多目标交叉编译技术*)
——GCC与Zephyr编译器构造的分析与比较

Cross Complation Techniques for the Support of Multiple Objects
——Comparison and Analysis of GCC and Zephyr Infrastructure

戴桂兰 张素琴 田金兰 蒋维杜 戴 军
(清华大学计算机科学与技术系 北京100084)

Abstract GCC and Zephyr are the two typical common compiler platforms. The paper presents the basic questions that need to be resolved for the support of the development of multiple languages and multiple objects compilers, analyses and compares their architectures and key techniques, summarizes their merits and shortcomings respectively, and probes preliminarily into the further research trend of cross compilation.

Keywords Compiler, Cross compilation, Intermediate representation, Machine description, GCC, Zephyr infrastructure

1 引言

编译系统是任何计算机系统中不可缺少的重要部分。编译系统的研制因其技术复杂、难度较高而需要投入较多的人力、物力和花费较长的研制周期。过去编译器均是针对某一特定语言和目标机而编写的。随着计算机的飞速发展，过去那种编写编译器的方法已很难满足需要。在八十年代初，针对各种程序设计语言的后端具有较大共性的特点，国外相继开发了支持多种语言的编译系统，并很快成为各计算机公司编译系统采用的通用方法。在这种编译系统中，每种语言具有独立的前端，所有语言共享一个后端。这大大降低了开发编译系统的开销。这种支持多语言的编译系统虽然满足了快速开发编译器的需要，但对计算机硬件不断更新换代的飞速发展仍显得力不从心。虽在七十年代末、八十年代初有一些人致力于支持多目标机编译系统的研究，但由于受到各种条件的限制，仅停滞于理论研究和实验阶段。进入八十年代后期，随着国外各种软件公司的兴起以及计算机芯片不断推陈出新，对支持多目标机的编译系统的研制显得越来越重要，编译程序的开发者深刻地意识到：只有支持多语言、多目标机的编译系统才具有较强的生命力和竞争市场[10]。

另外，为便于开展对计算机体系结构、编程语言和编程环境的实验研究，必须能够快速地产生高质量的编译器，而编译器的构造是一个劳动密集型的工作，经常是一个瓶颈。虽然对于编译前端自动生成工具的研究已比较成熟，并开发了一些普遍使用的工具，如：LEX、YACC等，但对编译后端的自动生成的研究仍处于探讨阶段。FSF(Free Software Foundation)[4]和美国相继启动了GNU系统和NCI(National Compiler Infrastructure)工程[2]，其部分研究目标是为聚集全世界的研究力量，协作发展编译技术，促进技术产业化。

为便于吸取现有多源语言多目标编译技术中具有优势的思想，为交叉编译技术的进一步发展提供理论基础，同时为编译器的开发者选择理想的编译器开发平台提供指导，本文首先对GCC和Zephyr这两个具有代表性的公共编译基础设施做一简单介绍，进而探讨支持多源语言多目标编译系统所要解决的根本问题，并从总体结构及其所采用的关键技术对GCC和Zephyr进行分析比较，总结出它们各自的优缺点。

2 GCC与Zephyr编译基础设施概况

GCC(GNU Compiler Collection)和Zephyr是支持多语言、多目标机中最有代表性的两个编译基础设施。GCC是FSF启动的GNU工程的一部分[4,5]，其开发目标在于提高GNU系统中编译器的质量。GCC目前已支持的源语言有C、C++、Objective-C、FORTRAN、Ada，已移植的平台有一百多种，涉及三十多种处理器、六十多种系统[5]。由于GCC属于自由软件，在全世界范围内得到广泛使用。

Zephyr框架由弗吉尼亚大学和普林斯顿大学联合开发，是美国启动的NCI工程的主要部分之一[2]。Zephyr框架是围绕着VPO(Very Portable Optimizer)而建立的编译基础设施，支持机器指令级的低级优化[3]。Zephyr框架的重点不在于建立一个编译器，而在于提供建立多个不同编译器的策略。目前已利用其开发了支持Java、C++等多种源语言，Alpha、Mips、Motola 88100等二十多种处理器的编译器。

3 总体结构

GCC和Zephyr通过使编译前端、编译后端和机器描述相对分离又相互配合，从而实现对多语言多目标机的支持。

GCC编译程序由与源语言相关的前端、与源语言无关的后端和目标机描述三部分组成，如图1所示。对于其支持的每种源语言，存在一个独立的、与语言相关的语法分析器。利用这些语法分析器，对不同的源语言产生相同的分析结果——语法树。与源语言无关的后端是GCC编译器构造器的主体部分，它将语法树转换成称之为RTL的中间代码，然后对中间代码进行各种优化并最终生成汇编代码输入文件；机器描述由目标机描述宏文件和机器描述文件两部分组成，它们分别描述目标机操作系统和目标机体系结构。

如图2所示，Zephyr编译程序由与源语言相关的前端、代码扩充器、VPO和机器描述四部分组成。前端输出由编译器书写者选择的中间表示，然后利用代码扩展器，结合与目标机有关的信息(机器描述)，将中间代码转换为满足机器不变性的RTL表示作为VPO的输入文件，然后由VPO产生高质

*)国家自然科学基金(名称：嵌入式软件开发环境研究，编号：60083004)。戴桂兰 博士后。

量的汇编代码[3]。

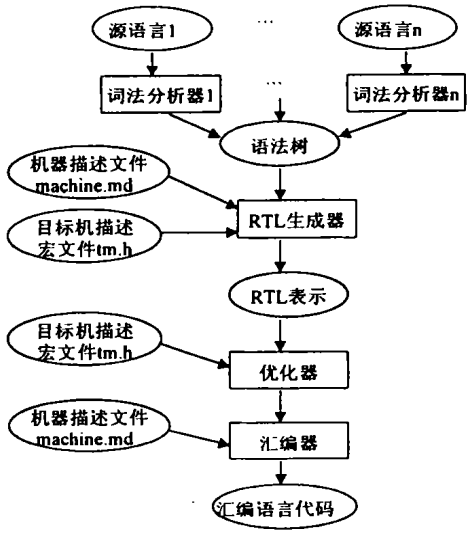

图1 GNU CC 编译系统的总体结构

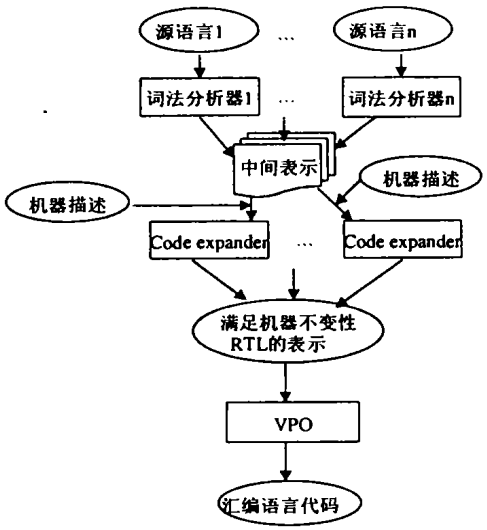

图2 Zephyr 编译系统的总体结构

从上述分析可以看出，GCC 与 Zephyr 的总体结构的主要不同在于：

(1)在 GCC 编译器构造器中，由前端产生的语法树没有保留下来，使之各成份间的界限不如 Zephyr 明显，从而给组装一个完整的编译器带来一定的困难；

(2)在 GCC 编译器构造器中，机器信息在不同编译阶段进入编译系统中，而在 Zephyr 中，机器信息是通过代码扩充器一次性地进入编译主体中。前者使得编译主体与机器描述的耦合度较高，从而降低了其可复用性，但这种形式有助于提高变换以及优化的效率。

4 多目标交叉编译的关键技术

如所知，由于不同的源语言间、不同的处理器之间和不同的操作系统之间存在着相当大的差异，因此对于支持多语言多目标的编译系统而言，它需要解决三个根本问题：(1)提供一种对目标机进行恰当描述的机制，既能描述系统和硬件的共性，又能描述它们各自的个性，同时还要适合于编译程序处理；(2)提供一种较好的中间表示，应在适当的抽象层次上，向上能支持多语言的映射，向下能适应多平台转换且宜于进行各种优化；(3)提供机器描述和编译主体间统一的接口。GCC 和 Zephyr 编译基础设施分别提供了解决这三个问题的编译技术，从而为快速实现多语言开发和多目标平台移植提供了有力的支持。

4.1 中间表示

GCC 和 Zephyr 都采用了双层中间表示，这有助于提高编译器成份的可移植性。GCC 的第一层中间表示为前端产生的语法树，另一层为与 LISP 语言的表达式结构相类似的 RTL(Register Transfer Language)。语法树是与源语言相关的前端和与语言无关后端之间的接口。在词法分析过程中，一旦一个函数被扫描分析，则以语句为单位生成对应的 RTL 中间代码。实际上，首先将读入的每条语句转换成语法树的形式，然后转换为 RTL。RTL 的基本元素是 rtx 表达式。每个 rtx 都具有统一的内部数据结构与外部语法形式。rtx 表达式的外部语法一般形式为：

(code: m opn1 opn2)

其中，code 为 rtx 操作码，m 为机器方式，opn 为操作数。根据其允许出现的场合不同，可将 rtx 分为表示指令的 rtx、具有副作用的 rtx、表示运算的 rtx 和表示初等值的 rtx。这四种 rtx 处在不同的位置上构成了 rtx 的层次结构。处在最上层的是表示指令的 rtx，其前三个操作数均为整常数，分别表示本条指令编号、前一条指令编号和后一条指令编号，它们相互拉成一条前向链和后向链。GCC 的 RTL 代码则由这样一条一条的指令组成，每条 RTL 指令都实际上表示着目标机的一条指令。

Zephyr 编译构造器对其高级中间表示没有明显的限制，它可为前端输出的任意形式，因此，编译器的书写者可以自由选择中间表示。低级中间表示是具有强类型树形结构的 RTL (Register Transfer List)，其语义与任何机器无关。由于 RTL 的设计目标是便于机器处理而不是人读写，其形式简单、包含有尽可能多的信息且无二义性[1]，从而使其不仅适用于机器规格说明，而且可用在编译器、二制转换器和其它工具的实现中。

一个 RTL 为一系列寄存器变换所产生的效果，每个变换效果表示将一个值变换到存储空间中，也即存储操作。只有当哨兵表达式为真时，变换才发生，且一个列表中的变换效果是同时发生的。RTL 部分语法用 BNF 可表示为：

```
rtl=RTL(guarded * )
guarded=GUARD(exp,effect)
effect=STORE(location dst,exp src,ty)
       |KILL(location)
cell= CELL(space,exp)
location=AGG (aggregation,cell)
exp =CONST(const)
    |FETCH (location,ty)
    |APP (operation,exp * )
Ty=(int)-size of a value,in bits
```

分析比较：(1)Zephyr 缺少对高级中级表示形式的限制，这虽提高了编译前端的可复用性，但增加了后端生成的复杂性；(2)Zephyr 的 RTL 的语义与机器无关，包含的信息多，虽便于机器处理，但不便于人理解；(3)Zephyr 的 RTL 是强类型的，支持静态检查，可较早地检查出 RTL 中间表示中错误或不一致的描述。

4.2 机器描述

为简化编译器和其它机器级工具的建构，机器描述已成为一个关键技术。在 GCC 编译系统中，机器描述由宏定义头文件 machine.h 和描述文件 machine.md 二部分组成。其中，machine.h 定义了各种有关目标机的参数，如指导编译程序

的宏定义、存储器的编址信息、有关汇编语言输出的宏定义等等。这些参数的引用散布在编译程序的各个部分。编译程序通过其中定义的宏名决定应当进行的各种处理。利用这种方式，GCC 编译程序虽然含有与这些参数相关的代码，但却不含有与具体机器相关的代码。machine. md 是描述目标机指令的正文文件，其中除了允许以“;”打头的注释行外，其余的均是采用 RTL 外部语法形式书写的 rtx 表达式。这些表达式最外层的操作码是专门用于机器描述的，由它们组成的机器描述包含了目标指令集的各种内容。这些内容主要包括：(1)指令样板，描述目标机所支持的每一条指令以及该指令对应的 RTL 指令形式与汇编代码的输出格式；(2)指令样板的补充，指出可以进行与目标机相关的优化动作及相应的 RTL 指令形式；(3)指令的有关特性，包括指令的分类、指令中允许的数据类型、指令的长度、指令的延迟槽口和功能部件。每条指令的内容都封装在一条 define_insn 表达式中。define_insn 的一般形式为：

(define_insn s e s s E)

其中，第一个操作数 s 为指令名，可为空；第二个操作数 e 为指令模板，它为一个不完全的 RTL 表达式或向量；第三个操作数 s 表示一个指令体是否与该指令样板相匹配的测试条件；第四个操作数 s 指定当前匹配的这条指令的汇编代码输出格式；最后一个操作数 E 为一 RTL 向量，给出指令的属性值。GCC 提供了114条标准指令，在机器描述过程中，有一条 RTL 指令则有一条与之相对应的目标机指令，但对同一操作意义而言，不同的目标机完成这一操作的指令条数、操作数的形式以及汇编格式各不相同。

Zephyr 提供了一个语言簇 CSDL 以描述机器的规格说明。CSDL 的成份有：SLED(The Specification Language for Encoding and Decoding)[6,7]、λ-RTL[1]、CCL (The Calling Conventions Language)[9]和 PLUNGE(The Pipeline Unifying Notation: Graphs and Expressions)。其中，SLED 主要用于定义机器指令的符号表示、二进制表示、汇编语言表示以及它们之间的映射；λ-RTL 用于指定指令的语义；CCL 用于指定过程间进行参数传递和结果返回的方式；PLUNGE(目前还不够成熟)结合功能单元、控制和数据路径网，描述处理器的流水线结构。虽然每个语言描述机器的单个特性或一组相关特性，但由于它们对可以机器化的核心部分(指令和存储)具有同样的视图，因此可以用来构造具有完整性的和一致性的机器规格说明。如前所述，RTL 虽然便于机器处理，但不便于人使用，用其书写机器描述相当不便。实际上，λ-RTL 是 RTL 的另一种形式，是基于 ML 的模块化元语言，利用 λ-RTL 书写的规格说明，通过转换器将其转换为相应的 RTL。我们以 SPARC 机的一个装入命令为例，说明 CSDL 成份间的关系。装入命令用 SLED 可描述为：

```
constructor
  ld [address],rd
```

该指令的汇编表示和二进制表示可从其符号表示中推断出来，因而可以忽略。这条指令的语义用相应的 λ-RTL 可表示为：

```
Default attribute of
  ld (address,rd) is $r[rd] := $m[Address]
```

CSDL 提供了57基本的 RTL 操作码。目标机上的运算尽可能地用相应的标准操作码表示。如果目标机上的运算没有与之相对应的标准操作码，就用相应的几个标准操作码表达式表示。在必要的情况下，用户可以自己定义操作码。现分析比较如下：

(1)Zephyr 用四种语言描述机器特性。这样做的好处在于：a)单个特性描述易于复用；b)机器描述仅与目标机有关，这样不仅可以用于构造编译器，也可用于建立其它工具；c)使语言的设计尽可能地根据其描述的特性进行剪裁，以简化描述。其不足之处在于：用多语言描述的规格说明的一致性比较难以保证；

(2)Zephyr 的 RTL 是结构化的，而 GCC 的 RTL 是无结构化，因此 Zephyr 的机器描述层次比较清晰，且易于处理比较复杂的问题；通常情况下，Zephyr 的机器指令的汇编表示和二进制表示都是隐式给出的，从而使机器描述相当简洁；

(3)在 Zephyr 中，指令的语义与机器无关，并提供了一些标准操作码，对于一个新的目标机而言，只定义其特有的部分指令就可以了，从而使机器描述比较简洁且可复用性好。而在 GCC 中，不同目标机上完成同一操作的指令条数、操作数的形式以及汇编格式各不相同，因而机器描述几乎不具有可复用性；

(4)在 GCC 中，对于任何目标机，insn 集合不发生变化，因而从语法树到 RTL 中间表示的转换比较简单；而在 Zephyr 中，用户可以定义新的 RTL 操作符，这虽提高了机器描述的灵活性，但不利于从高级中间表示到 RTL 表示的转换。

4.3 编译主体与机器描述间的接口

GCC 和 Zephyr 的机器描述文件都是普通的正文文件，在编译过程中，如果为构造或匹配 RTL 在这种文件中搜索，那将是极其缓慢的。为此，一方面，它们在编译内部设计了一套专门的函数和数据结构作为编译主体与机器描述之间的接口；另一方面，在编译之外设计了一套独立的、专用的机器描述处理程序，这套程序将正文形式的机器描述转换成供接口调用的数据结构与函数。

在 GCC 中，机器描述处理程序由11个独立的程序组成。每个程序处理 MD 的一部分内容，它们各自形成相应的 C 源程序，分别作用于 RTL 生成、机器相关优化和汇编代码输出等过程。

Zephyr 将编译主体与机器描述间的接口根据其充当的角色不同将其分为 RTL 生成接口、汇编语言接口及 VPO 代码生成接口这三个通用接口和两个与目标有关的接口[8]。其中，RTL 生成接口提供用于创建 RTL 的函数，汇编语言接口提供段、符号、标号、可重定位地址的表示机制以及输出初始数据的指令；VPO 代码生成接口提供对汇编器以及定义与调用函数的命令的访问。机器指令通过这个接口以 RTL 的形式输出。将与目标机有关的特定信息，利用 SLED、CC、λ-RTL 对其形式化规格说明，并从这些机器规格说明中自动生成相应的函数接口中，放在相应的机器描述接口和特定于目标机的 VPO 代码生成函数，以便于代码扩充器调用。

由此可见，GCC 与 Zephyr 编译主体与机器描述间的接口的设计思想基本类似，但其具体形式存在一定的差异：(1)在 Zephyr 中，抽取一系列与目标机无关的以及各种目标机的共同信息(如：标准操作符的 RTL 生成函数)，并根据其作用不同进行区分，保持各自的相对独立性。这样这些接口不仅可以用于编译器的构造，而且可以用于其它工具的构造[8]。将通用的函数与数据结构与特定于目标机的接口函数和数据结构

(下转第120页)

作用。

数据挖掘是从大量的、不完全的、有噪声的、模糊的、随机的数据中抽取和精化新的模式，也可以称为知识发现(KDD)。1989年，Fayyad 定义“KDD 是从数据集中识别出有效的、新颖的、潜在有用的，以及最终可理解的模式的非平凡过程”。相对来讲，数据挖掘主要流行于统计界、数据分析、数据库和管理信息系统界；而知识发现则主要流行于人工智能和机器学习界。

数据挖掘过程可简单地理解为：数据准备(data preparation)、数据开采，以及结果的解释评估(interpretation and evaluation)。数据挖掘使用的主要方法有关联规则发现、分类规则发现、聚类规则发现、依赖关系发现、预测模式发现等。数据挖掘的范围非常广泛，可以是经济、工业、农业、军事、社会、商业、科学的数据或卫星观测得到的数据。数据的形态有数字、符号、图形、图像、声音等。数据组织方式也各不相同，可以是有结构、半结构、非结构的。数据挖掘的结果可以表示成各种形式，包括规则、法则、科学规律、方程或概念网。应用于 Web 的数据挖掘流程如图6所示。

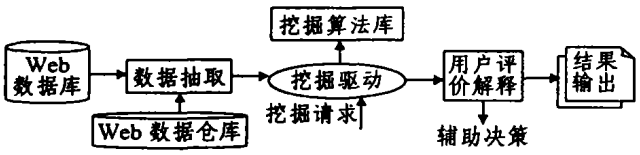

图6

Web 数据挖掘的方法是从 Web 数据库及数据仓库中提取相关数据，其中数据仓库是面向主题的、集成的、稳定的、不同时间的数据集合，用以支持经营管理中的决策制订过程。对提取的数据进行二义性分析，消除不一致性，集成数据。挖掘驱动根据挖掘请求在挖掘算法库中选择合适的挖掘算法，并使用选定的算法执行数据挖掘任务。用户评价可以借助专家系统的方法，以一种直观的方式来表现数据挖掘的结果，提供一种人机交互的友好界面。将数据仓库、数据挖掘和专家系统结合，是进行综合决策最有效的技术途径。

目前已有一些著名的系统和应用实例，如加拿大 Simon Fraser 大学开发的一个多任务 KDD 系统 DBMiner(目的是把关系数据库和数据采掘集成在一起，以面向属性的多级概念为基础发现各种知识)，IBM 公司 Almaden 研究中心开发的 QUEST(为新一代决策支持系统的应用开发提供高效的数据采掘基本构件)，Acknosoft 公司开发的诊断和预测系统 CASSIOPEE，HNC 公司开发的信用卡欺诈估测系统 FALCO 等。

结束语 Internet 的出现和普及给企业提供了更新的挑战和机遇，这对管理者也提出了更高的要求。面对网络提供的共享信息、快速沟通、相互合作等优点，管理者应能快速、准确地对生产和经营的业务过程作出决策。通过研究和开发基于 Web 的决策支持系统，可以为管理者提供远程决策支持和服务，通过网络为企业的管理和经营活动获得丰富信息资源、案例咨询和决策建议，以更好地满足客户的需求，实现更高的自身价值。

参 考 文 献

1 史忠植. 高级人工智能. 科学出版社，1997
2 史忠植，沈沧海，李云峰. 智能决策支持系统开发平台 IDSDP. CIMS-CHINA'96，1996
3 陆汝钤. 专家系统开发环境. 科学出版社，1994
4 高洪深. 决策支持系统--理论方法案例. 清华大学出版社，2000
5 Carmel E，Herniter B C. MEDIANSS：Conceptual design of a system for negotiation session. In：Proc. of 9th Conf. on DSS，DSS-89 Transactions，1989. 239～253
6 Kersten G E，Noronha S J. Supporting international negotiations with a WWW-based system：[Interim Report (IR-97-49)]. International Institute for Applied Systems Analysis，Laxenburg，Austria，1997

(上接第139页)

相分离，大大提高了模型的清晰度和模型的可复用性，从而减少在编译器构造过程中的重复劳动；(2)在 GCC 中，对机器描述处理程序处理方式要比 Zephyr 的层次性好。

结束语 GCC 和 Zephyr 编译基础设施提供了相应的编译器重组技术，将编译前端、后端和机器描述各部分相对分离又相互配合，使在编译系统的重组过程中，仅修改或替换与用户研究者有关的部分，大大缩短了编译器的开发周期，从而达到快速地建立高质量的编译器的目的。通过提供支持多源语言多目标编译系统所需解决的三个根本问题的关键技术，从而为快速实现多语言和多平台移植提供了有力的支持。而这些编译器输出可执行代码的质量并没有因为它支持多目标而降低。研究表明[3]，对于多数目标机而言，利用这两种设施产生的编译器输出代码的质量与产品编译器所输出代码的质量差不多。

本文通过从总体结构及其所采用的关键技术进行了分析比较，得出了 Zephyr 所采用的技术比 GCC 具有较强的优势，但 GCC 是自由软件，目前使用更广泛一些。GCC 和 Zephyr 编译基础设施所采用的技术代表着交叉编译技术的发展方向。如何进一步提高编译后端生成的自动化程度，是我们进一步探讨的课题。

参 考 文 献

1 Norman R，Jack W D. Machine Description to Build Tools for Embedded Systems. http://www.cs.virginia.edu/zephyr/
2 Andrew A，Jack D，Norman R. The Zephyr Compiler Infratructure. http://www.cs.virginia.edu/zephyr/
3 Manuel E B，Jack W，D. Target-specific Global code improvement：Principles and Application：[Technical Report CS-94-42]. Department of Computer Science University of Virginia. http://www.cs.virginia.edu/zephyr/
4 http://gcc.gnu.org/.
5 赵克佳，杨灿群，罗红兵. 多语种多平台编译系统剖析——GNU CC 的解剖与移植. 1997
6 Norman R，Mary F F. Specifying Representations of Machine Instructions. ACM Trans. on Programming Languages and Systems，1997，19(3)：492～524
7 Norman R，Mary F F. New Jersey Machine-Code Toolkit Architecture Specifications. http://www.cs.virginia.edu/zephyr/
8 Jack W D，Steve C L，Norman R. Zephyr Code-Generation Interfaces. http://www.cs.virginia.edu/zephyr/
9 Mark W B，Jack W D. A Formal Model and Specification Language for Procedure Calling Conventions. The 22nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages，POPL '95，SanFrancisco，CA，January 1995
10 Nikil D，Alex N，Hiroyuki T，Ashok H. New Directions in Compiler Technology for Embedded Systems. In：Proc. of the Conf. on Asia South Pacific Design Automation Conf. Yokohama Japan，2001