文章编号:1001 - 9081(2005)07 - 1674 - 02

一种可用于编译器调试的目标代码验证方法

琚小明[1],姚庆栋[2]
(1. 华东师范大学 软件学院,上海 200062;2. 浙江大学 信息与电子工程学系,浙江 杭州 310027)
(xmju@sei.ecnu.edu.cn)

摘 要:传统编译器测试方法是通过比较预期的结果和待测的结果是否一致,以确定编译器是否存在错误。在此基础上,提出了引入参考编译器和参考仿真器的测试方法,在指令集软件仿真过程中生成可用于编译器调试的动态数据信息文件,对参考动态数据信息文件和待测动态数据信息文件进行比较,编译器测试工具可根据比较的结果来确定待测编译器存在错误的位置,这对编译器的调试是非常有用的。

关键词:目标代码验证;软件测试;编译器;软件调试
中图分类号:TP314 **文献标识码**:A

A method of object code verification for debugging compiler

JU Xiao-ming[1], YAO Qing-dong[2]
(1. *Software Engineering Institute, East China Normal University, Shanghai* 200062, *China*;
2. *Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou Zhejiang* 310027, *China*)

Abstract: Traditionally, compiler-testing method confirms whether some bugs occur in the tested complier through comparing the anticipative results with tested results and figuring out whether the former is equal to the latter. Based on the traditional complier testing, a method that introduced a reference compiler and a reference simulator into compiler testing was proposed. Firstly, dynamic data information files for testing and debugging were generated during object code on software simulation, according to the relationship between reference compiler and tested compiler. Then, the comparison of reference dynamic data information file and tested dynamic data information file helped to locate the bugs in tested compiler.

Key words: object code verification; software testing; compiler; software debugging

0 引言

软件可靠性的一个重要领域是软件测试[1],软件测试的概念是根据比较期望的软件执行结果和被检测软件产生的结果,查找在计算机程序中的错误[2,3]。软件输出结果的语义是根据应用的不同而不同的,可以是一个数据、数组、数据结构等的不同表示。软件可靠性的验证可以通过动态测试进行,动态测试是使用预先选择的测试用例来执行程序以达到测试的目的。

程序的验证通常是在源代码上执行的,然而,最终执行的是目标代码而不是源代码,因此,对源代码的验证并不能保证目标代码执行的正确性[1,4]。

编译器是一种特殊的系统软件,因为编译器的输入和输出都是应用软件或系统软件。由于编译器的这个特殊性和编译器本身结构的复杂性,使编译器的测试验证面临着巨大的挑战[1]。目前,关于编译器的验证问题已经有广泛的研究,例如编译器使用的主要算术逻辑的验证、算术表达式的验证、编译器结构的验证等。但这些编译器的验证方法都比较复杂,实用性和灵活性也不强。

为了更有效地验证编译器,本文在分析传统编译器测试的基础上,提出了引入参考编译器和参考仿真器的测试方法,并生成可用于编译器调试的信息文件。

1 指令集软件仿真器

媒体处理器 MD32 是由浙江大学信息与电子工程学系开发的,其指令集由三部分组成:MDF,MDD 和 MDS。其中 MDF 是面向 RISC 的寄存器操作数的指令,与 MIPS-I 的指令系统是兼容的;MDD 是面向 DSP 具有内存操作数的指令,在 MDF 指令的基础上扩展了多种内存地址寻址方式;MDS 是面向媒体处理的指令,支持多个亚字并行操作的单指令多数据流(SIMD)指令。由 MDF,MDD 和 MDS 构成的指令集是一个复杂的指令系统,为 MD32 指令系统开发的汇编器和指令集软件仿真器支持所有上述这三部分指令的功能仿真。

MD32 指令集软件仿真器主要由两部分组成:源汇编代码的汇编模块和指令功能仿真模块。汇编模块将输入的汇编代码进行词法、语法和语义分析之后,输出二进制的机器码存储在内存中,指令编码存放在代码段,相应的数据存放在数据段。然后指令功能仿真模块从内存中的代码段依次逐条取出指令,读取源操作数,按照指令操作码域指定的操作,执行相应的指令操作并输出指令的仿真结果。

2 引入参考编译器的测试方法

为获得正确的参考结果,采用参考比较的方法,在测试过程中引入一个经过长期实践证明的性能稳定的参考编译器[2],并假设这个参考编译器是完全正确的。用待测编译器

收稿日期:2004 - 12 - 20;**修订日期**:2005 - 04 - 11 **基金项目**:国家 863 计划项目(2002AA1Z1140)
作者简介:琚小明(1967 -),男,浙江衢州人,博士,主要研究方向:计算机体系结构、面向媒体处理器和 DSP 编译器、结构相关的代码优化;姚庆栋(1932 -),男,浙江杭州人,教授,博士生导师,主要研究方向:数字通讯、信息处理、SOC、VLSI.

和参考编译器对同一个测试用例集进行编译测试,然后将各自得出的结果一一比较,如有不同,即可认为待测编译器某个部分存在错误。当然,目前还不存在一个完全正确的编译器作为参考编译器,虽然无法证明一个编译器是完全正确的,但是至少可以证明一个新的编译器和一个已经十分成熟和稳定的编译器至少达到同等的可靠性。

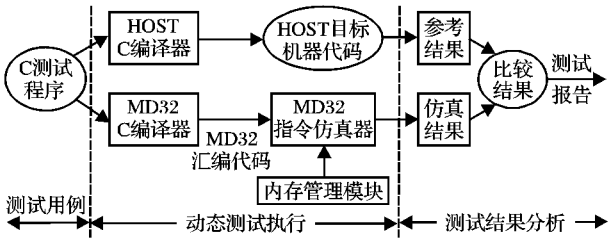

图 1　以 HOST 编译器为参考的测试方法

如图 1 所示,选用 HOST 机器上的编译器(如 Visual C ++)作为参考编译器,测试用例在 HOST 编译器上进行编译并运行,可以方便得到测试用例的运行结果,并将这个结果作为参考结果。然后测试用例在待测编译器上编译,输出的汇编代码在 MD32 指令集软件仿真器上仿真,可以得到测试用例的仿真结果。将仿真结果与上述的参考结果进行比较,得出关于编译器正确性与否的结论。

在这种编译器测试方法中,测试用例的选择是任意的,在选用测试用例时无须生成预期的测试结果。通常测试用例选择一些经典的算法程序,如 FIR 滤波器、FFT 快速傅立叶变换、DCT 离散余弦变换等,这些程序的参考结果很容易经过 HOST 编译器得到。这种引入参考编译器的测试方法,有效排除了预先给定运行结果的测试方法中人为因素对测试结果的影响。

由于 MD32 的指令集包含 MIPS I 的指令集,MD32 指令集软件仿真器能够仿真 MIPS R3000 的目标代码,并能得出正确的运行结果。基于这个前提,本文提出了采用 MIPS 编译器作为参考编译器的测试方法,这种测试方法不仅能够实现上述编译器测试方法的功能,而且通过软件仿真器还能进一步分析和确定编译器出错的位置[5],给编译程序的调试提供纠错的信息。

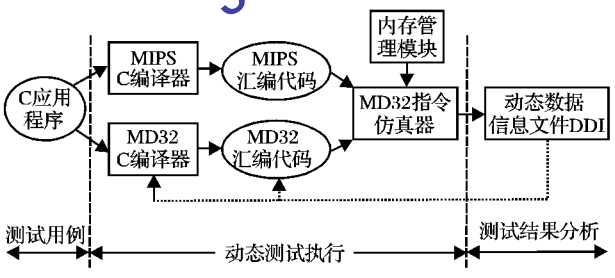

图 2　以 MIPS 编译器为参考的测试方法

图 2 是以 MIPS 编译器作为参考编译器的测试方法,其中 MIPS 和 MD32 使用同一个指令集软件仿真器。MIPS R3000 和 MD32 的目标代码都不能直接在 HOST 机上运行,都需要在 MD32 软件仿真器上进行仿真,比较两者的仿真结果就能实现以 HOST 编译器作为参考编译器的测试功能。

采用参考编译器和参考仿真器的测试方法只要考虑有效的测试用例,而在测试用例中无需包含预期的测试结果,提高测试用例选择的随机性和灵活性。这种编译器测试方法具有以下的功能和特点:

1)测试用例的选用是任意的,具有随机性;

2)无需计算预期的结果;

3)宽度测试和深度测试兼有,测试更加全面;

4)提供可定位编译器错误位置的调试信息;

5)实现编译器测试自动化更容易。

3　编译器测试工具的结构

在以 MIPS 编译器作为参考编译器的测试方法中,由 MIPS 参考编译器产生的 MIPS 汇编代码在 MIPS 仿真器上仿真,在 MD32 编译器的测试环境中,由于 MD32 包含 MIPS R3000 的指令集,MIPS 仿真器可以用 MD32 仿真器来替代。MIPS 仿真器可以进一步推广到其他目标机器的仿真器,本文称之为参考仿真器。

图 3 为编译器测试工具的结构,MD32 软件测试工具也是 MD32 集成开发环境(IDE)的一个部分,代码测试工具控制了两个模块,被 IDE 控制的这两个模块分别用于产生参考结果和产生待测结果。为了验证 MD32 目标代码,采用参考编译器来得到一个相应的参考汇编代码,并在引入的参考仿真器上进行仿真,将所需要的数据、信息送到参考动态数据信息(DDI)文件中,以此作为目标代码测试的参考 DDI 文件。同样地,MD32 汇编代码在 MD32 仿真器上仿真,得到待测的 DDI 文件,将待测的 DDI 文件与参考的 DDI 文件进行逐项比较,如果存在错误,可以将目标代码的错误定位到某个函数。

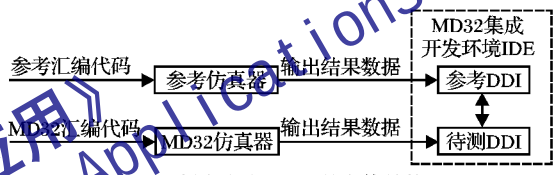

图 3　编译器测试工具的主体结构图

在动态数据信息文件中记录的不仅仅是一个运行结果,而是关于汇编代码在 MD32 软件仿真器上仿真时函数调用入口参数、函数返回值以及函数所在的行号等信息。这些信息的收集是在汇编程序和指令集软件仿真程序中插装代码实现的,图 4 是动态数据信息文件收集的流程图。

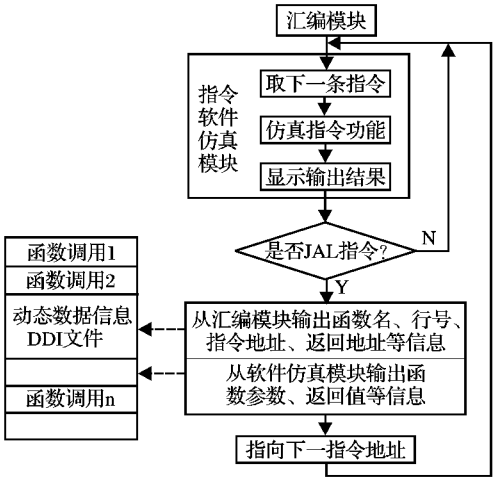

图 4　指令仿真阶段收集 DDI 的流程图

在对汇编代码程序进行语法分析时,动态数据信息文件可以得到函数名、所在行号、函数参数个数及有无返回值等信息,此时还不能得到函数参数与返回值的运行数据。在汇编代码语法分析结束后,汇编代码按照数据段和代码段分别存储在内存的不同地址中,然后,指令集软件仿真程序从代码段每次读取一条指令进行指令的仿真操作。如果被仿真的指令是关于函数参数或返回值的,仿真结果将同时被写到动态数据信息文件,以此实现将仿真运行数据动态填到动态数据信息文件。

(下转第 1694 页)

两种数据报格式基本一样,只是在原因代码部分,C 函数库支持协议包的格式将 16 ~ 30 位均作为系统保留位。ADP 协议包结构如图 4 所示。

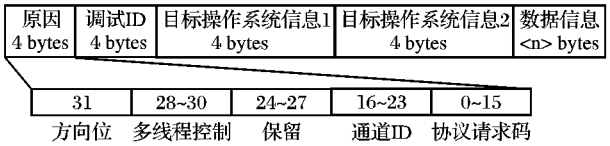

图 4 ADP 调试协议包格式

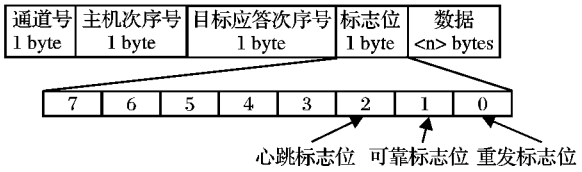

图 5 通信层协议包格式

通道层协议是 ADP 协议的关键协议层,数据提供层将数据包提供给通道层,通道层协议在 ADP 预先定义好的通道上,经过通道层一系列完善的可靠性和错误检测机制,最后在该数据包前加上 4 个字节的头信息,再发送给设备层。这 4 个字节的头信息按按传输的顺序分别是:通道 ID、主机次序号、目标应答次序号、标志位,格式如图 5 所示。通道号是在数据提供层的头信息中预先定义好的,每项服务都有一个全双工的通道分配给它,一个请求通道和一个应答通道,应答信号包在应答通道被接收,请求信号包在请求通道被传送。主机次序号和目标应答次序号用来检测传送方与接收方之间的相对进程,数据包被传送时,含有当前传送和应答的次序号,次序号都是从 0 开始,只有当数据包被检测为可靠的数据包时,次序号才会加 1。通道层传送的数据有三种类型:请求重发的数据包、“心跳”数据包以及可靠/非可靠数据包。当接收方发现错误的数据包时,接收方便发送一个重发请求数据包,数据包中重新分配新的次序号;“心跳”数据包中没有数据,只包含数据包所必需的次序号,是为了确定传输线的另一端是否还处在活动状态;通过检测数据包的长度和数据包的次序号是否符合要求,来设定通道层的可靠标志位。当接收端接收到一个可靠的数据包后,发出的应答信号返回后,下一个可靠的数据包才有效,在这个来自对方端的应答信号中,应答次序号为下一个更高的次序号。

设备驱动层协议根据所使用的设备不同而采用不同的协议。对于串行通信设备,使用的是点对点的字节串行传输。对于以太网通信设备,采用的则是 UDP 协议。设备驱动层实现检测错误数据包的功能,如帧数据包的检测、错误数据长度的检测,通过检测接收 CRC 过程是否发生中断。为了避免数据值在传输过程中出错,设备层协议会将当前数据换码,即先将原始数据与 0x40 相或,所得结果数据值后面再添加一个特征值(0x1B),这样将原始数据转换成一个数据对来传输。例如:原始数据 0x11,先将 0x11 与 0x40 相或,得到 0x51,再和 0x1B 组成一个数据对,最后传输的是 0x1B 0x51 数据对。显然特征值 0x1B 自身也要被打包,变成数据对 0x1B 0x5B 来传输。

3 结语

本文给出的基于 Ethernet 接口的通过 MCU + CPLD 实现的仿真系统,不但速度上较常用传统串口、并口仿真器快许多,而且通过 Ethernet 可以实现多人同时开发及远程调试,价格远远低于市面仿真器产品,且实现也不是很复杂,是一种易于被 ARM 个人开发者和团体开发组接受的高性价比、实用型仿真器。

参考文献:

[1] 杜春雷. ARM 体系结构与编程[M]. 北京: 清华大学出版社, 2004.

[2] 李驹光, 聂雪媛, 江泽明, 等. ARM 应用系统开发详解——基于 S3C4510B 的系统设计[M]. 北京: 清华大学出版社, 2003.

[3] ARM Ltd. Angel Debug Protocol_ ARM DUI 0052C[S], 1997.

[4] ARM Limited. ARM Software Development Toolkit User Guide (version 2.51)[M]. ARM DUI 0040D, 1997.

(上接第 1675 页)

由于汇编和软件仿真是两个不同的阶段,动态数据信息文件 DDI 的收集是由以下几步来完成的:

1) 在汇编阶段收集函数名字、参数个数、参数所在寄存器编号、返回值所在寄存器编号、函数所在行号、指令地址及返回地址等信息,这里指令地址是指汇编器将指令存放在代码段中的地址。

2) 在指令仿真阶段根据函数被调用的情况,将传递函数参数的寄存器的内容以及存放函数返回值的寄存器的内容输出到 DDI 文件中,得到一次函数调用的动态数据信息 DDI。

3) 对于每次函数调用重复以上步骤,可以得到整个程序函数调用序列的信息。

函数在被调用之前要将实参值存放到指定的寄存器中,这些寄存器是约定用于传递函数参数的,然后由汇编指令 JAL 跳转到被调用函数的代码段。根据 MD32 汇编代码中函数调用的这个约定,在函数调用之前对 R4 ~ R7 寄存器用数据装载指令(LW)将实参放到寄存器中。同理约定函数标量返回值是由 R2 寄存器来返回的,在函数返回之前,也是用数据装载指令 LW 将函数返回值放到该寄存器中,然后函数返回。

指令软件仿真的功能是每次从内存中的代码段读取一条指令编码,根据指令的操作数域编码读取源操作数,并按照操作码域的编码执行相应的指令操作,然后显示指令的执行结果。为了获得函数参数及返回值的实际数据,在函数入口处(即在 JAL 指令跳转到函数模块时),记录参数传递寄存器 R4 ~ R7 的值,在函数出口处记录函数返回寄存器 R2 的值。将参数寄存器 R4 ~ R7 和 R2 的值输出到 DDI 文件的当前函数位置,实现一次函数调用 DDI 信息的收集。

本文编译器测试方法的主要特点是将编译器测试看作是一个连续的过程,并且主要关心由编译器产生的目标代码的应用质量。如果编译器不能产生有效的应用,则根据数据信息文件中的信息定位出错误产生的地方并消除相关的错误,而这种错误单凭调试器是很难发现的。该测试方法通过对目标代码间接的测试,实现对待测编译器的验证或测试的目的。

参考文献:

[1] BOYLE JM, RESLER RD, WINTER VL. Do you trust your compiler?[J]. Computer, 1999, 32(5): 65 - 73.

[2] POPOVIC M, KOVACEVIC V, TEMERINAC M. Software testing concept used for MAS/C-compiler[A]. Proceedings of the 26th Euromicro Conference[C], 2000, 2. 224 - 229.

[3] JAY C. Experience in functional-level test generation and fault coverage in a silicon compiler[A]. Proceedings of the European Design Automation Conference[C], 1990. 485 - 490.

[4] SUBRAMANIAN S, COOK JV. Automatic verification of object code against source code[A]. COMPASS'96[C], 1996. 46 - 55.

[5] METZ ML, JORDAN J. Verification of object-oriented simulation designs[A]. Proceedings of the Winter Simulation Conference[C], 2001, 1. 600 - 603.