器流水线设计[J]．计算机工程与应用，2005，41(14):115-117.

[9]Xiao Z. Optimizing pipeline for a RISC processor with multimedia extension ISA[J]. Journal of Zhejiang University Science A,2006,7(2):269-274.

[10]Schirmer,Oskar.NOP - A Simple Experimental Processor for Parallel Deployment[J].2016.

作者简介

余晓江（1992-），男，羌族，四川省绵阳市人。硕士研究生在读。研究方向为编译器优化。

罗欣（1993-），女，四川省南充市人。硕士研究生在读。研究方向为图像处理。