针对GCC编译器指令调度的DUMP算法优化

余晓江，吴亚娟，罗 欣

（西华师范大学计算机学院，四川 南充 637000）

摘 要：GCC编译器可通过DUMP算法记录指令调度过程并输出，但是，只针对使用O0以上的编译优化的前提下，指令调度DUMP算法使编译器在使用O0编译优化时，也能够记录输出指令调度过程，为GCC编译器在移植开发过程中更加直观了解编译器的指令调度并进行执行调度优化。对DUMP算法进行改进优化，使得在使用编译选项-O0的情况下，也可以使用-fsched-verbose=n的n来控制编译器输出指定的调度信息。

关键词：GCC编译器；指令调度；DUMP算法

中图分类号：TP314 **文献标识码：**B **文章编号：**1673-0097(2021)01-0017-02

Optimization of DUMP Algorithm for GCC Compiler Instruction Scheduling

YU Xiao-jiang, WU Ya-juan, LUO Xin

(School of Computer Science, China West Normal University, Nanchong 637000, China)

Abstract: The GCC compiler can record the instruction scheduling process and output it through the DUMP algorithm. However, the instruction scheduling DUMP algorithm enables the compiler to record the output instruction scheduling process when using O0 compilation and optimization only under the premise of compiling and optimizing above O0. For the GCC compiler to understand the compiler's instruction scheduling more intuitively and optimize the execution scheduling during the transplantation development process. The DUMP algorithm is improved and optimized, so that when the compiler option -O0 is used, n with -fsched-verbose=n can also be used to control the compiler to output the specified scheduling information.

Keywords: GCC Compiler; Instruction Scheduling; DUMP Algorithm

0 引言

GNU的编译程序集合称为GNU编译器套件，即GCC（GNU Compiler Collection），是现今最重要的开放源代码软件[1]。GNU编译器套件包括C、C++、Objective-C、Fortran、Java、Ada和Go语言前端，也包括这些语言的库（如libstdc++，libgcj等）[2]。目前，也有很多移植GCC编译器的案例，龙芯编译器是基于GCC4.7.0开发的编译器，RISC-V GCC是基于GCC7.2.0移植开发的编译器。RISC-V GCC编译器是UCB公开并开源RISC-V时，基于GCC编译器研发的配套交叉编译器。

GCC编译器中重要的环节是指令调度。指令调度是根据机器模型对生成的机器指令进行一次重排序，使得机器运行指令时，速度更快，更加准确[3]。指令调度模块中，GCC对指令的整个调度过程进行DUMP操作，使得编译器在编译源程序时，可以通过控制编译选项-fsched-verbose=n的n值来输出编译过程中的信息。指令调度中，DUMP算法是根据编译选项-fsched-verbose=n的n值传递给编译器进行运算，从而有选择性地将编译器的调度过程输出到标准输出流或文件。

虽然GCC编译器设计了调度信息输出DUMP的算法，但是，在编译优化选项为-O0时，编译器会进行指令调度，但不会输出调度信息，编译选项-fsched-verbose=n也不会起作用。开发移植测试GCC编译器时，为了更加明了地梳理调度，常常会使用-O0的优化。GCC编译器的缺陷，即在需要-O0的优化下分析指令调度时不能按照要求输出指令调度过程，从而影响对指令调度的分析与研究。

1 指令调度DUMP算法优化

C和C++等高级语言的源程序经过RISC-V GCC编译器编译时，为了方便优化和更好地适配跨平台，会将高级语言的程序翻译成中间表示，如抽象语法树（AST/GENERIC）、GIMPLE、寄存器传输语言（Register Transfer Language, RTL）等，指令调度是在编译器中接近机器码的RTL阶段进行指令调度和优化。GCC编译器设计了指令调度数据流DUMP算法，但在使用编译优化选项-O0时，不能按照预定输出指令调度信息。因此，对指令调度DUMP算法进行改进，使GCC编译器在使用编译优化选项-O0的情况下，能够完整准确地输出编译器的指令调度数据流信息，便于后期对GCC编译器的移植改进与优化。

算法1.指令调度DUMP优化算法

输入：N：sched_verbose的值

输出：sched_dump：dump到stderr或者dump_file

```
1:  function setup_sched_dump (void)
2:  sched_verbose ← sched_verbose_param
3:  if (sched_verbose_param = 0 and (dump_file=1) then
```

收稿日期：2020-08-24

作者简介：余晓江（1992- ），男，四川绵阳人，主要研究方向：编译器性能分析及优化.

基金项目：西华师范大学英才科研基金项目（课题编号：17YC163）.

```
4:  sched_verbose ← 1
5:  end if
6:  return sched_verbose
7:  if (sched_verbose_param ＞ 10 or (dump_file = 0) then
8:  sched_dump ← stderr
9:  end if
10:  sched_dump ← dump_file
11:  return sched_dump
12:  end function
```

算法1是在编译源程序时，在编译优化选项-O0下使用编译选项-fsched-verbose=n给sched_verbose赋值为n。优化后的DUMP算法根据n值选择性的输出指令调度信息。算法1判断sched_verbose_param的值＞10或者dump_file不存在的话，使用stderr输出；否则，使用dump_file进行输出。判断sched_verbose_param的值大于10或者dump_file不存在的话，使用stderr输出；否则，使用dump_file进行输出。

为了减小对编译器的影响，向GCC编译器添加编译选项-O2-dump控制编译器对指令调度数据流DUMP算法的选择，编译器默认设计sched-verbose的值为1，按照原算法的设计在编译时，优化后的算法在sched-verbose=1时会输出部分调度信息，所以需要设置sched-verbose的默认值为0，即没有任何输出信息。

2 实验分析

指令调度DUMP算法优化是通过对传递值做运算，让编译器在使用编译选项-O0时，能够使用-fsched-verbose=n来输出指令调度过程信息（见表1）。

表1 指令调度DUMP优化算法输出的数据流（部分）

```
basic block 5 from 1236 to 68 -- after reload
1:  | insn | prio |
2:  | 1236 |  11 | a0=[ s10 ]
    …
3:  |  534 |  1 | s1=s11
    …
4:  | 1095 |  4 | a5=[ sp+0x4 ]
5:  |  781 |  1 | a2=a5
6:  |  68 |  1 | pc={(a0>=0)?L74:pc}
    …
7:  Ready list (t =  16):
8:  16--> b  0: i 534 s1=s11  :alu
9:  Q-->Ready: insn 781: moving to ready with 1 stalls
10:  Advancing clock by 1 cycle[ s ]to 18
11:  Ready list after queue_to_ready:
                781:93:prio=1
12:  Ready list after ready_sort:
                781:93:prio=1
13:  Ready list (t =  18):  781:93:prio=1
14:  18--> b  0: i 781 a2=a5  :alu
    …
15:  total time = 19
16:  new head = 1236
17:  new tail = 68
18:  | insn | prio |
    …
19:  | 1095 |  4 | a5=[ sp+0x4 ]
20:  |  534 |  1 | s1=s11
21:  |  781 |  1 | a2=a5
    …
```

表1的指令调度数据流显示指令调度pass的执行结果，为后面分析和调度提供了可视化的输出信息。由于篇幅有限，只列出了部分有用信息。GCC编译器在RTL阶段会进行两次表调度，第一次调度为重载前的调度（before reload），第二次调度为重载后的调度（after reload）；1~6行列出当前基本块待调度的指令编号、指令依赖和指令内容；7~14行说明了具体调度过程，第8行和第14行意为一条指令解决所有依赖进行发射，即放入scheduled队列，前面的数字16和18表示当前的时钟周期；第10行说明指令insn 781（a2=a5）的依赖没有解决，需要等待一个周期再发射，但是，当前只有这一条指令可以被调度，所以，时钟周期为17的时候没有指令可以发射，就自动增加一个周期到18；第15行计算出当前基本块的总时钟周期；第16行和17行显示了指令调度后基本快的头部和尾部；第18行之后是调度后的指令顺序。

3 结束语

指令调度DUMP算法优化是GCC编译器对指令调度整个过程的可视化技术，在GCC编译器移植改进过程中作为辅助功能有着重要作用，尤其是在编译器指令调度优化研究过程中意义重大。通过对GCC编译器的DUMP算法改进和优化，进一步移植GCC到RISC-V等架构上奠定编译器基础。

参考文献：

[1] 蔡杰.GCC编译系统结构分析与后端移植实践[D].杭州：浙江大学,2004.

[2] 黄培镇,王忠仁,李胜坤.基于GUN工具的嵌入式软件编译体系的设计与实现[J].成都信息工程学院学报,2006(2):213-215.

[3] 高峰,吴海涛.简化GNU编译器套件抽象语法树的算法研究[J].上海师范大学学报(自然科学版),2018,47(4):479-482.

[责任编辑 胡艺文]