第 42 卷 第 6A 期
2015 年 6 月

计 算 机 科 学
Computer Science

Vol. 42 No. 6A
June 2015

TI DSP C 语言编译器正确性测试

孙海燕 陈跃跃 王 峰 杨灿群 阳 柳 王 霁

(国防科学技术大学计算机学院 长沙 410073)

摘 要 TI DSP 广泛应用于工业控制任务中,其可执行代码的可靠性不仅依赖于程序本身的可靠性,而且也依赖于编译器的可靠性。选取在工业控制领域中应用广泛、具有代表性的 DSP 芯片 TI C6701 对其编译器进行正确性测试。测试结果表明,用户在不加限制地使用 TI C6701 编译器时,可能会遇到编译器的正确性问题,进而影响整个应用系统的正确性。

关键词 编译器测试,C89 规范,DSP,C6701

中图法分类号 TP31 文献标识码 A

Correctness Test of TI DSP C Compiler

SUN Hai-yan CHEN Yue-yue WANG Feng YANG Can-qun YANG Liu WANG Ji

(School of Computer Science, National University of Defense Technology, Changsha 410073, China)

Abstract TI DSP is widely used in industry control areas. The reliability of the executable code not only relies on user programs, but also relies on the compilers. This paper did correctness test on the C compiler of TI C6701 DSP, which is widely used in many industry control areas. The test result shows that if users use TI C6701 Compiler without any restricts, they may meet the C compiler's correctness problems which may affect the correctness of the whole applications.

Keywords Compiler test, C89 standard, DSP, C6701

1 引言

在工业控制任务中使用了大量的 DSP(Digital Signal Processor)处理器来处理高速数据并进行复杂运算。目前,很多 DSP 应用软件是基于 C 语言开发的,而使用 C 语言编写的程序需要经过编译器编译生成可执行代码。因此,可执行代码的可靠性不仅依赖于程序本身的可靠性,也依赖于编译器的正确性、安全性。当前,国内很多工程任务中使用的 DSP C 语言编译器大多是厂家随 DSP 产品提供的,他们并没有提供安全使用规范和编译器源代码,因此有必要对其编译器的正确性进行测试和验证,为可靠性要求较高的工程应用[1]产品质量提供保障。TI 公司所提供的 C6000 系列 DSP[2]是工业控制中应用最广泛的 DSP 芯片之一,本文选取在工业控制领域中应用广泛、具有代表性的 TI DSP 芯片 C6701 并对其编译器进行正确性测试。

2 编译器测试方法

一般的软件测试方法可以分为黑盒测试和白盒测试,因此对编译器的测试也有两种[3]。由于无法得到 TI C6701 编译器的源代码,因此只能对其进行黑盒测试,黑盒测试一般通过能够覆盖各项功能的大量测试用例对被测软件进行测试验证。而编译器测试技术不同于普通软件的测试技术,编译器测试基本原理如图 1 所示。编译器测试首先是将测试用例的源代码通过被测编译器进行编译;若被测编译器通过该测试用例,则将可执行代码在目标机上运行,查看运行输出结果,如果和预期结果相同,就可以认为被测编译器对该测试用例是正确的,否则就是错误的;如果测试用例程序含有和被测编译器所支持的语言规范不一致的代码,编译器也应当能够识别并报告错误。

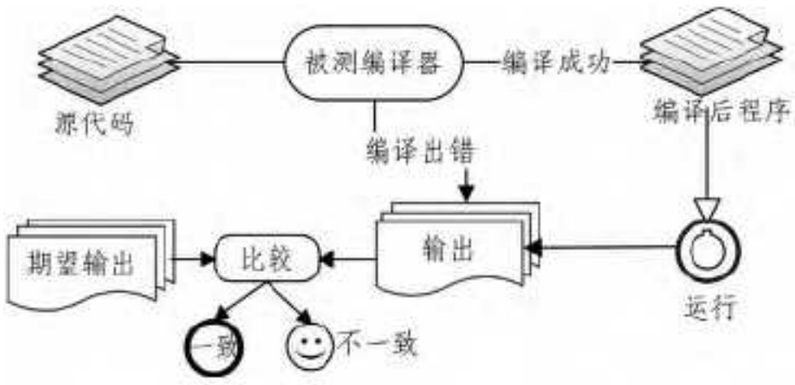

图 1 编译器测试基本原理

基于上述原理进行的编译器测试过程的关键是要保证测试用例的正确性以及测试用例对被测编译器所支持语言各功能点的覆盖率。因此,编译器测试的关键是测试用例的开发。

目前常用支持编译器测试的工具有 PV-Suite[4]和 CV-Suite[5]。PV-Suite 是 Perennial 公司开发的完整的商用编译器和操作系统测试套件。该套件是利用断言的方法,基于语言所定义的规范,而不是根据某一个特殊的编译器开发的通用目标测试程序的集合,可针对 C、C-Freestanding、C++、Embedded C++和 Java 等语言进行一致性测试。CV-Suite 是 Plum Hall 公司基于 Perl 语言开发的商用编译器测试程序包。

由于目前国内外普遍采用的编译器测试套件均为商用测

孙海燕(1976—),女,博士,副教授,主要研究方向为计算机编译技术、嵌入式应用、分布计算等,E-mail:Helen@nudt.edu.cn。

试包，因此有必要自主开发编译器验证程序包，以验证编译器自身的正确性。

3 C语言测试用例开发

C语言是目前工程控制领域中应用最广泛的高级语言，因此本文的编译器测试主要指C语言编译器测试。C语言编译器的正确性测试，主要包括C语言编译器对C语言规范的符合性测试以及表达式的计算正确性测试两个方面。因此，C语言编译器测试用例主要包括覆盖C语言规范的语法点测试用例和覆盖编译器所支持表达式的表达式测试用例

3.1 语法点测试用例

TI C6701编译器支持多种C语言标准，包括C89，C99和K&R兼容的C语言标准等。由于ANSI C X3.159-89标准（简称C89）[6]相对其它C语言标准，对安全性要求更高，因此本文的语法点测试用例根据C89标准编写，用于验证TI C6701编译器对C89规定的Lexical Elements，Conversions，Expressions，Constant Expressions，Declarations，Statements，External Definitions，Preprocessing Directives规范的支持。

在开发语法点测试用例时，我们严格按照C89规范的定义，完整地对语法元素进行测试，对语法点的覆盖率达到100%，并通过正面测试用例和反面测试用例对每个语法点进行覆盖。正面测试用例主要用于覆盖C89规范中所定义的语法点的规定行为，若TI C6701编译器完全符合规范定义，则该编译器应当接受并正确编译该测试用例；若TI C6701编译器在编译过程中或目标码在运行过程中出错，则TI C6701编译器不符合规范对该语法点的定义。反面测试用例主要用于测试不符合规范中对于语法点定义的行为，如果TI C6701编译器接受此测试用例，则说明TI C6701编译器针对此语法点是错误的。由于在编写有参数的测试用例时都编写了基于参数边界值的测试用例，因此也能够保证所有语法点的参数边界值测试。

如在C89规范6.3.2.1 Array subscripting小节中对数组进行了如下限定：

One of the expressions shall have type "pointer to object type", the other expression shall have integer type, and the result has type "type".

该规范说明数组中的一个表达式类型应该为“指向对象的指针类型”，另一个表达式类型为整数类型，结果类型为对象的类型。针对此规范开发了如下5个测试用例，包括4个反向测试用例和1个正向测试用例。

测试用例1：

```
int main()
{
  100[100]=1;
  return 0;
}
```

测试用例2：

```
void ign(const void*);
void f(int,…);
int main()
{
  struct unknown *p;
  f(0,p[0]);
  ign(&p);
  return 0;
}
```

测试用例3：

```
void ign(const void*);
int main()
{
  int a[2],b[2];
  a[b]=1;
  ign(a);
ign(b);
  return 0;
}
```

测试用例4：

```
int main()
{
  struct unknow s[10];
  int a;
  a=s[1];
  return 0;
}
```

测试用例5：

```
int iarray[10],i=3;
iarray[0]=17;
iarray[3]=3;
/* the array name is equivalent to
a pointer to the first element */
iequals(—LINE—,*iarray,17);
iequals(—LINE—,iarray[i],3);
```

其中测试用例1—4为反向测试用例，测试用例5为正向测试用例。测试用例1和测试用例2测试规范中的第1句描述，表达式类型为“指向对象的指针类型”。测试用例1中数组元素类型为常整型，测试用例2中数组元素类型不完整，编译器应当给出出错信息。如果没有报错，则说明编译器不符合此数组规范。

测试用例3测试规范中的第2句描述，数组下标为整数类型。而测试用例3中第5行数组下标不是整型，因此编译器应当报第5行出错。

测试用例4测试规范中的第3句描述，结果类型为前者。测试用例4第5行，将struct类型的数组元素赋值给整型，也是不符合规范的，编译器也应当报错。

对于符合规范的数组，我们在更详细的数组语义测试中使用，测试用例为可运行的程序，如可以使用测试用例5的测试程序片断。

3.2 表达式测试用例

由于编译器需要支持复杂的表达式运算，因此测试用例开发的另一个关键是表达式测试用例的开发。在C语言的实际应用中，表达式形式多种多样，靠手工编写测试用例，不仅工作量大，而且其正确性、有效性和覆盖率也难保证。为了保证表达式的覆盖率和正确性，我们开发了一个表达式测试用例自动生成器（GeneRator of Expression Automatic Test，GREAT），来辅助进行表达式测试用例的开发。

GREAT 的输入是数据集和表达式模板，表达式模板可以是操作符的任意组合。数据集中定义了变量以及初值，GREAT 从数据集中随机选取符合条件的变量代入到表达式模板中，生成所需测试的复合表达式。GREAT 将复合表达式解析为简单表达式的组合，并依次计算这些简单表达式的值，得出最终的期望值。GREAT 最终输出复合表达式与期望值进行比较的自测试用例。

GTEAT 工具的基本思路是将复合表达式转化为简单表达式的组合，将两者的计算结果进行比较，判断复杂表达式的结果是否正确，生成自测试用例。例如下面的表达式：

(a＊b)＋(c＊d)

编译器在进行处理时可能因为内部分配临时寄存器时出错，但是在进行如下计算时，其出错的可能性就大大降低了：

temp1＝a＊b

temp2＝c＊d

temp1＋temp2

这些简单的二元操作会在基本的验证测试中首先进行测试。这样比较两者的结果就能确定编译器对复杂表达式的处理是否正确。

4 C 语言编译器验证工具 CVT

由于测试用例数量巨大，单靠手工对编译器进行测试并对测试结果进行整理和分析是根本不可能的。为此，我们开发了一个自动的编译器验证工具——C 语言编译器验证工具(C Validation Tools，CVT)用于支持 C 语言编译器的自动测试。CVT 的主要功能包括支持测试用例的运行，支持测试中运行结果值和期望值的比较，支持测试结果的分析和统计，针对编译的整个过程进行验证并有详细日志记录。

CVT 测试工具工作流程如图 2 所示。CVT 根据用户输入的配置参数自动地完成编译器测试，测试过程不需要用户的干预。测试用例包括 C 语言语法点测试用例以及表达式测试用例。CVT 工作流程包括执行测试和测试结果分析两个阶段，其中执行测试过程包括用被测编译器编译测试用例并在目标机上运行测试用例，得到测试输出结果；测试结果分析阶段将测试输出结果与期望值进行比较、统计，并最终形成测试报告。

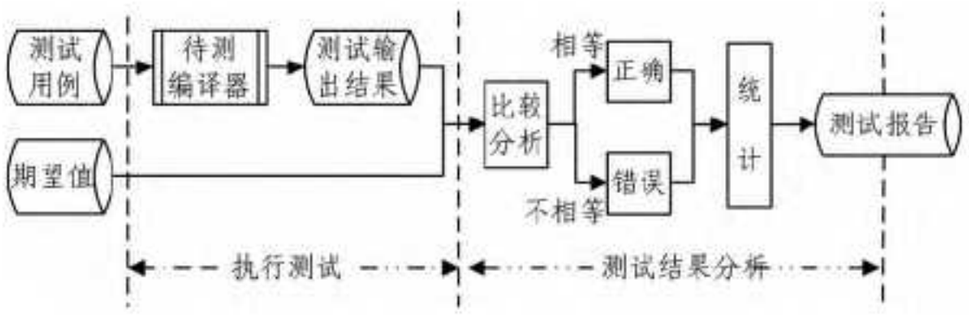

图 2 验证工具 CVT 的工作流程

通过 CVT 工具，可以对编译器的测试用例以及测试用例的不同编译选项进行自动测试，同时能够灵活配置测试内容，能够支持测试结果的自动分析和统计。

5 TI C6701 编译器测试环境

对工程应用较为广泛的 TI 集成开发环境 Code Composer Studio 3.3 版本中的 TI C6701 编译器进行测试。基于 TI C6701 的工程应用调研，发现用户常用的编译选项如表 1 所列。

表 1 TI C6701 编译器常用的编译优化选项

优化选项	含义
－g(Symbolic debugging option)	产生源代码调试指导信息
－q(Suppresses progress messages)	控制所有工具进程信息的产生，只输出源文件名和错误信息。
－mv6700	针对目标芯片类型产生目标码
－ml1	将所有的函数都分配到 32k 内存以外的存储空间
－ml3	将所有的数据和函数都分配到 32k 内存以外的存储空间
O2	二级优化
O3	三级优化

经过分析整理各个优化选项的功能、使用时机以及对应用程序的影响，按照用户要求对 TI C6701 编译器按表 2 所列进行组合(共 5 种组合)测试。这样每个测试程序实际都对应 5 个测试用例，实际测试用例个数＝测试程序＊5。语法点测试用例 100％覆盖 C89 语法点，而表达式测试用例的开发基于 GREAT 工具，覆盖了从 2 到 5 重括号所有表达式的组合。

表 2 TI C6701 编译器优化测试组合

	测试选项	包含的选项
S1		－g －mv6700
S2	S1＋ml1	－g －mv6700 －ml1
S3	S1＋ml3	－g －mv6700 －ml3
S4	S1＋o2	－g －mv6700 －o2
S5	S1＋o3	－g －mv6700 －o3

TI C6701 编译器运行平台为基于 windows 操作系统的 PC 机，目标代码运行环境为 TI 公司的 TI C6701 evm 板。TI C6701 编译器的测试过程分为编译、链接、运行 3 个阶段，我们在每个阶段都进行了详细的记录，从而在出错时能够快速准确定位。测试过程如图 3 所示。基于 CVT 工具针对测试用例逐个进行测试，将测试过程中产生的信息存放到文件 cvt.log 中。

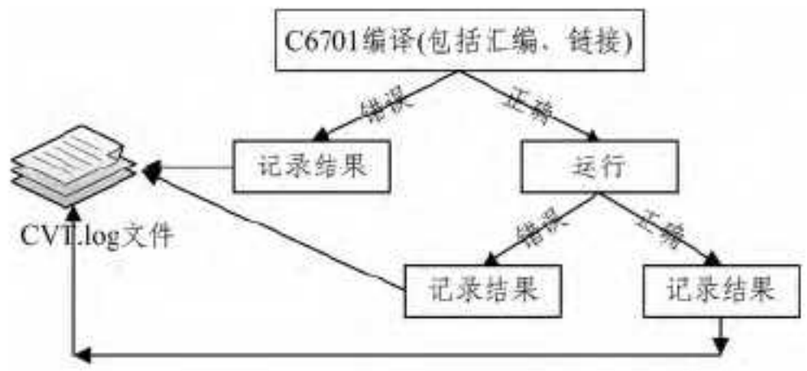

图 3 TI C6701 编译器测试过程

6 测试结果

基于上述测试环境，对 TI C6701 编译器进行了测试，其测试结果如表 3 所列。

表 3 TMS320C6701 编译器测试结果

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

＊＊ C Validation Tool ＊＊

＊＊ (C) 2000—2012 School of Computer，NUDT. ＊＊

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

＊＊＊＊＊ statistics about the whole CVT test ＊＊＊＊＊

	Total	Passed	Failed	Unsupported	Skipped
errtests (compile)	524	479	45	0	0
errtests (uncertain)	92	58	34	0	0
errtests/gnu	0	0	0	0	0
lang	4946	4932	14	0	0
exprtest	54548	54474	78	0	0
total	60110	59943	171	0	0

(下转第 545 页)

理这种情况。

然而，使用本文方法，假如决策者分配的权值如下：

r_1：a 〈3〉

r_2：$\neg b \leftarrow a$ 〈2〉

r_3：$\neg c \leftarrow a$ 〈2〉

r_4：$\leftarrow \neg b, \neg c$ 〈1〉

首先，得出该程序的一致性扩展回答集及其相应的代价为：$S_1=\{a,\neg b\}$，$pd_1=\{r_3\}$，$\Phi p(S_1)=2$，$S_2=\{a,\neg c\}$，$pd_2=\{r_2\}$，$\Phi p(S_2)=2$，$S_3=\{a,\neg b,\neg c\}$，$pd_3=\{r_4\}$，$\Phi p(S_3)=1$。

根据最优解的定义，最终得出的优化解为：$S_3=\{a,\neg b,\neg c\}$，$pd_3=\{r_4\}$。即根据决策者的偏好，删除规则 r_4 付出的代价最小，是决策者最满意的解决方案。

朱涛等人提出了基于最小原理的方法来处理不一致的回答集程序[10]。最小原理的基本的思想是：对于一个不一致的回答集程序，当严格规则一致时，优先删除缺省规则，最大限度地保留严格规则，具体细节参考文献[10]。然而这种方法不能根据用户自己的需求选择最适合自己的解决方案。

如引言中的程序 P：

r_1：rice←

r_2：steak← rice

r_3：¬steak← rice，not pepper

根据最小原理的思想，严格规则 r_1，r_2 是一致的，不存在冲突，所以通过删除 r_3 达到一致性。这种处理方法无法保证删除的规则满足代价最小性，即不能根据用户的偏好关系选择自己最满意的解决方案。通过本文的方法，可以根据用户喜好程度分配不同的权值，优先选择删除规则代价之和最小的。

r_1：rice← 〈1〉

r_2：steak← rice 〈2〉

r_3：¬steak← rice，not pepper 〈3〉

该程序的一致性扩展回答集及其相应的代价为：

$S_1=\{\text{rice},\neg\text{steak}\}$，$pd_1=\{r_3\}$，$\Phi p(S_1)=3$，$S_2=\{\text{rice},\text{steak}\}$，$pd_2=\{r_2\}$，$\Phi p(S_2)=2$，$\Phi p(S_2)<\Phi p(S_1)$，即 S_2 是最优解，删除 r_2 付出的代价最小。

结束语　在实际应用中，由于知识库中存在不一致的知识，导致该回答集程序没有回答集或推出相互矛盾的结果。本文以加权逻辑程序为基础，提出了一种加权的定量方法来处理不一致的回答集程序。相比之前的方法，该方法通过引入代价最小性，方便而简洁地找到了决策者最满意的解，能更好地处理不一致性问题，具有重要的现实意义。

参 考 文 献

[1] Eiter T, et al. Answer Set Programming: A Primer [J]. Reasoning Web: Semantic Technologies for Information Systems, 2009, 5689: 40-110

[2] Gelfond M, Leone N. Logic programming and Knowledge representation. The A-Prolog perspective[J]. Artificial In telligence, 2002, 138: 3-38

[3] Baral C. Knowledge representation, reasoning and declarative problem solving[M]. Cambridge Univ Press, 2003

[4] 赵岭忠，张超，钱俊彦. 基于 ASP 的 CSP 并发系统验证研究[J]. 计算机科学，2012，39(12)：133-136

[5] Deng Wen-jun, Liang Yin-wen. Reason on UML Diagrams with Answer Set Programming[C]// Proc. of International Conf. on Computer Science and Software Engineering. 2008(1): 205-209

[6] Šefránek J. Updates of argumentation frameworks [C]// Proceedings of the 14th International Workshop on NonMonotonic Reasoning. 2012: 1-9

[7] luukkala V, Niemel I. Enhancing a smart space with answer set programming [M]// Semantic Web Rules. Springer Berlin Heidelberg, 2010: 89-103

[8] Balduccini M, Gelfond M. Logic programs with consistency-restoring rules[C]// International Symposium on Logical Formalization of Commonsense Reasoning, AAAI 2003 Spring Symposium Series. The AAAI press, 2003: 9-18

[9] Van Nieuwenborgh D, Vermeir D. Preferred answer sets for ordered logic programs[J]. Theory and Practice of Logic Programming, 2006, 6: 107-167

[10] Zhu Tao, Zhang Zhi-zheng. A Processing Method for Inconsistent Answer Set Programs Based on Minimal Principle[M]. 2012: 270-274

[11] D Nieuwenborgh V, Heymans S, Vermeir D. Weighted-Answer Sets and Applications in Intelligence Analysis[C] // MProc of the 11th Int. l Conf on Logic for Programming. ArtificialIntelligence, and Reasoning, 2006: 169-183

(上接第 515 页)

整个测试过程共运行测试用例 60110 个，其中通过 59943 个，失败 171 个。基于二进制代码生成的反汇编语言，对 TI C6701 编译器错误产生原因进行分类统计，并按照其对系统可靠性的影响进行分类。其中最严重优先级 bug 11 个，次严重优先级 bug 17 个，普通优先级 bug 有 36 个，次普通优先级 bug 有 8 个，最低优先级没有。

通过上述测试，可以发现：用户在不加限制地使用 TI C6701 编译器时，可能遇到编译器的正确性问题，进而影响整个系统的正确性。

结束语　本文的测试结果表明 TI C6701 编译器存在正确性错误，对于可靠性要求较高的领域的 DSP 应用开发有一定的参考价值。相关领域应用在进行软件开发时，程序员能够有意识地规避编译器中存在的正确性错误，从而提高整个应用的可靠性。

参 考 文 献

[1] Kocher P, Lee R, McGraw G, et al. Security as a New Dimension in Embedded System Design[C] // Proceedings of the Design Automation Conference. ACM Press, 2004(6): 735-760

[2] TMS320C62x Code Composer IDE Help (SPRH197), Parallel Debugging Manager: An Introduction[EB/OL]. http://www.ti.com, 2005-06

[3] Popovic M, Kovacevic V, Temerinac M. Software testing concept used for MAS/C-compiler[C] // Proceedings of the 26th Euromicro Conference. Maastricht. IEEE Computer Society, 2000 (2): 224-229

[4] 何群. C 编译器自动测试工具(Ctcgen)的剖析与移植(A)[J]. 计算机工程，2004，30(20)：95-97

[5] Plum Hall Inc. The Plum Hall Validation Suite for CTM[EB/OL]. http://www.plumhall.com/stecl.html

[6] ANSI X3.159-1989. Programming Language-C[OL]. http://www.freestd.us/soft/109648.html