OpenJDK的RV32G支持

中科院软件所PLCT实验室 史宁宁

2022-8-25

目录

项目基本情况

缘起

- PLCT实验室在多个编译器/解释器领域，分别进行了基于RISC-V的支持开发工作。
- PLCT实验室对多个版本Java虚拟机的RISC-V支持进行了调研（详见后续内容）。
- 在调研OpenJDK领域的RISC-V支持的时候，了解到华为Bisheng JDK团队 已经完成了 OpenJDK 11的RV64G的支持，但是RV32G的还未有团队进行支持。
- 基于OpenJDK的RISC-V支持现状，中科院软件所PLCT实验室开始计划对OpenJDK 11的RV32G进行支持。
- PLCT实验室经过调研之后，从2021年1月份开始，基于华为Bisheng JDK团队开源的OpenJDK 11的RV64G，开始 OpenJDK 11的RV32G支持工作。

前期部分调研工作

1. **OpenJDK对于RISC-V的支持现状以及路线图**
2. **Maxine-VM对于RISC-V的支持进展调研与搭建测试**
3. **OpenJ9对于RISC-V的支持进展调研与搭建测试**
4. **RISCV64 DaCapo-9.12-bach-MR1基准测试**
5. **OpenJ9 RISCV64移植步骤大纲**
6. **交叉编译OpenJDK15 for RV64G（ZERO VM）**

团队

- 参与OpenJDK 11 for RV32G的团队成员在不断变化，最多的时候4个人。我为曾经参与过这个项目的成员做了一面照片墙。

史宁宁

张定立

章翔

曹贵

项目基本信息

- PLCT实验室目前所进行的RV32G工作，工作过程和工作产出都在github公开： https://github.com/openjdk-riscv/jdk11u
- 在移植过程中，PLCT实验室产出了几十篇技术文章和技术报告，这些内容分别公开在：
  1）https://github.com/openjdk-riscv/jdk11u/wiki
  2）https://www.zhihu.com/column/c_1287750038518161408
  3）https://space.bilibili.com/296494084/video?keyword=openjdk

项目进展

- 2022年1月11日，OpenJDK 11 for RV32G的模板解释器跑通了“Hello world”。
- 2022年2月25日， OpenJDK 11 for RV32G的模板解释器已经完成，开始移植OpenJDK 11 for RV32G的C2编译器。
- OpenJDK 11 for RV32G的当前情况：

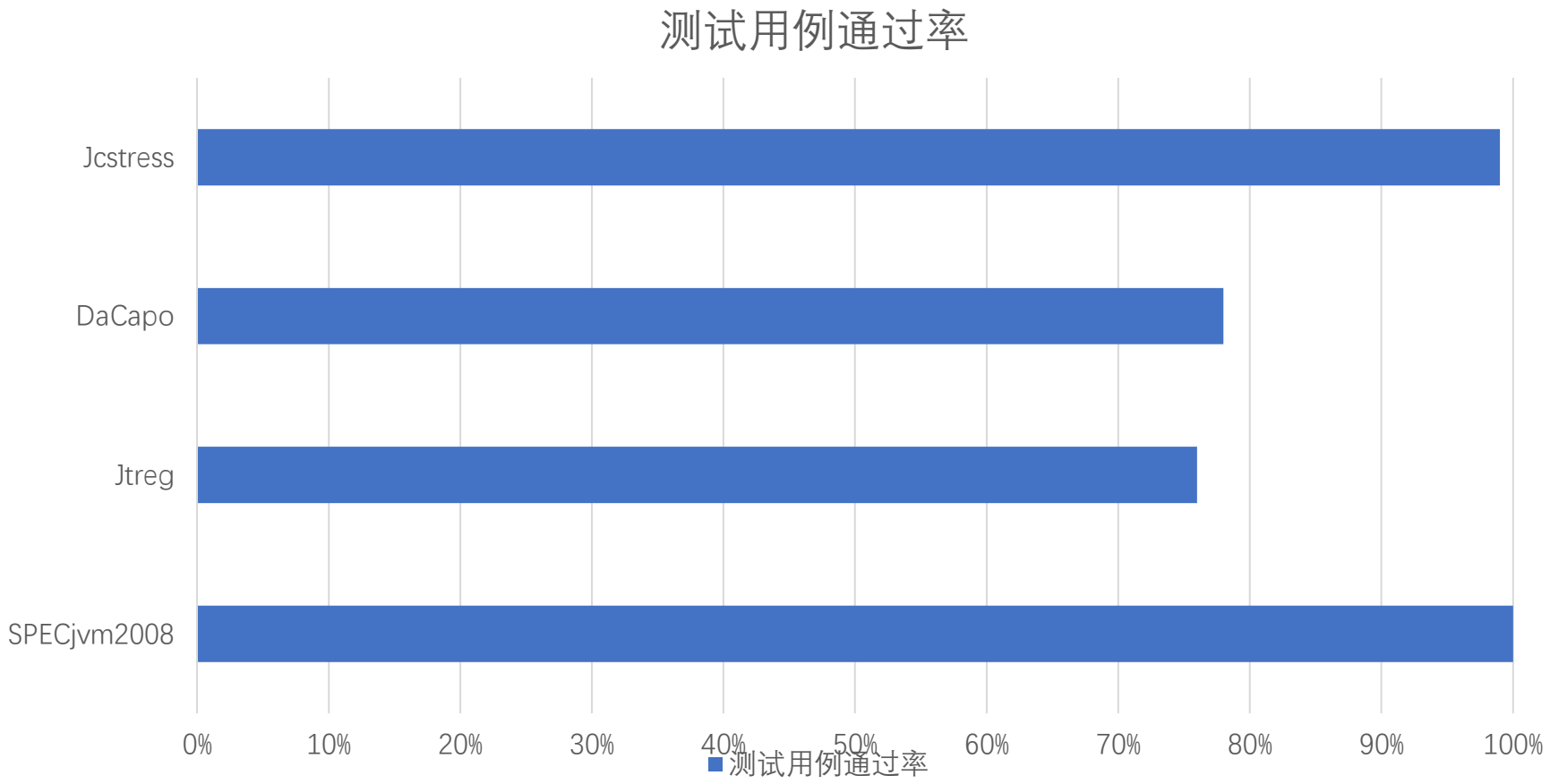
  1） OpenJDK 11 for RV32G的模板解释器可以支持绝大多数测试集合中的测试用例；

  2） OpenJDK 11 for RV32G的C2 仍然在debug中，编译基础类库还有一些问题；

  3） OpenJDK 11 for RV32G的C1 处于TODO状态。

模板解释器测试情况

- OpenJDK 11 for RV32G的模板解释器所支持的测试用例情况：



具体细节：
https://github.com/openjdk-riscv/jdk11u/issues/335

成　果

文档与技术报告

- 在移植过程中，PLCT实验室产出了几十篇技术文章和视频报告，这些文章都公开在：

github: https://github.com/openjdk-riscv/jdk11u/wiki

Zhihu: https://www.zhihu.com/column/c_1287750038518161408

B站：https://space.bilibili.com/296494084/video?keyword=openjdk

曹贵 - OpenJDK-TOS介绍及相关实现探索 - 20210929 - PLCT实验

307 10-4

JCK介绍 - OpenJDK - 陈家友 - 20201223 - PLCT实验室

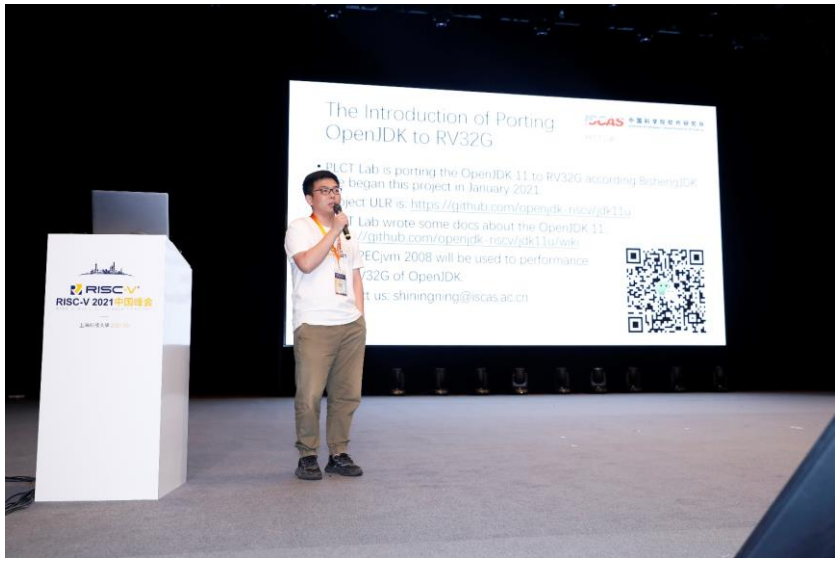
165 2020-12-23

OpenJ9构建测试及OpenJDK移植进展简介 - 张定立 - 20201107 -

380 2020-11-7

社区贡献——RISC-V 2021中国峰会

RISC-V 2021中国峰会
主会场poster&&海报展示



社区贡献（续）

RISC-V 2021中国峰会——PLCT开放日

《方舟、ART和OpenJDK的RISCV支持》

《关于「在 RISC-V 峰会召开前
将 OpenJDK 移植到 RV32GC 」结果却没有
赶上 Deadline 这件事》

社区贡献（续）

RISC-V Managed-Runtimes SIG 2021-8-24

ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences | 智能软件研究中心 Intelligent Software Research Center

The Introduction of Porting OpenJDK to RV32G

部署和验证

1. SPECjvm2008基准测试
2. 毕昇JDK 11 for RV64GC在D1开发板构建过程
3. 在 QEMU 上运行 RISC-V 32 位版本的 Linux
4. 在RISCV-yocto上运行 RV32G的OpenJDK11(ZERO)
5. HiFive Unleashed原生系统与Fedora写入及毕昇JDK的GDB调试
6. 毕昇JDK 11 for RICSV64构建及HiFive Unleashed测试
7. 在ubuntu i386中编译OpenJDK11

From: https://www.zhihu.com/column/c_1287750038518161408

遇到的主要问题

问题一：指令转换

加减乘除部分更新和总结修改规则如下：

addw->add
addiw->addi
subw->sub
mulw->mul
divw->div
divuw->divu

mul,mulh,mulhsu是32/64通用指令，不用修改。

sllw->sll
slliw -> slli
sraw->sra
sraiw -> srai
srlw->srl
srliw -> srli

load/store系列转换规则：

load 和 store 从64位到32位转换的规则更新

ld->lw
lwu->lw
sd->sw

lb，lbu，lh，lhu，lla，lui，lw
sb，sh，sw
fld，flw，fsd，fsw
lr.w， sc.w
这些都是RV32/64通用的指令。

lr.d通常表示为lr_d，需要更新为lr.w(lr_w)。
sc.d通常表示为sc_d，需要更新为sc.w(lc_w)。

fcvt.l.s 通常表示为 fcvt_l_s，需要更新为 fcvt.w.s(fcvt_w_s)，
fcvt.lu.s 通常表示为 fcvt_lu_s，需要更新为 fcvt.wu.s(fcvt_wu_s)，
fcvt.s.l 通常表示为 fcvt_s_l，需要更新为 fcvt.s.w(fcvt_s_w)，
fcvt.s.lu 通常表示为 fcvt_s_lu，需要更新为 fcvt.s.wu(fcvt_s_wu)，
fcvt.l.d 通常表示为 fcvt_l_d，需要更新为 fcvt.w.d(fcvt_w_d)，
fcvt.lu.d 通常表示为 fcvt_lu_d，需要更新为 fcvt.wu.d(fcvt_wu_d)，
fmv.x.d 通常表示为 fmv_x_d，需要更新为 fcvt.x.w(fcvt_x_w)，
fcvt.d.l 通常表示为 fcvt_d_l，需要更新为 fcvt.d.w(fcvt_d_w)，
fcvt.d.lu 通常表示为 fcvt_d_lu，需要更新为 fcvt.d.wu(fcvt_d_wu)，
fmv.d.x 通常表示为 fmv_d_x，需要更新为 fcvt.w.x(fcvt_w_x)。

From:https://github.com/openjdk-riscv/jdk11u/issues/39

问题二:64位字节的拼接和传递

long类型在RV32下依然为64位，需要用两个寄存器进行存取，并且传递时候也需要特别处理。

```
70    void InterpreterRuntime::SignatureHandlerGenerator::pass_long() {
71      const Address src(from(), Interpreter::local_offset_in_bytes(offset() + 1));
72  +   const Address high(from(), Interpreter::local_offset_in_bytes(offset()));
73
74      if (_num_int_args < Argument::n_int_register_parameters_c - 1) {
75        __ lw(g_INTArgReg[++_num_int_args], src);
76  +     __ lw(g_INTArgReg[++_num_int_args], high);
77      } else {
78        __ lw(x10, src);
79        __ sw(x10, Address(to(), _stack_offset));
```

问题三：偏移量

RV32G作为32位的架构，其寄存器、栈对齐等内容都与64位不同。很多代码所包含的计算，尤其是汇编指令所包含的计算，是以偏移作为一种计算手段，在这种情况之下，由于偏移量所导致的错误，就很难定位和修复。

尤其是在以RV64G代码为基础，进行RV32G移植的时候，这类问题就更加的隐秘。但是，只要找到几个典型，认识到这类问题的几种形式，那么同类别的问题解决起来就会快速很多。

问题四：调试问题

- 模板解释器相对于为每一个指令都写了一段实现对应功能的汇编代码，在JVM初始化时，汇编器会将汇编代码翻译成机器指令加载到内存中。如果这部分代码的偏移或者计算出错，比较难定位到具体出错的地方。
- 调试模版解释器时，输出的bytecode并不是代码直接完整翻译过来的，而是根据调用关系以及具体的值，去选择路径。不在路径上的bytecode是不会输出的。所以调试错误时候，跟踪bytecode的路径走向，是一个解决问题的思路。
- AD文件的调试（https://zhuanlan.zhihu.com/p/515274874）。

致 谢

致谢

- 华为Bisheng JDK团队 。
- 中科院软件所PLCT实验室：张定立、章翔、曹贵。
- 中科院软件所PLCT实验室实习生：陈家友。

Thanks!