一种面向 CoSy 编译框架的编译优化开发方法

刘 博[1],李蜀瑜[1],阮 园[2]
(1. 陕西师范大学 计算机学院,陕西 西安 710072;
2. 中国电子科技集团公司第五十八研究所,江苏 无锡 214035)

摘 要:鉴于编译器在系统开发中日趋重要的地位和 CoSy 在编译器开发中的良好应用前景,文中引入并介绍了基于 CoSy 的编译器开发的整体流程,并且指出了开发过程中遇到的重点问题和难点问题。文章对编译器开发的主要工作和内容进行介绍,对编译过程进行详细分析,其中对编译器中间代码的优化进行了详细讲解,对窥孔优化思想做了详细介绍。最后,在文章末尾引入具体的开发实例并且结合窥孔优化思想进行具体分析,从而更好地体现基于 CoSy 的编译器开发方法的优势。

关键词:CoSy;CoSy 中级中间表示;编译器开发环境;cgd 文件;DFG;中间代码优化;窥孔优化
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2013)03-0061-04
doi:10.3969/j.issn.1673-629X.2013.03.016

A Development Method of Compilation Optimization Faced Framework of CoSy Compilation

LIU Bo[1], LI Shu-yu[1], RUAN Yuan[2]
(1. Dept. of Computer Science, Shaanxi Normal University, Xi'an 710072, China;
2. China Electronics Technology Group Corporation No. 58 Research Institute, Wuxi 214035, China)

Abstract: In view of the compiler in system development is the important position and the good application prospect of CoSy in compilers, introduce and propose the CoSy compiler based on the development of the whole process and point out the keys and difficulties of the development process. The compiler development work and the main content is introduced. Analyze the compile process in detail, including among the compiler code optimization for detailed explanation. Finally, introduce the development of specific end examples and combine peep hole optimization idea for specific analysis, and better reflect the advantages of the method based on the CoSy compiler development.

Key words: CoSy; CCMIR (common CoSy medium-level intermediate representation); compiler development environment; cgd file; DFG; intermediate language code optimization; peep hole optimization

0 引 言

编译器是计算机软件系统中的重要组成部分,其功能可以简单描述为:将高级语言翻译为机器语言。随着计算机软件行业的飞速发展,编译器也有了迅猛的发展。而新的编译器生成工具也随之应运而生[1]。

CoSy 是一种新的编译器开发框架。它提供开发工具和开发引擎来开发编译器,编译器开发人员只需要专注于目标机进行相关的研究进行开发。CoSy 生成的编译器有可扩展性同可移植性。编译器前端产生的中间代码为 CoSy 中级中间表示(Common CoSy Medium-level Intermediate Representation, CCMIR)[2]。

CoSy 有三类输入文件,分别是 TDF(Target Description),编译器描述文件 EDL(Engine Description Language),以及 CGD(Code Generator Description),输出为面向专用处理器的编译器。CoSy 编译框架生成的编译器使用中间表示语言 CCMIR,并引入引擎的概念。

图 1 为 CoSy 的调试信息流。编译器前端引擎将 C 源码转化为 CCMIR,部分 CCMIR 经过 lowering 引擎处理生成 LIR,LIR 是与目标机相关的部分内容,

收稿日期:2012-07-04;**修回日期**:2012-10-16
基金项目:核高基重大专项基金资助项目(2009ZX 01034-001-002-003);中央高校基本科研业务费专项资金(GK201002011);陕西师范大学研究生培养创新基金(2012CXS056)
作者简介:刘 博(1988-),男,硕士,研究方向为嵌入式系统开发。

LIR 与 CCMIR 统称为 CCMIR,CCMIR 中间还经过优化引擎处理,生成新的 CCMIR[3,4]。最后后端引擎根据输入的目标机信息,匹配 CCMIR 生成目标机指令集的汇编代码。通过它的调试流程图,可以看出编译器的生成流程。基于 CoSy 的编译器开发相对老的开发方式在执行效率上有很大的提高。

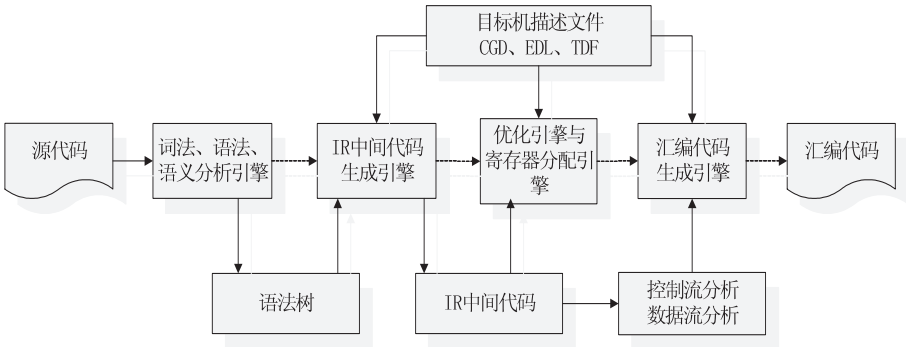

图 1 CoSy 调试信息流

1 生成 CoSy 编译器工作的主要内容

基于 CoSy 编译框架生成编译器主要工作主要集中在 CCMIR 的中间代码优化方法上,使用了多种优化策略,并且针对 CCMIR 的特点进行结合和改进,从而进一步达到优化的目的,以下是生成过程。

1.1 指令集

芯片的指令集和体系结构作深入的了解。对指令集和体系结构有较为深刻的理解对开发工作有很大的帮助,必需详细掌握硬件的寄存器结构[5]。在掌握寄存器结构后才能进行下一步的研究。

1.2 中间语言 CCMIR

给后端引擎提供生成所需汇编代码的信息,前端引擎由 CoSy 自行生成,不需要人为描述。后端的描述主要包括目标机的体系结构特性和 ABI(Application Binary Interface)[6],以及 CCMIR 到汇编代码的映射规则信息。图 2 为 C 源码和 CCMIR 的对应关系图。

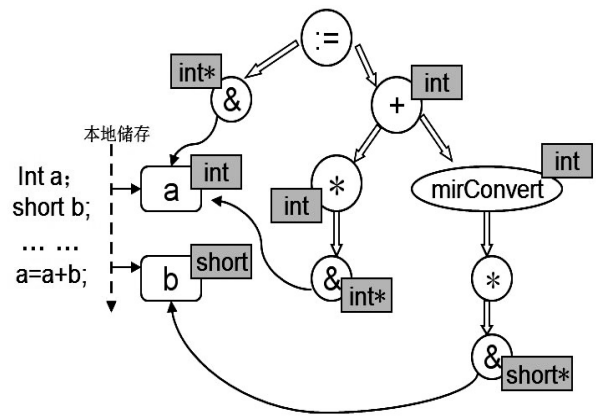

图 2 C 源代码和 CCMIR 的对应关系图

1.3 基于 CCMIR 的中间代码的优化

编译器的优化是除了后端目标机描述之外的非常重要的内容,直接决定了编译器的性能好坏,CoSy 提供了大部分常用的优化引擎,只需要在配置文件中调用即可,但是目标机特有的优化特性,需要新的引擎或修改后端描述文件[7]。

机器相关的代码优化涉及和机器特性相关的部分,主要包括:

1) 寄存器的分配,确定哪个变量使用哪个寄存器。

2) 指令的调度,调整代码的前后顺序以便有效利用流水线技术。

3) 窥孔优化技术,用高效率的代码替换重复率高的低效率代码。

在文中,对指令调度做简单解释,然后对窥孔优化思想在 CCMIR 中的具体实施做以解释和测试。

1.3.1 指令调度算法

指令调度是在满足依赖关系和资源约束还有硬件施加的相关其它约束条件的前提下,重新排列指令执行的顺序,以提高资源利用率,同时让多个操作能够并行的执行,并且减少处理器停顿,缩短程序执行时间。流水线停顿的主要原因是指令间的相关,也是 ILP 开发的各种技术必须解决的关键问题[8]。

1.3.2 窥孔优化技术在 CCMIR 中的应用

窥孔优化技术的基本思想,对编译器所生成的中间代码(或目标代码)中的相邻指令进行考察,将其中一些指令组合替换为效率更高的指令组。窥孔优化的对象是中间代码和目标代码。它的特点是对每个改进都有可能会引发新的改进,为了得到更好的改进,可能需要对目标代码重复进行窥孔优化。

针对本项目,使用窥孔优化的思想对 CCMIR 代码进行一些列的优化包括如下几种策略:

1)在 CCMIR 中消除不必要取数,例如:

Restore reg, A

Load A, reg

很容易看出,后一条指令是不必要的。

但如果后一条指令前有标号,则不能删除它。

2)消除 CCMIR 中不可达代码。

可以删除任何无条件分支后无标号的语句,这样减少了一部分 CCMIR 的代码量。

3)消除间接跳转。

因为 CCMIR 中也使用了 goto 语句,鉴于 goto 语句的复杂性,也鉴于窥孔优化的思想,如果一个 goto 跳到另外一个 goto,则第一个 goto 改为直接跳到第二个 goto 所指的目的地;当清除所有非直接 goto 后,消除标号

后跟 goto 的句子。

4)算术处理。

执行代数简化、常数传播还有强度削减。例如:乘以一个较小整数可用多个加法代替;较大整数的乘法可以联合使用位移和加法来优化。强度削弱优化后的代码长度可能比代码优化前要长一些。

5)在 CCMIR 中间代码中识别特殊指令或模式。

如:add 1, reg

可以改为:inc reg

有时可将多余指令合并为一条特殊指令。这些模式转换既可能依赖于源语言一,又可能依赖于机器。

在文章的第二部分,利用窥孔优化思想,结合实际的算术指令,对编译器生成过程进行详细讲解。

1.4　内嵌汇编

是否支持内嵌汇编是编译器的一项重要指标,文中的编译器只对内嵌汇编的格式做语法检查,但不对指令助记符等进行检查[9]。

2　算术指令 ADD 开发实例

在基于 CoSy 编译器开发过程中,首先需要做的工作是对编译器运行的硬件环境做详细的分析和了解,具体掌握芯片的指令集和体系结构。CoSy 会提供给开发者一个简单的编译器框架,使开发者可以基于此框架进行编译的编写。ADD 指令是编译器中最为普遍的一条指令之一,一般的通用芯片中都要使用此指令[10]。所以,在本实例中以一条 ADD 指令的转换流程作为例子,来具体地展示 CoSy 的开发流程,以便对开发者对流程有一个清楚的认识[11]。

2.1　ADD 指令

ADD 指令在内存中的结构如图 3 所示:

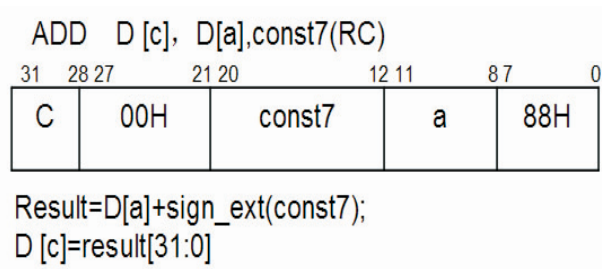

图 3　ADD 指令结构

2.2　ADD 指令到 CCMIR 指令的映射,修改并描述 * cgd 文件

根据上文所示的 ADD 指令结构,基于编译器框架把 ADD 指令映射到 CCMIR 指令,从而编写 * . cgd 中的代码如下。由于文章篇幅有限,在这里只编写 immediates. cgd 中的一条 RULE 指令作为示例:

```
RULE
[add_imm_RC]0:mirPlus(rs1:reg,s2:simm9)->rd:reg;
CLASS IMMEDIATE;
TEMPLATE ALUTMPL;
COST 1;
EMIT{
fprintf(OUTFILE,"\tadd\t%s,%s,#%d",
REGNAME(rd),REGNAME(rs1),s2.Value);
}
```

在开发过程中还要根据具体的芯片指令集和体系结构进行许多的 * . cgd 文件描述,例如 main. cgd, address. cgd, move. cgd 等等。

2.3　acc 编译器的建立

在 linux 环境下运行 CoSy,使用 product extract 指令根据 components 提取需要的引擎和编译器生成文件。然后使用 product enter 进入编译树。选择 compiler 使用 mkc 指令生成可执行的 acc 编译器。

2.4　编译器的测试和调试

建立 test. c 文件,使用./acc - dlir test. c 指令,编译器生成 test. s 的汇编代码,dlfinishgra. out 和 test. gdl 文件。test. c 文件是 C 源代码,用于测试编译器的功能,一般使用较为简单的 C 语言代码段。test. s 是编译器生成的汇编代码,链接器链接编译生成汇编代码从而进行下一步工作[12]。

通常,在编译器生成后,通常是要进行一次而调试的。如果遇到有错误产生,可以从 dlfinishgra. out 文件中,通过方括号中的 RULE 名称,在 * . cgd 文件中找到需要调试的 RULE 语句。如 2)中提到的 immediates. cgd 的 add_imm_RC 指令。这样就能精确地定位错误语句的位置,从而使调试过程更加的方便[13]。

以下是结合窥孔优化思想的代码优化实例,对一些基础指令做出相应修改:

test. c	test. s
int main( )	. ref__builtin__max
{	. ref__builtin__min
	_main;
int a, b;	L_1:
	mov A14, A10
b=a+7;	inc A10, -8, A10
	st. w [A14] # -4, D15
return 0;	inc D0, D0, #7
	movi D15, #0
}	st. w [A10] #0, D0
	movrr D2, D15
	L_2: ret
C 源代码	由对应的 C 源码生成汇编源代码

下面的指令是编译器生成的. out 文件:

```
dlfinishgra.out
L_1[bb2]:  'test.c  L:1  EL:5'  UE=1.00
  @1 [prologue] MultiResult( )
```

```
    @4 [add_imm_RC]Normal(ResPsr-37!)
  @2[usepsr] Normal(VarPsr-34{a}!,ResPsr-36!)
        @3[simm_9] Normal
      @6 [defpsr_Reg_Assi] Normal
        @4[add_imm_RC] Normal(ResPsr-37!)
        @5[localaddr] Normal
  @7 [ldi_reg] Nomal(ResPsr-38!)
      @9 [mov_reg] More (ResNr=0 ->ResPsr-39!)
        @7 [ldi_reg]Normal(ResPsr-38!)
      @8 [func_return_next_is_next]Normal
  @[mov_reg]Move(ResNr=0->ResPsr-39!)
```

在进行调试时,CoSy 还提供了一种可视的图形化的结构分析工具 isee2.0。可以直观地看出 CCMIR 语句结构。

图 4 为使用 isee2.0 软件查看 test.c 对应的 CCMIR 的语句结构。

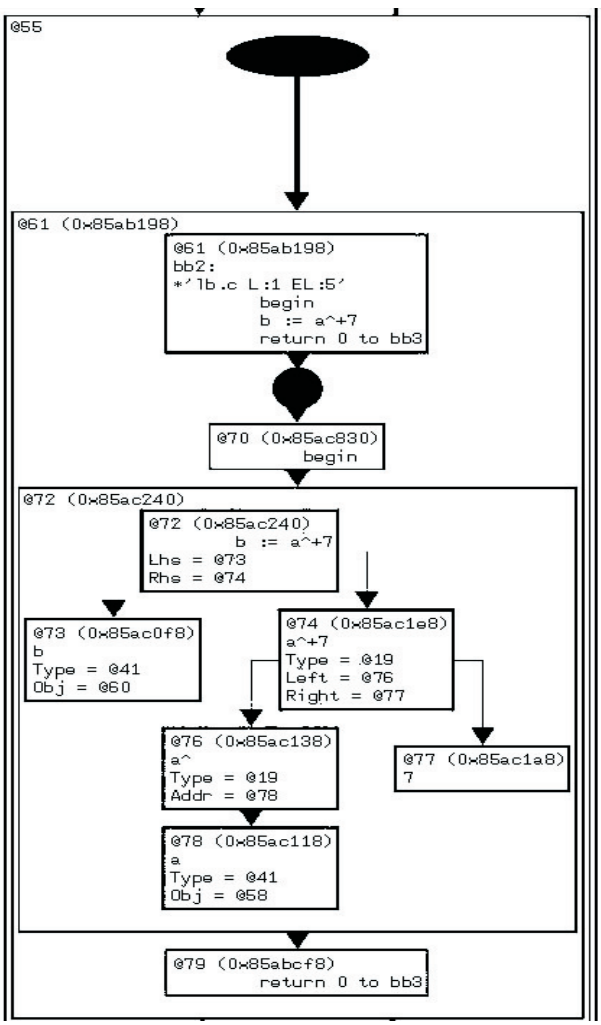

图 4 代码的可视化结构图

3 结束语

文中探讨了使用 CoSy 编译器开发工具来开发编译器的整体过程,总结了使用CoSy作为编译器开发工具的优势。与此同时也对 CoSy 开发过程中涉及到的重要开发部件进行详细地讲解,特别是有关中间代买优化的部分,并且通过示例代码来展示。总结了基于 CoSy 编译器开发的整体流程,这些信息是从大量的实验过程中总结而来的,对基于 CoSy 编译器开发者有一定帮助。在文章最后,通过算术指令转换过程来向读者展示完整的开发流程,使开发者更加清楚地理解 CoSy 的开发过程[14]。

但是,在国内研究 CoSy 的研究人员有限,使得基于 CoSy 的编译器开发过程还是有很多不完善的地方,存在有很多不足。文中只是对基于 CoSy 的编译器开发流程做了简单介绍,在今后的研究中,将进一步深入学习研究 CoSy 编译器开发流程。

参考文献:

[1] 俞甲子. GCC 编译器安全验证方法研究[D]. 杭州:浙江大学,2008.

[2] ACE Associated Compiler Experts bv. CCMIR Primer[EB/DK]. Amsterdam,the Netherlands:[s. n.],(2008-02-07)[2011-04-24].

[3] ACE Associated Compiler Experts bv. CCMIR Definition[EB/DK]. Amsterdam,the Netherlands:[s. n.],(2008-02-07)[2011-04-24].

[4] ACE Associated Compiler Experts bv. CoSy Tutorial[EB/DK]. Amsterdam,the Netherlands:[s. n.],(2008-02-07)[2011-04-24].

[5] Nickray M,Dehyadgari M,Afzal-iKusha A. Power and delay optimization for network on chip[J]. Circuit Theory and Design,2005(3):273-276.

[6] 王凤芹,刘春林,胡定磊. 一种支持 DSP 条件执行指令的编译框架[J]. 计算机工程,2006,32(11):106-108.

[7] 王发鸿,周会平,贾丽丽,等. 基于 GCC 的容错编译器的研究与实现[J]. 计算机工程与科学,2011,33(8):89-94.

[8] 田祖伟,李勇帆. 基于汇编代码的指令调度器的设计与实现[J]. 计算机科学,2009(3):45-47.

[9] 陈 沉,白振兴,向 新,等. 基于 Trace 的即时编译器中代码生成策略的改进[J]. 计算机工程与设计,2011,32(6):2027-2030.

[10] 邓晴莺,张民选. 基于映射表的寄存器文件设计以及编译器优化[J]. 电子学报,2008,36(2):245-247.

[11] 唐沛蓉,黄 春,杨学军,等. Co-array Fortran 编译器的设计与实现[J]. 计算机工程,2007,33(23):84-86.

[12] 李 刚,丁 佳,梁盟磊,等. 安全编码预编译器的设计与实现[J]. 计算机工程,2011,37(3):230-235.

[13] 张 锐,彭启民,赵军锁. 指令级的变量容错恢复[J]. 计算机工程,2010,36(5):43-45.

[14] 禹 丹,严宏志,王继娜. 基于 ANTLR 的 NC 代码编译器的设计与实现[J]. 计算机应用,2008,28(2):522-524.

一种面向CoSy编译框架的编译优化开发方法

作者： 刘博， 李蜀瑜， 阮园
作者单位： 刘博,李蜀瑜(陕西师范大学 计算机学院,陕西 西安 710072)， 阮园(中国电子科技集团公司第五十八研究所,江苏 无锡 214035)

刊名： 计算机技术与发展
英文刊名： Computer Technology and Development
年，卷(期)： 2013(3)