用于多目标编译系统构造的目标机体系结构描述*)

杨 萍[1] 王生原[2]

(北京语言大学信息学院 北京 100083)[1] (清华大学计算机系 北京 100084)[2]

摘 要 本文是关于多目标编译系统构造的目标机体系结构描述的一个综述。主要涉及的话题:机器描述应当描述什么和描述语言的设计原则,现行的体系结构描述语言的状况及分类,一个典型的机器描述示例,结构信息描述,以及机器描述所面临的挑战及机遇等。

关键词 体系结构描述(语言),编译系统,可重定向

Architecture Descriptions for the Building of Multi-target Compiling Systems

YANG Ping[1] WANG Sheng-Yuan[2]

(School of Information, Beijing Language and Culture University, Beijing 100083)[1]

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084)[2]

Abstract A survey on the architecture descriptions used in the building of multi-target compiling systems is presented. It is concerned to several topics, such as what should be described, the design principle of architecture description languages (ADLs), the current status and classification of ADLs, a typical ADL example, the structural information, the challenge and opportunity, and etc.

Keywords Architecture description (languages), Compiling systems, Retargetability

1 引言

可重定向(retargetability)是多数机器级软件工具追求的目标之一。其特点是需要使机器相关的部分可以适应多样的、可变的目标机体系结构。如果采用合适的目标机信息描述,使得及其相关的部分可以自动生成,而不是通过手工修改,那么机器级软件工具的重定向就会容易得多。

现阶段,多数支持多目标体系结构的可重定向编译系统或基础设施(Infrastructure),如 Free Software Foundation 的 GCC(GNU Compiler Collection)[1],美国 NCI(National Compiler Infrastructure)项目 SUIF[2]和 Zephyr[3],以及 HP、NYU 和 UIUC 的联合项目 Trimaran[4]等,大都采用了目标机信息描述,即机器描述。编译器后端受到机器描述的驱动,产生特定目标机的代码。机器描述与编译主体分离,可以使编译器很方便地重定向到新的体系结构,或者称此过程为代码生成器的自动产生(参见文[10]第 6 章)。这里,重定向的含义包含两个方面:一是适应性,即所产生的代码能够适应目标机的运行环境,这是最基本的要求;另一方面是有效性,即能够充分利用目标机体系结构的特点产生优化的代码。

用于机器描述的体系结构描述语言(ADLs,或称机器描述语言)目前并无标准可循,在语言的简洁性,目标机体系结构的复杂性、多样性以及代码优化目标之间有较大的权衡空间。这为基于机器描述的编译器后端构造技术提供了许多挑战,同时也是机遇。本文正是出于这样的目的,通过对多目标编译系统的构建中,机器描述应当描述什么和描述语言的设计原则,现行的体系结构描述语言的状况及分类,典型机器描述示例,结构信息描述,以及机器描述所面临的挑战及机遇等方面进行了探讨,以作为与国内同行进行交流的基础。

2 描述什么和描述语言的设计原则

谈到机器描述语言,不少人自然会联系到硬件描述语言,如 VHDL 和 Verilog。但它们并不适合多数系统软件的构造,也包括不适合用于可重定向编译系统的目标机器描述。使用 VHDL 或 Verilog 的不利因素有:它们的抽象级别太低,含有大量琐碎的对于系统软件的构造无用的细节;它们侧重于用来描述行为(即,如何做),而不是用于描述语义(即,做什么),然而想要从实现级的信息自动获取体系结构信息是有一定难度的;此外,用硬件描述语言来描述同一个机器也会在表示级别和风格上有很大的不同,这会使问题进一步复杂化[8]。

实际上,多数可重定向编译系统的机器描述以 ISA 信息为主体,包含某些抽象的结构信息,如 GCC 的 md 文件中包含的 define_delay,define_function_unit 等;有的机器描述包含较丰富的结构信息,如 Trimaran 的机器描述语言 MDES[5]的 Resource_Unit、Reservation_Table、Operand_Latency、Operation_Latency 及 Scheduling_Alternative 等信息。

我们认为,机器描述语言应当描述什么,取决于:

(1)需要支持的系统级工具的类型。如编译器,模拟器,汇编器,综合器及其组合等。

(2)资源优化的粒度,即所要考虑哪些功能部件的使用。如 ALU,IR,流水线,分发槽,延迟槽,缓存层次等。

(3)所考虑体系结构的层次。如互联拓扑结构,通信链路,ISA(指令集体系结构),指令、时钟、或相位周期,微体系结构等不同的层次。

(4)应用程序的范型。如并发,实时,函数式,多媒体等不同的范型。

从编译器构造的角度,所需描述的目标机信息应当包括:

*)本文受到国家自然科学基金项目(#60403022)资助。杨 萍 副教授。

用于目标代码生成和优化的指令集，其文法和语义；执行的上下文，包括资源占用，延迟，调度等信息；有时还需要描述 OS 环境以及宏体系结构（如多处理器或多机结构）信息等方面。其中，一些较多用到的目标机信息列举如下：

(1)指令格式（操作数类型，格式，操作码）。

(2)存储种类及机器编码形式（大小端）。

(3)寄存器信息（包括专用寄存器）。

(4)调用约定（函数入口、出口）。

(5)ISA 约束（指令级并行，操作数及操作之间）。

(6)资源占用信息（如预约表）。

(7)延迟等待时间（操作数延迟，操作延迟）。

(8)调度选择（满足上述条件下规定优先级）。

(9)代码映射（通过表，模式匹配或其它规则）。

(10)各种结构信息（如流水线结构，资源约束、冲突等）。

至于描述语言，文[9]中谈到了在设计 Zephyr 编译系统[3]的机器描述语言时所遵循的目标及原则（N. Ramsey, Harvard Univ.; J. W. Davidson, Univ. of Virginia; M. F. Fernandez, AT&T）：

(1)可重用性（Reusable descriptions）。同样的机器描述应当可以用于多种、甚至不同类型的软件工具。

(2)含义明确的局部描述（Meaningful partial descriptions）。描述框架应当能够仅描述面向特定任务的信息。

(3)描述应当是可信赖的（Reliable descriptions）。描述框架应当提供这样的机制，使书写者能够对自己所描写的以及自己想要表达的意图充满自信。

(4)语言的可用性好，描述的可读性好（Usable languages, readable descriptions）。领域专家应当能够读懂和理解已有的描述，并可以使用该描述框架创建新的描述。

(5)描述具有简洁性（Concise descriptions）。

这些准则虽然具有一定的共性，但应当看到，系统工具的特征及开发团队的出发点差别很大，对体系结构描述的需求是多样的，因而机器描述语言的设计目标和原则也会有所不同或有所侧重。

MESCAL 体系结构描述语言（MADL）[8]面向嵌入式系统开发，拟同时服务于编译器和模拟器等软件。文[8]中强调了体系结构建模（即描述）应具有的 4 个特性（目标）：

(1)紧凑性（Compactness）。为了减小开发的工作量，体系结构模型应当足够紧凑，使得设计者可以有效地捕捉其精髓。

(2)灵活性（Flexibility）。大的设计空间意味着多的优化空间，支持更广的体系结构范围将会导致更好的设计。此外，体系结构模型应当足够通用，以如实反映目标体系结构的重要特征，最小化评估过程中的失误。

(3)高效性（Efficiency）。体系结构模型应当具有支持高效模拟的潜力，以减少设计的时间。

(4)可分析性（Analyzability）。编译器设计需要对高级体系结构的特性进行获取和分析。

至于机器描述同系统工具设计者以及编译主体的接口，目前多数编译系统都备有相应的实用程序。这些实用程序如编辑程序，语法预处理程序，各类体系结构信息的自动提取程序等。

3 描述语言的现状与分类

总的来说，现行的体系结构描述语言并未遵循任何标准，呈多样性的特征。造成这种状况的原因有许多。多数情况下，这些语言仅适合于各自的领域，如仅服务于编译器或仅服务于模拟器。描述语言的设计还要考虑在语言简单性和所能描述的目标机范围大小之间进行权衡。一般来说，粗略的描述容易，而完整的描述较为困难。另外，一些难以预料的新的体系结构特征，不规则的约束等也是造成体系结构描述语言复杂多样的重要因素。

本节仅从服务于编译器的角度对体系结构描述语言的分类。

有些描述语言，如 GCC[1]的机器描述语言，它们不仅取决于所描述的目标机，而且还取决于特定的中间描述语言，因此其重用性不是特别好，仅支持单一种类的工具（如 GCC 的机器描述语言仅服务于编译器的构造）；另外一些描述语言较为独立，它们的可重用性较好，如 LISA[6]，CSDL（Zephyr）[3]等。

有些描述语言除了服务于编译器外，还同时服务于其它类型的系统工具。如在 Trimaran 编译系统中，机器描述语言 HMDES2[7]服务于编译器后端 Elcor，同时也服务于参数化 ILP 体系结构 HPL-PD 的模拟器。

有些编译系统的描述语言只有一个，如 Trimaran 编译系统只有一个体系结构描述语言 MDES。而有些编译系统的描述语言有多个，如 Zephyr 编译系统的体系结构描述语言 CSDL 由 SLED，λ-RTL，CCL（调用约定语言）以及 PLUNGE（流水线描述语言）等多个语言的组合。

如果依所描述体系结构的级别大致可将描述语言分为结构级，行为级和混合级[8]。结构级侧重于描述各功能部件的结构及其互联关系；行为级以指令集体系统结构的描述为主，如 nML[22]，ISDL[12]等；混合级则是结构级和行为级的组合，如 HMDES[7]，LISA[6]，MADL[8]等。对于服务于编译系统的体系结构描述语言来说，至少应该包含行为级的描述，多数都不同程度地包含结构信息的描述。

4 示例-GCC 的机器描述

本节以自由软件联盟的 GCC（GNU Compiler Collection）[1]为例，说明其目标机器描述所涉及的对象，描述方法及其与编译器主体的接口，以使读者对典型多目标编译系统的机器描述有一个较为直观的整体认识。

GCC 的机器描述由两部分组成：一部分是 C 头文件 machine. h 和 C 源文件 machine. c，另一部分是描述目标机指令的机器描述文件 machine. md。这里 machine 泛指处理机的芯片名，如针对 MIPS 体系结构，则相应文件应为 mips. h，mips. c 和 mips. md。

描述文件 machine. md 是一个文本文件，主要包含指令模式，ISA 属性，定义目标机相关的优化等信息；C 宏定义头文件 machine. h 主要描述目标机各种参数，而 machine. c 主要用于定义目标机相关的函数与数据。

指令模式由模式名，RTL 模板，模式匹配测试条件，汇编代码输出模板，以及含有匹配该模式属性值的向量（可选）组成。指令模式通常都有一个模式名。有时模式名也可以是空串代表一种无名模式，但只用于之后的优化遍，而不用来生成 RTL 代码。RTL 模板用来刻画指令的样式。这里，RTL（Register Transfer Language）是 GCC 的低层中间表示结构。

ISA 属性包括指令属性的描述（长度，类型，等），所需延迟槽的定义，以及功能部件相关的指令调度信息。

一条指令需要延迟槽是指物理上位于该指令之后的一些指令的执行好像位于该指令之前被执行一样。machine.md 中用 define_delay 表达式来说明延迟槽，其格式为：

```
(define_delay test
    [delay-1 annul-true-1 annul-false-1
     delay-2 annul-true-2 annul-false-2
     ……
     delay-n annul-true-n annul-false-n
    ]
)
```

其中 test 以及 delay-i(i=1…n)是指令属性测试条件，annul-true-i，annul-false-i(i=1…n)。只有符合 test 测试条件的指令才按照此表达式进行调度，如果符合条件的话，n 为需要的延迟槽数目，放置在第 i 个延迟槽的指令满足属性测试 delay-n。annul_true_i 是属性测试，它指明如果分支条件为真时在延迟槽的哪些指令将被作废，类似地，annul_false_n 指明当分支条件为假时在延迟槽的哪些指令将被作废。

machine.md 中用 define_function_unit 表达式来说明由某一类指令使用的功能部件的用法。其形式为：

```
(define_function_unit name multiplicity simultaneity test ready-delay issue-delay [conflict-list]
)
```

其中，*name* 表示功能部件的名称；*multiplicity* 表示相同功能部件的数目；*simultaneity* 表示能同时执行的最大指令条数，零表示该部件是流水部件且对指令条数无限制；*test* 是属性测试；*ready-delay* 表示该指令的结果准备好所需要的时钟周期数；*issue-delay* 表示后继指令可以开始使用此部件所需的时钟周期数；*conflict-list* 是由条件表达式构成的表，如果后续指令满足其中的条件，那么冲突延迟是 *issue-delay* 指定的值，否则冲突延迟为零。

machine.md 中还可以定义目标机相关的优化信息。例如在确信可以改进指令或延迟槽调度时，可以通过 define_split 来定义相关指令的分解。而对于机器相关的窥孔优化，则可以通过 define_peephole 来定义。关于 define_split 和 define_peephole 的格式，这里不去赘述。

machine.h 主要用于给出目标机的参数，如存储布局(定义数据的大小及排列)，寄存器的命名及硬件寄存器的描述，硬件寄存器的类别，堆栈及调用方式，寻址方式，不同操作的有关代价，指令调度器行为的调整信息，存储器分段等信息。

最后，其它一些与目标机相关的数据和函数可以在 machine.c 中加以补充。

GCC 编译器通过十几个函数来获取源于机器描述的信息，主要用于 RTL 指令的生成，RTL 指令的各优化遍(指令调度，寄存器选择，局部及全局寄存器分配，延迟分支调度，机器相关的窥孔优化等)，转换 RTL 到汇编代码等方面。在 GCC 的各优化遍中，使用到机器描述的 C 源码文件包括 sched.c，regclass.c，local-alloc.c，global.c，reload.c，reload1.c，reorg.c，final.c。而当机器描述文件改变时，这些文件不需要任何变化，可见机器描述在可重定向编译系统中所起的作用。

有经验的 GNU 使用者可以通过修改 machine.h，machine.c 和 machine.md 的内容将 GCC 重定向到新的处理器。但是，欲成为一名有经验的使用者需要有一定的时间代价，也并非朝夕之事。

5 结构信息的描述

在结构描述方面，GCC 机器描述语言的能力是十分有限的(通过 define_function_unit，define_delay 定义结构相关的信息)，仅能描述结构规则的体系结构，难以刻画不规则的结构信息。而有些机器描述语言，则是根本不支持结构信息的描述，如 ISDL[12]。ISDL 仅通过列出所有不合法操作的组合来体现结构信息，对于较复杂的处理器资源冲突，这样的组合是很庞大的。

在结构信息描述方面，有许多值得关注的工作。

每个操作的详细冲突信息是编译器所需要的体系结构描述的一个重要方面，常见的方法是采用预约表(Reservation Tables)，它是一个位向量矩阵，大小为资源的数目乘以最长流水线的长度。预约表用于表示每个操作所用到的流水线和数据通路资源。通过比较预约表可以检测操作之间的冲突。采用这种方法的体系结构描述如 Trimaran[4] 的 MDES[5]。目前，针对预约表的研究主要包括紧凑表示(为提高冲突检测效率)，如与/或树表示[5]和位向量表示，以及预约表的自动生成[11]等。

有限状态自动机(Finite State Automaton)可以用来表示一个处理机的所有合法操作调度集合，从而可以检测流水线资源的竞争条件[13,14]。与采用预约表的方法相比，此类方法在结构冒险检测方面有更高的效率，对于给定处理器仅需构建优先自动机一次。另外一个优点就是可以适用于全局指令调度。但这种方法不太适合于资源数目较大的情形。同时，这种方法并不利于某些先进的调度技术[15,16]的采用。

Petri 网[17]非常适合于多线程、并发及资源流动系统的建模，并且具有很强的模拟及形式验证能力，因此近年来也被人们尝试用于体系结构的描述[18,19]。适合于体系结构的描述的 Petri 网模型多采用有色时间网(Timed Colored Petri Net)，各种类型的指令、数据可以看作为功能部件间流动的资源，采用有色网(Colored Petri Nets)可以支持资源类型的定义，而采用时间网(Timed Petri Nets)描述功能部件、指令、数据及控制信息之间的各种时序关系。但在多目标编译器的实际构造方面，目前此类方法的研究还远远不够，有待进一步发掘其潜力。

MADL[8]中的 OSM 模型采用一种扩展的有限状态机来描述操作的动态行为。这种扩展的有限状态机拥有可变的内部状态，用以保存操作的中间计算结果，同时可以压缩状态图大小并屏蔽一些与外部无关的细节。OSM 的每一转移边都关联一个条件，代表结构、数据等资源的可用性。一个 OSM 与外部环境之间的通信是借助于所谓的托肯管理器(Token Manager)。该模型某种程度上与 Petri 网模型有相似之处，但二者是有差别的，因而不能直接采用丰富的 Petri 网分析和模拟工具。

还有其它各种不同风格方法，如基于抽象状态机(Abstract State Machine)ASM 的方法[23]，基于项重写系统(Term Rewriting Systems)理论的一种以操作为中心的体系结构描述方法[20]，以及基于函数式语言的方法[21]等。

6 面临的挑战与机遇

设计一个好的体系结构描述语言，需要在可重用性、灵活性、可信赖性、可分析性、简洁性、紧凑性、高效性等多方面的尺度上进行权衡，是一项十分复杂和具有挑战的工作。至今还没有任何已有的体系结构描述语言能够在所有这些方面做得都十分出色。

现行的方法明显存在以下两方面的不足：

(1) 目标机的结构特征和运行时信息没有被很好地利用。

(2) 对应机器描述和优化目标的变化,编译后端不能很灵活地重构。

从前面一节的叙述可知,在体系结构的结构信息描述方面,人们采用了多种可能的、又是不同风格的方法。然而,在体系结构描述中,风格差异最大,需要权衡的因素最复杂的部分就是结构信息的描述。这一点对研究人员来说,既是挑战,又是机遇。

总的来说,在可执行规范基础上建立机器描述将会成为新的趋势。这确实是一个值得探讨的问题,因为在充分结构关系,并发,冲突等信息的刻画方面,可执行的机器描述要比静态的文本描述有许多优势;并且可执行规范通常有一定的问题求解能力,这对于代码优化是有帮助的。

有了合适的可执行机器描述,可以充分地提取所关心的运行时动态信息。然而,还有一个重要问题必须面对,即优化目标的重定向问题。一方面,不同的优化目标关心不同的运行时信息,如执行效率,代码量,低功耗以及它们的组合等都是不同的优化目标;另一方面,如何方便地实现优化功能的重组,以适应优化目标和运行时信息的变化。解决优化目标的重定向问题,似乎可以遵循 Aspect-Oriented Programming (AOP)的方法学[24]。不同的优化目标即为不同的 Aspects。

如何在技术上支持 AOP 也是一个专门的研究话题,我们认为基于反射(Reflective)机制的方法[25]是相对现实的和较有前途的。基于反射机制的系统构造,能够很自然地分开考虑功能部分和非功能部分。若采用具有一定反射能力体系结构描述语言,将会方便体系结构信息和优化目标的重定向。

结束语 现行的体系结构描述语言无任何标准,呈多样性的特征。多目标编译系统大都带有各自的体系结构描述语言,但多数之间都有着明显的差异,没有哪一种描述语言能够完全符合理想的设计原则,特别是在结构信息描述方面尤为明显。尽管如此,体系结构描述是多目标编译系统设计的核心,这是毫无疑问的。这是一个既有挑战又是机遇的研究课题。

参 考 文 献

1 GCC online documents. available at http://www.gnu.org/

2 Wilson R P, et al. SUIF: An Infrastructure for Research on Parallelizing and Optimizing Compilers, ACM SIGPLAN Notices, 1994, 29(10): 31～37

3 Zephyr online documents. available at http://www.cs.virginia.edu/zephyr/

4 Trimaran online documents. available at http://www.trimaran.org/

5 Gyllenhaal J C, Hwu W W, Rau B R. Optimizati on of machine descriptions for efficient use. IJPP, 1998,26(4):417～447

6 Zivojnovic V, et al. LISA—Machine description language and generic machine model for HW/SW co-design, In: Proc. IEEE VLSI Signal Processing Workshop, 1996. 127～136

7 Trimaran Release. http://www.trimaran.org. The MDES User Manual, 1997

8 Qin Wei. Modeling and Description of Embedded Processors for the Development of Software Tools, Ph. D dissertation, Princeton University, 2004

9 Ramsey N, Davidson J, Fernández M. Design Principles for Machine-Description Languages. Submitted to ACM Transactions on Programming Languages and Systems, 2001

10 Muchnick S S. 高级编译器设计与实现. 北京:机械工业出版社影印,2003

11 Cornea R, Halambi A, Grun P, Dutt N, Nicolau A. Compiler-Architecture Exploration using Reservation Tables Generation, Technical Report #99-31, Department of Information and Computer Science, University of California, 1999

12 Hadjiyiannis G, et al. ISDL: An instruction set description language for retargetability. In: Proc. of the 1997 Design Automation Conference, June 1997

13 Proebsting T A, Fraser C W. Detecting Pipeline Structural Hazards Quickly. In: Proc. of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages, Portland, Oregon, United States, 1994. 280～286

14 Bala V, Rubin N. Efficient instruction scheduling using finite state automata. In: Proc. of the 28th annual international symposium on Microar chitecture. Ann Arbor, Michigan, United States, Year of Publication, 1995. 46～56

15 Novack S, Nicolau A. Mutation Scheduling: A Unified Approach to Compiling for Fine-Grain Parallelism. Lecture Notes in Computer Science, 1995,892:16～30

16 Rau B R. Iterative modulo scheduling: An algorithm for software pipelining loops. In: Proc. of the 27th annual international symposium on Microarchitecture, 1994

17 袁崇义. Petri 网原理. 北京:电子工业出版社,1997

18 Burns F, Koelmans A, Yakovlev A. Modelling of superscalar Processor architectures with design/CPN. In: Proc. of Workshop on Practical Use of Coloured Petri Nets and Design, June 1998

19 Zuberek W M, Govindarajan R, Suciu F. Timed Colored Petri Net models of distributed memory multithreaded processors. In: Proc. of Workshop on Practical Use of Coloured Petri Nets and Design, June 1998

20 Hoe J C, Arvind. Synthesis of operation-centric hardware escriptions. In: Proc. of Intl. Conf. on Computer-aided Design, 2000. 511～519

21 Matthews J, Cook B, Launchbury J. Microprocessor specification in Hawk. In: Proc. of the Intl. Conf. on Computer Languages, 1998. 90～101

22 Fauth A, Praet J V, Freericks M. Describing instructions set processors using nML. In: Proc. of Conf. on Design Automation and Test in Europe, Paris, France, 1995. 503～507

23 Teich J, Weper R, Fischer D, Trinkert S. A joined architecture/compiler environment for ASIPs. In: Proc. of Intl. Conf. on Compilers, Architectures and Synthesis for Embedded Systems, San Jose, CA, Nov. 2000. 26～33

24 Kiczales G, Lamping J, Mendhekar A, Maeda C, Lopes C, Loingtier J-M, Irwin J. Aspect-oriented programming. In ECOOP'97? Object-Oriented Programming, 11th European Conference, Lecture Notes in Computer Science. 1997,1241:220～242

25 Maes P, Concepts and Experiments in Computational Reflection, In: Proc. Conf. on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'87) (ACM SIGPLAN Notices, 22,10, 1987),1987. 147～155