

利用硬件抽象机模拟执行技术设计 JAVA 处理器

王海晨, 赵祥模

(长安大学 信息工程学院, 陕西 西安 710064)

摘 要: 提出了一个基于硬件抽象机的流水线微处理器设计框架, 创造性地使用了一种基于标签结构的模拟执行技术. 基于这一框架, 描述了一个堆栈抽象机的工作原理, 实现了一个 Java 指令级并行处理器. 利用堆栈硬件抽象机和堆栈指令折叠技术的组合解决了 Java 处理器中的堆栈依赖瓶颈问题. 软件模拟证明了该处理器能够最大限度地挖掘出 Java 程序中的指令级并行, 并且拥有更高的处理能力.

关键词: 抽象机; Java 微处理器; 堆栈; 指令级并行; 超长指令字

中图分类号: TP368

文献标识码: A

文章编号: 1000-7180(2011)01-0104-04

A Study of Employing Mock Execution Technique in Hardware Abstract Machine to Implement a Java Processor

WANG Hai-chen, ZHAO Xiang-mo

(School of Information Engineering, Chang'an University, Xi'an 710064, China)

Abstract: A framework of designing a pipe-lined microprocessor using hardware abstract machine is proposed in the paper, and creatively used mock execution technique with tags. Based on the framework, the paper presented a stack hardware abstract machine and proposed a Java ILP processor. The software simulation proved that the Java ILP processor enabled to exploit most of instruction level parallelism (ILP) in Java programs and provided the high performance.

Key words: abstract machine; Java processor; Stack; ILP; VLIW

1 引言

抽象机通常用在软件程序编译器中. 它们省略了许多与实际硬件计算机实现有关的细节^[1]. 正如在软件编译器中那样, 软件抽象机经常使用堆栈, 不同的是, 我们提出的硬件抽象机可以应用在堆栈结构, 也可以应用在寄存器结构.

本文提出了一个基于硬件抽象机的处理器设计框架(AMBF, 见图 1). 该框架的创新在于它通过使用带标签的指令执行技术, 用来构造指令级并行微处理器. AMBF 融合了许多现有的计算机体系结构, 可以应用于所有基于流水线的处理器结构设计中. 硬件抽象机通常可以在指令译码阶段实现, 并且依靠带标签的模拟执行技术, 挖掘出程序中的指令

级并行. 在解决了数据冲突后, 带标签的指令被发射到执行引擎, 完成从标签到具体数据的映射, 然后被发射到相应的计算单元去执行. 在框架中, 指令标记单元完成标签模拟执行, 也就是实现具体的硬件抽象机的功能. 指令标记单元只使用标签执行, 不使用具体数据, 并且由于它使用简单的硬件实现, 所以处理速度可以满足指令并行执行的要求.

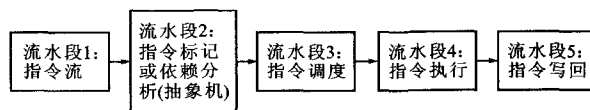


图 1 通用的基于硬件抽象机的流水线框架模型

JAVA 技术广泛应用于网络应用和一些嵌入式设备中. Java 处理器多采用基于堆栈的体系结

构. 由于操作数堆栈的顺序执行造成了 Java 程序执行的瓶颈,限制了指令级并行(ILP). 本文应用我们提出的基于硬件抽象机的处理器设计框架,设计了一个新颖的 Java ILP 处理器. 在处理器中,基于抽象机的模拟执行技术解决了 Java 程序执行的堆栈瓶颈问题,实现了更高的处理器性能.

2 带标签“模拟执行”的抽象机工作原理

在堆栈处理器中,操作数被操作器指令使用后,就会被从执行堆栈中抹去,操作器指令生成的结果可以用一个标签唯一识别. 给 Java 指令“加标签”与 Java 并行注释^[2]近似,都是为了实现并行执行,但是我们的实现是在硬件底层. 本文中的硬件抽象机由指令标记单元(TU)和操作数标签堆栈(OTS)构成. 我们使用下面的 Java 代码片段说明如何标记一个 Java 字节码指令流:

```
iload_1, iload_2, imul, iload_4, iload_5, iadd,
iadd, istore_3
```

表 1 显示了一个 Java 指令标记单元的模拟执行工作原理,以及标签在 OTS 上的改变过程. 每条堆栈指令,当它进入命名单元时,被分配一个标签. 当一条算术逻辑指令在标记后,当它要执行时,作为指令的输入标签要从 OTS 弹出;执行后,生成的结果标签将被压入 OTS,作为后面指令的输入标签. 在例子中,指令 T2 加载后,标签 T2 被压入 OTS;指令 T3 需要 T1、T2 作为输入,将 T1、T2 从 OTS 弹出,或者说消费掉 T1、T2;生成的结果在 T3 中保存,T3 被压入 OTS,等待后续指令把它作为输入变量.

表 1 JAVA 抽象机使用标签工作的实例

字节码指令	指令标签命名单元	操作数标签堆栈
1 iload_1	T1 iload_1	T1
2 iload_2	T2 iload_2	T1 T2
3 imul	T3 imul T1 T2	T3
4 iload_4	T4 iload_4	T3 T4
5 iload_5	T5 iload_5	T3 T4 T5
6 iadd	T6 iadd T4 T5	T3 T6
7 iadd	T7 iadd T3 T6	T7
8 istore_3	T8 istore_3 T7	

我们看到,通过指令的模拟执行过程,“Java 字节码指令对”之间的消费者/生产者关系可以通过标签识别出来. 这一模拟执行过程使用的是标签而不是实际的值,而且“模拟执行”是顺序的. 标记后的指

令就可以被调度执行了. 从结果看,指令模拟执行(标记技术)实现了一个程序的数据流图表达,记录、分析操作数标签的使用情况,就等价于分析指令间的数据相关问题,从而揭示出更多指令级并行. 本文选择了一个超长指令字(VLIW)执行引擎实现 Java 处理器.

3 应用硬件抽象机设计高性能 JAVA 处理器

3.1 处理器架构描述

本文提出的 Java ILP 处理器也采用了流水线结构. 它包含 6 级流水线,分别是取指令,指令标记,堆栈折叠,指令调度及发射,指令执行和写回. 通过指令标记单元对指令标记后,在堆栈折叠逻辑的支持下,把堆栈指令转化为带标签的基于寄存器的指令. 指令标记单元和堆栈折叠单元一起实现硬件抽象机对指令进行“模拟执行”,发现与解决堆栈依赖问题,把可以并行执行的指令成组发射出去. 标签匹配单元(TMU)根据在堆栈高速缓存(寄存器文件)中标签映像的寄存器,来跟踪指令的操作数状态. 当指令的操作数为准备好,则将它加入到准备好指令队列,由硬件逻辑将多条指令捆绑成为 VLIW 指令格式. 这些 VLIW 指令将定序发射到 VLIW 指令执行引擎. 虽然 VLIW 指令是定序发射的,但是在生成时,子指令可以不按照程序的顺序生成. 另外,我们使用 VLIW 执行引擎有利于采用简单的指令控制逻辑. JAVA 处理器逻辑结构的原理框图如图 2 所示.

标签匹配单元记录并更新与每条指令的控制有关的信息. 标签匹配单元使用映射表来管理从标签到寄存器的映射. 标签匹配单元的功能类似一个重排序缓冲区(reorder buffer)^[1],但是进行了扩展. 每当一个结果在寄存器文件中生成,与该结果有关指令的状态信息会被同时更新,并且唤醒在等待生成的结果的指令.

3.2 基于标签的堆栈指令折叠算法

堆栈指令折叠是一种减少堆栈操作数相关的技术,它转换一组字节码指令为一条寄存器指令,节省流水线执行周期^[2]. 我们利用 Java 指令模拟执行技术和 POC^[3](生产者—操作者—消费者)堆栈折叠方案实现了一个基于标签的指令折叠方案. 在该方案中,根据指令在方案中的角色,将 Java 字节码指令划分为生产者,操作者和消费者(P—O—C). 另外,我们增加了一个类型 IP(中间生产者类型),这

类指令表明这一指令会生产出一个中间结果,而其结果将会被它的消费者指令所使用.在 IP 类型的支

持下,我们可以把嵌套字节码指令用一致的方法生成多条加标签的类 RISC 指令.

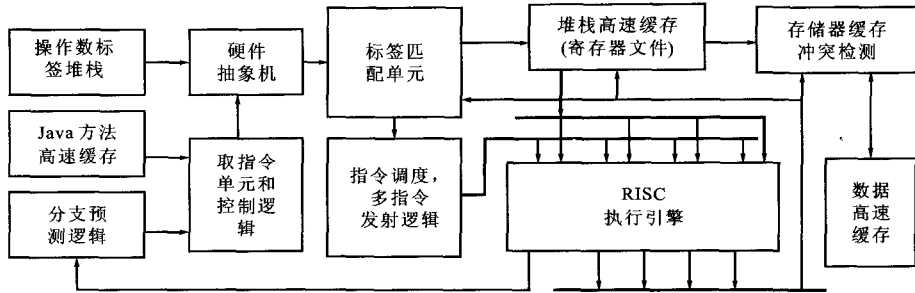


图2 利用Java硬件抽象机实现的Java处理器逻辑结构

指令标记技术可以分析和识别出指令之间的数据相关性,因此,在一定程度上可以简化指令折叠技术的实现,从而使指令标记和堆栈折叠同时运行.依据 POC 指令类型和它们之间的相关信息,堆栈折叠逻辑生成加标签的指令,然后将它们存储在缓存区中,由发射逻辑调度.只有操作者指令可以产生加标签的指令.至于消费者指令,折叠逻辑将检查相关的指令的 POC 类型.如果它是生产者或 IP 类型,仅需做寄存器复制操作;如果是操作者指令,则早期产生的目的标签将由消费者标签替换.表2中用一段Java 字节码说明了基本的指令折叠过程.指令标记过程是顺序执行的,同时也遵循堆栈的工作行为,这样程序执行的正确性可以得到保证.

表2 Java指令折叠的一个实例

Java 字节码	命名单元	操作数标签堆栈	折叠后指令
1 iload a	T1 load a	T1	
2 iload b	T2 load b	T1 T2	
3 imul	T3 mul T1 T2	T3	
4 iload c	T4 load c	T3 T4	i1: imul T1, T2, T3
5 iload d	T5 load d	T3 T4 T5	
6 iadd	T6 add T4 T5	T3 T6	i2: iadd T4, T5, T6
7 iadd	T7 add T3 T6	T7	i3: iadd T3, T6, T8
8 istore	T8 store T7 e		

表2中列出了字节码示例片段的指令折叠结果,新生成的基于标签的指令列在第4列.示例片段包括一组嵌套的指令序列,最后折叠成3条基于标签的指令,i1,i2,和i3.标签T3和标签T6需要被i3重用,在指令i3中,它们被作为中间生产者(IP)折叠类型.在第7行,i3的目的标签(有下画线的)被从标签T7修改为标签T8.从表2我们可以看出,如果没有使用指令折叠技术,需要8个周期顺序执行这段代码;使用指令折叠技术,仅需要3个周期.如果处理器支持多指令发射,则仅需要2个周期.这里

由于折叠指令序列中存在标签数据相关,指令i3必须在i1,i2完成后才可以发射执行.

4 软件模拟和性能评价

我们用踪迹驱动(trace-driven)仿真的方法,执行JVM SPEC98测试程序集,具体实现见参考文献[4].根据Java处理器的硬件体系结构对Java指令进行调度,获得了Java处理器指令并行执行的百分率分布(表3).为了表明我们提出的结构挖掘指令级并行方面的优势,我们首先把它与已有的国外文献进行比较.文献[5]对Pico Java处理器有重要的改进,使用了Cache来存储一些已执行的Java程序,从而提高程序的执行效率,同时采用了定序多发射结构,通过使用堆栈排歧(stack disambiguation)技术提高执行效率.在它给出的结果中,只有少部分三指令组被并行执行,没有四指令并行执行组存在.与之相比,在我们的仿真结果中,三指令并行执行组占总执行指令的比率为1.55%~10%;四指令并行执行组占总执行指令的比率为0.86%~33.43%.这里,并行执行比率的不同与基准测试执行程序的特性有关.文献[6]是一种使用超标量引擎实现的多发射Java处理器机构,与之相比较,我们提出的结构实现了更高的3指令和4指令并行发射(dispatch)百分率.这说明,我们的基于抽象机的Java处理器结构可以挖掘出更多的指令级并行,提高了处理器的指令级并行能力.

5 结束语

本文提出了一个基于硬件抽象机的流水线微处理器设计框架,创造性地使用了一种基于标签结构的模拟执行技术.我们首先描述了一个硬件抽象机结构,利用它,设计了一个Java指令级并行处理器.

该处理器解决了堆栈处理器中严重影响性能的堆栈依赖问题. 通过软件模拟验证,证明了该处理器可以挖掘出更多的存在于 Java 程序中的指令级并行,获得更好的性能.

表 3 使用硬件抽象机方法获得的
Java 处理器指令并行执行的百分率

基准 测试程序	指令并行执行百分率			
	单发射 指令组	两发射 指令组	三发射 指令组	四发射 指令组
compress	67.37	15.43	10.78	6.42
Db	79.97	14.98	3.78	1.27
Jack	79.54	14.22	3.89	2.35
javac	72.85	21.87	4.24	1.04
jess	81.51	13.47	3.26	1.76
mpegaudio	43.26	16.53	6.78	33.43
mtrt	87.92	9.67	1.55	0.86

本文提出的基于硬件抽象机的理论框架可以应用于所有基于流水线的处理器结构设计中. 我们下一步的研究工作,首先探索与国内现有的 RISC 处理器设计相结合,利用文献[7-8]中的 RISC 处理器结构,设计我国自有版权的处理器;接下来,把这一理论框架应用到现代多核处理器的设计中,设计基于硬件抽象机理论的硬件多核高性能处理器.

参考文献:

[1] 朱红, 张赛男. Java 并行注释规范 JAC 的扩展 [J]. 微电子学与计算机, 2009, 26 (8): 42—44.

[2] Wang H C, Yuen C K. Exploiting dataflow to extract Java instruction level parallelism on a tag-based multi-issue semi in-order (TMSI) processor [C]// IEEE International Parallel & Distributed Processing Symposium 2006 (IPDPS2006). Rhodes: IEEE, 2006.

[3] Radhakrishnan R, Talla D, John L K. Allowing for ILP in an embedded Java processor [C]// Proceedings of the 27th International Symposium on Computer Architecture. Vancouver: IEEE, 2000: 294—305.

[4] Achutharaman R, Govindarajan R, Hariprakash, et. al. Exploiting Java—ILP on a Simultaneous Multi—Trace Instruction Issue (SMTI) Processor [C]// International Parallel and Distributed Processing Symposium. Nice: IEEE, 2003: 76—84.

[5] 张伟功, 段青亚, 刘曙蓉, 等. 一种适于 16 位 RISC 处理器的伪四级流水结构研究 [J]. 微电子学与计算机, 2008, 25 (1): 73—75.

[6] 张英武, 袁国顺. 32 位嵌入式 RISC 处理器的设计与实现 [J]. 微电子学与计算机, 2008, 25 (6): 14—17.

[7] 张伟功, 段青亚, 刘曙蓉, 等. 一种适于 16 位 RISC 处理器的伪四级流水结构研究 [J]. 微电子学与计算机, 2008, 25 (1): 73—75.

[8] 张英武, 袁国顺. 32 位嵌入式 RISC 处理器的设计与实现 [J]. 微电子学与计算机, 2008, 25 (6): 14—17.

作者简介:

王海晨 男, (1969—), 博士, 副教授. 研究方向为高性能处理器结构、并行与分布式计算.

赵样模 男, (1968—), 博士, 教授, 博士生导师. 研究方向为分布式测控系统.