计算机科学2004Vol.31№.12

一种面向RISC体系结构的全局延迟槽调度策略*)

蒋 奕 吴承勇 张兆庆 冯晓兵
(中国科学院计算技术研究所体系结构室 北京100080)

摘 要 精简指令集(RISC)体系结构常使用延迟槽来减少因分支而造成的延迟。而优化编译器如何高效地利用延迟槽对于性能来说十分重要。本文对延迟槽调度中调度范围、所处编译阶段等问题进行了分析,对全局延迟槽调度可能出现的冲突及候选指令的区域进行了探讨,并提出了一种全局延迟槽调度算法,实验结果证明了它的性能和健壮性。

关键词 RISC体系结构,延迟槽,全局延迟槽调度,候选指令范围

Global Scheduling Technique of Delay Slot on RISC Architecture

JIANG Yi WU Cheng-Yong ZHANG Zhao-Qing FENG Xiao-Bing
(Institute of Computing Technology, Chinese Academy of Sciense, Beijing 100080)

Abstract RISC architecture is widely adopted in processor design. Delay slot has been used in hardware to reduce latency and miss penalty of branch prediction. It is vital for an optimizing compiler to take advantage of delay slots effectively. The paper analyze the scheduling scope of delay slots, resolves possible conflicts in global scheduling of delay slots and improves the selection of instruction candidates. Then we presents a global scheduling algorithm of delay slots. The experiment results show that our approach is effective and robust.

Keywords RISC architectures, Delay slot, Global delay slot scheduling, Searching scope of candidates

1 引言

精简指令集(RISC)体系结构是应用非常广泛的一类体系结构,在采用这种体系结构的流水处理器中,分支指令是提高性能的一个重要障碍,这是因为从程序上看,其下一条指令的地址将被分支指令的结果所决定,但是这个结果需要在分支指令执行到比较后的流水级时才能产生,中间的延迟会造成流水线停顿等待分支结果,从而直接影响后续指令序列的执行,降低指令并行度[1]。延迟槽结构有助于解决此类流水线控制相关而带来的性能问题。其原理是,如果分支指令的后面放上不受分支指令结果影响的指令,那么上面所提到的延迟就会被这样的指令屏蔽,流水线依然可以保持顺畅,延迟槽就是用来存放这些指令的结构。但是硬件提供的这种结构要在实际程序执行中真正获得好的效果,需要编译的支持,如果在延迟槽中填充有用的指令,那么性能就可能提高,而如果找不到合适的指令使得延迟槽不得不填充空操作指令的话,那么控制相关带来的性能损失仍然存在。因此,如何让延迟槽得到尽量的填充以及如何将延迟槽调度融入到整个编译优化的框架中,成为获得高性能和健壮的编译器的一个重要因素。

在传统对延迟槽的使用上,大多数编译器采用的是局部的调度,比如说gcc(GNU C Compiler)在汇编阶段做的延迟槽调度。在此之后,Gross 和 Hennessy[2]提出了全局的延迟槽调度,在理论上将指令跨越基本块边界填充延迟槽进行了分类并分别阐述了可能的效果。Muchnick[3]进一步针对分支指令性质以及延迟槽执行规则的不同对全局延迟槽调度进行了更细致的划分。

本文主要对全局和局部延迟槽调度进行了比较,并在整个编译器的框架内讨论了延迟槽调度和其他优化阶段的关系从而确定了它在编译器优化过程中的位置。针对前人关于全局延迟槽调度的分类,我们给出了具体的填充办法,并阐述了可能出现的冲突和预防措施。然后,为了减少编译时间以及降低实现难度,我们还利用编译器中已有优化的条件,对候选指令的区域进行了改进。最后,我们设计并实现了一个全局延迟槽调度算法。

本文第2节是对延迟槽调度中的主要问题进行分析和研究,以及我们做出决策和改进;第3节提出了一种全局延迟槽调度算法;第4节给出了实验结果;最后是结论和对未来工作的展望。

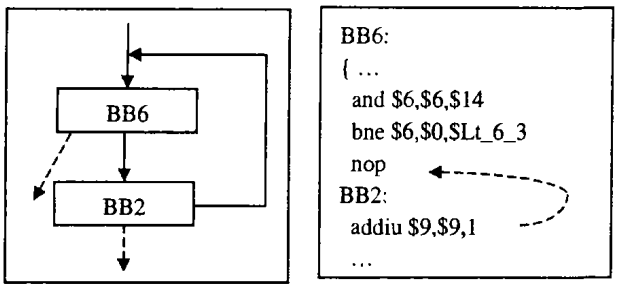
2 延迟槽调度:问题的提出和策略的选择

2.1 延迟槽调度的范围

延迟槽调度的范围分局部和全局两种,前者是指被填充的指令从分支所结束的基本块中选取,如果没有合适的指令可供选择,那么就填充空操作指令;而全局的延迟槽调度[2]在发现本基本块中没有合适指令可填充延迟槽后,还会基于一定规则从其他基本块中选择合适的指令填充,也就是说它允许跨越基本块边界的代码移动,如果这个过程也告失败,那么再选择填充空操作指令。

下面,我们比较一下两者的优劣。前者由于实现简单,只需要做比较简单的局部数据流分析,因此它可以在编译或者汇编的时候介入,而且也被很多编译器所采用;而后者在实现上有一定的复杂度,需要有全局的数据流分析,但是它可以增加延迟槽有效填充率(即延迟槽中填充有用指令的比率),从而得到更好的优化效果。图1是一个来自于spec2000基准程序parser的例子。

*)本课题得到国家863计划软件重大专项(2002AA1Z2104)的支持。



(a) (b)

图1 一个延迟槽全局填充的例子

在这当中,图 a)所示循环是程序控制流图热路径的一部分,根据实际运行的统计,这个循环的执行次数在千万次以上。图 b)是利用汇编器中的局部延迟槽调度所产生的结果,在 BB6中,它找不到可以填充的指令,所以只能填充空操作指令,但是其实,我们通过分析发现,把 BB6的后继基本块即 BB2的第一条指令移进 BB6的延迟槽(即沿图示的虚线移动)是可行的,因为它不但是一个安全的移动,而且可以据此获得执行时间上千万指令周期的减少。

在实际的程序当中,包含多个基本块,即循环体中有分支的循环是很常见的,所以,采取全局延迟槽调度的策略对于一个高性能编译器来说是合适和必要的。

2.2 延迟槽调度阶段的选择

延迟槽调度作为后端优化的一部分,其位置以及与优化其他阶段的关系将会对优化的效果产生影响。在后端优化当中,指令调度(包括全局和局部)以及寄存器分配是其中最为典型的优化,图2是它们所遵循的一种典型的顺序。

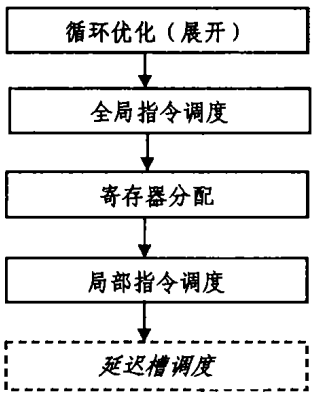


图2 一种典型的编译器后端优化流程

全局延迟槽调度可以选择纳入全局指令调度的框架,这样的好处是可以只作一遍全局的数据流分析,但是这种策略也有局限性:由于延迟槽的大小有限,因此,一旦指令填充进去,其位置的变动是不容易的,过早地进行延迟槽的填充,会给后续优化的寄存器分配阶段对溢出代码处理以及局部指令调度等基于代码移动的优化都带来相当的麻烦。

所以,延迟槽调度应该处于指令调度和寄存器分配的后面,这时已经到了后端优化的晚期,其调度产生的结果就不会对原来存在的优化阶段产生负面影响。

2.3 被填充指令相关问题的分析和改进

2.3.1 *被填充指令移动的类型和对策* 对于应用全局延迟槽调度的目标基本块来说,其延迟槽中填充的指令可以有三个来源,即本基本块、跳转目标基本块和非跳转目标基本块[2]。据控制流的情况,延迟槽的填充又可以分为局部填充、移动性全局填充和复制性全局填充。局部填充指的是从本基本块内选取指令填充,如果从基本块的后继基本块中选择指令填充,而且这个后继基本块没有其他前驱,那么它就是一个移动型全局填充,反之就是一个复制型全局填充。表1表示的是针对指令来源和控制流情况的组合所给出的处理办法(假设延迟槽已经填充了空操作指令)。

从表中可以看出对于局部填充和移动型全局填充,我们可以直接将被选指令移动到相应延迟槽中替代空操作指令,这样不但有性能上的提高,还可以减少代码体积。对于来自跳转目标基本块的复制型全局填充,为了保证在执行到此基本块的其他前驱基本块时的正确性,不能做简单的移动,而是需要拷贝相同的指令填入延迟槽并修改跳转目标。对于来自非跳转目标的情况,由于两个基本块在实际代码排列上存在自然的衔接关系,我们只需简单地删除空操作指令即可。

表1 各种指令填充情况的讨论

填充类型	填充影响因素	填充来源	填充办法
局部填充	代码体积,性能	本基本块	选中指令移动替代空操作指令
移动型全局填充	代码体积,性能	跳转目标基本块和非跳转目标基本块	选中指令移动替代空操作指令
复制型全局填充	性能	跳转目标基本块	选中指令拷贝一份替代空操作指令,同时修改跳转目标.
	代码体积,性能	非跳转目标基本块	直接删去空操作指令

2.3.2 *基本块之间延迟槽调度可能的冲突及预防* 由于在全局类型的填充中涉及到跨越基本块边界的移动,在延迟槽调度当中要考虑所有一般全局代码移动中保证合法性的措施,其中包括:1)代码移动不能违背现有的依赖关系;2)由跳转分支的一侧填充进目标基本块的延迟槽中的指令,其目标操作数在分支的另一侧一定是不活跃的;3)被填充的指令本身不能再是分支指令;4)被填充的指令不能是被规定必须处于同一个基本块的几条指令中的一条等。

除此之外,基于全局的调度还有可能造成基本块之间延迟槽调度的冲突,下面是一个实际的例子:

图3a)中,由于做了复制型全局填充,BB2的一条指令 op 沿着 a)中虚线所指方向被复制并填充到前驱 BB1的延迟槽中,与此同时,跳转目标也从 lable 变成了 lable+4,即图 b)情况,而随后在考虑 BB2的延迟槽时,又将第一条 op 沿 b)中虚线方向做局部填充到达了分支指令的后面,即图 c)的情况,显然,从 a)到 c),虽然独立地看每一步变换都是合法的,但是它们之间有冲突,使得程序的语义发生了变化。因此,针对这种情况,我们对于做复制型全局填充的指令设上标志,避免它又被填充到本基本块内。

2.3.3 *候选指令范围的改进* 传统的延迟槽调度候选范围是整个基本块,拿局部填充来说,具体做法是从跳转指令开始往上搜索,直到搜索到一条合适放进延迟槽的指令或基本块的头。需要指出的是,一般的高性能编译器中有全局和局部指令调度,在此基础上,我们发现可以把候选指令的范围从基本块缩小到指令,即对于本基本块内的局部填充,只检查跳转指令的前一条指令是否合适放进延迟槽。

简单地说,通过对指令调度算法的分析,我们得到,延迟槽调度的输入指令序列可以认为是优化的,如果这时选取的

被填充指令不是来自于跳转指令的前一条，那么填充后的结果就是对原来优化的指令序列进行了重排，这就很可能影响性能。事实上，影响指令序列执行时间的因素是数据流图的关键路径长度，根据上面的分析初始序列的关键路径是最短的。现在假设放进延迟槽的指令不是跳转指令的前一条指令，那么，如果这条指令恰好在关键路径上，则在它被放入延迟槽之后执行时间拖后了，这样很可能会增加关键路径的长度，从而破坏了指令序列的优化性。反之，如果它不处于关键路径上，那么根据关键路径的定义，其延迟可以被屏蔽，这样即使将它填充进延迟槽，也不能带来性能的提高。

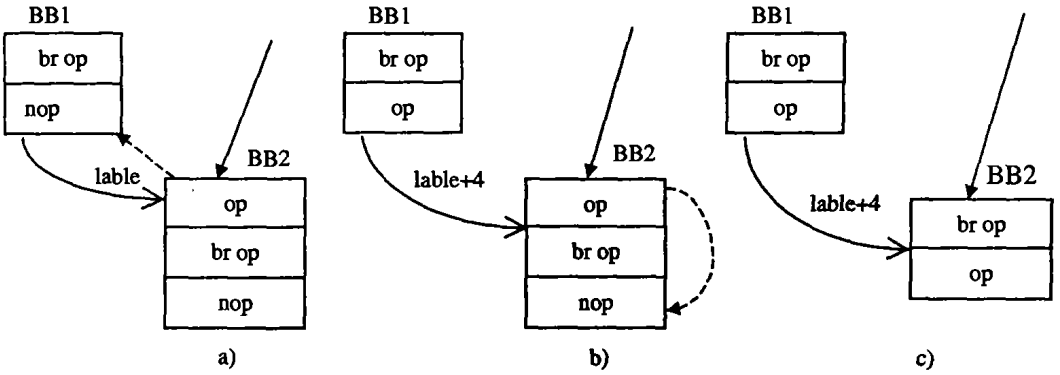


图3　各种指令填充情况的讨论

类似地，我们很容易推之，我们的候选指令也可以限于后继基本块的第一条指令。由于实际资源的限制，关键路径上指令的执行可能会因此推后，即被认为应该同时发射的指令被分在了不同周期发射，这样，它们之间的重排不影响序列的优化性，传统方法搜索整基本块而恰好找到这类指令的话也有可能获得更好的效果。但是对比传统做法，通过对候选指令范围数量级的缩小，可以提高编译速度，同时降低实现难度，是一种有实际意义的改进。

3　全局延迟槽调度算法

依照上面一系列问题的分析和决策，我们可以给出一个全局延迟槽调度的算法实现如下：

1. 做全局数据流和控制流分析。
2. 按拓扑序遍历所有基本块，并对于每一个基本块的延迟槽，选择候选填充指令的来源，优先考虑本基本块，如果失败，则按照分支概率从大到小选择后继基本块。
3. 通过对控制流的分析，确定填充的类型。
4. 根据2.3.3讨论的候选范围，选取候选指令做合法性检查，如果合法，则按照表1进行填充变换，标记本基本块为调度完成，为否则回到3。
5. 如果是复制型全局填充，把相应 op 置上标志。

在不违反拓扑序的原则下，我们优先选择执行频率高的基本块提前调度，这样做的目的是使得它潜在的延迟槽被填充的概率更大。

4　实验结果

根据上文中的分析和决策以及给出的全局延迟槽调度算法，本文作者在龙芯 I 编译器中完成了实现，通过对2000多个 C 程序以及 spec2000基准程序的测试，其健壮性得以证实。

龙芯 I 处理器是一款兼容 MIPS 的芯片[4]，龙芯 I 编译器就是基于上述芯片的一款优化编译器，它由 ORC 移植而来，在后端包括基于 Bernstain 调度框架[5]的指令调度等现代编译器优化手段。

我们的实验环境是：交叉编译，即在 x86体系结构的机器下用 O2级别优化选项生成可执行码，在龙芯1机器上执行并统计标准测试程序 Spec2000的性能，龙芯 I 机器的配置情况为：200MHz 的主频，256M 内存，一级的指令和数据 cache 都是8k。

图4和图5分别表示了相对于原来编译器中使用的 gcc 汇编阶段的延迟槽调度，我们在改进后延迟槽的填充率以及相

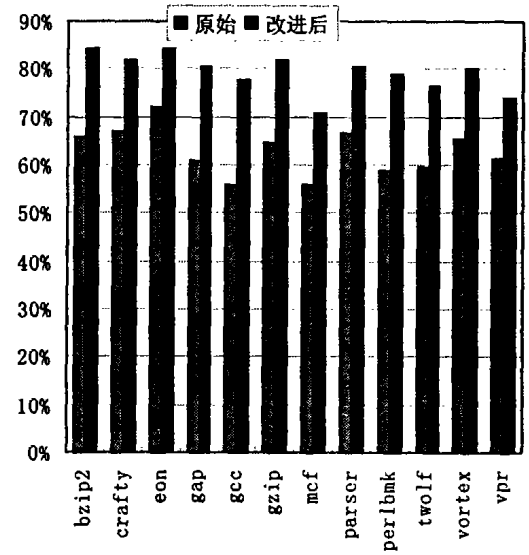


图4　改进后延迟槽填充率的变化

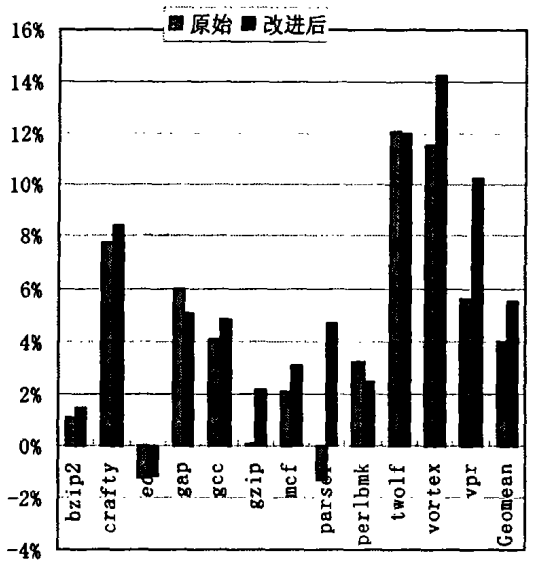


图5　改进后相对不调度性能加速的变化

对于不填充延迟槽(即全部填充空操作指令)所得到的加速比的变化，在图5中，12个例子延迟槽填充率均有明显提高，平均增幅达16.50%，这是性能提高的主要原因之一。在图5中加速

比衡量是基准程序的运行性能比率(ratio),对比原来的调度,经改进后可以看到在 bzip2,parser 等9个例子中,性能有了不同程度的提高。总体上,改进后,相对于不填充和改进前,我们的调度获得了平均5.5%和1.3%的性能提高。

结论及未来工作展望 从得到的试验结果来看,全局的延迟槽调度区域以及我们在调度阶段等问题上所选择的策略对性能提高是有成效的,对于调度中出现冲突的预防也十分必要。测试程序中改进后提高水平呈现参差不齐的状况,这是因为性能的提高取决于很多因素,除了和填充率大小有关外,还与例子本身分支的多少以及改进后额外的填充所处路径的热度有关。而对比改进前的调度,gap,perlbmk 和 twolf 两个例子还略有下降,这是因为我们全局的填充有一定的"投机性",因此今后的工作可以结合 profiling 信息设计精确选择填充指令的启发性算法来降低风险。

另外,还有一个现象值得注意,在进行延迟槽调度后,parser 和 eon 的性能有所下降,这可能和程序某些动态行为有关,计划在后续工作中进行讨论。

参 考 文 献

1 Hennessy J L, Patterson D A. Computer Architecture: A Quantitative Approach. Morgan Kaufmann Publishers, San Mateo, CA, 1990

2 Gross R T, Hennessy J L. Optimizing Delayed Branches, In: Proc. of the 15 th Annual Workshop on Microprogramming, Oct. 1982

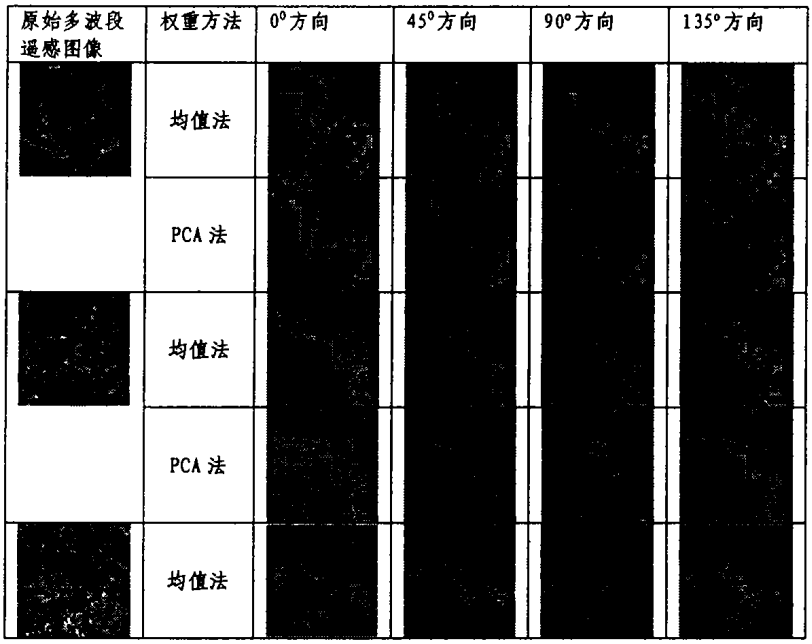
3 Muchnick S S. Advanced Compiler Design and Implementation. Morgan Kaufmann Publishers Inc. 1998

4 中科院计算所 CPU 组. 龙芯-1微处理器用户手册(版本1.0). 2002

5 Bernstein D, Rodeh M. Global Instruction Scheduling for Super-Scalar Machines. Programming Language Design and Implementation, 1991

(上接第163页)

移量的重新组合,子图像与原始的多波段遥感图像的比例问题在这里都不予过多的考虑。



总结 在本文中,我们提出了当对多波段遥感图像特征提取时,一种新的确定特征向量权重的思想;明确阐述了基于灰度共生矩阵的多波段遥感图像特征提取的方法。由于灰度共生矩阵能比较精确描述纹理的定量信息,因此用灰度共生矩阵方法描述纹理特性是可行的,实验结果也证明了这一点。由实验结果还可以看出,我们对特征空间的数据服从高斯分布的假设是合理的。如果能在本文所提出的各波段权重选取方法的基础上考虑到像素的距离,位移量的重新组合,子图像与原始的多波段遥感图像的比例等问题,理论上将进一步提高多波段遥感图像分类的精度,这些是下一步准备解决的问题。

参 考 文 献

1 肖温格 R A 著. 遥感中的图像处理和分类技术[M]. 北京:科学出版社,1991

2 骆玉霞,陈焕伟. 遥感图像的特征提取与选择研究. 信息记录材料,2002,3(2)

3 Lu C S, Cheng P C, Chen C F. Unsupervised texture segmentation via wavelet transform. Pattern Recognition, 1996, 30(5): 729～742

4 赵锋,赵荣椿. 纹理分割及特征提取方法综述. 中国体视学与图像分析,1998,3(4)

5 Dempster A P, Laird N M, Rubin D B. Maximumlikehood from incomplete data via the EM algorithm. J. Royal Statist. Society (B), 1997,39 (1):1～38

6 Redner R A, Walker H F. Mixture density, maximum likelihood and the EM algorithm. SIAM Review, 1984,26(2): 195～239

7 郭平,卢汉清. 贝叶斯概率图像自动分割研究. 光学学报,2002,22(12)

8 Baraldi A, Parminggian F. An Investigation on the Texture Characteristics Associated with Gray Level Co-occurrence Matrix Statistical Parameters [J]. IEEE Trans. on Geoscience and Remote Sensing, 1995, 32(2): 293～303

9 Haralick R M, Shanmugam K, Dinstein I. Texture features for image classification. IEEE Trans. System Man Cybernet, 1973, 3(6): 610～621

10 张利. 基于图像几何形态与纹理分析的刀具磨损状态监测技术的研究. 浙江工业大学,2002(3)

11 张妙兰,付新文. 一种纹理图像分类方法的研究. 中国图象图形学报,1998,4(8)