

OpenJDK for RV32G的 移植与探索

中科院软件所PLCT实验室项目主管 史宁宁

2021-12-17

目录

- 项目背景
- 项目进展
- 项目过程中遇到的问题
- 2022年目标

目录

- 项目背景
- 项目进展
- 项目过程中遇到的问题
- 2022年目标

前期调研

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 \(ZERO VM\)](#)

OpenJDK/HotSpot的RV64支持

- 华为在2020年11月开源了BishengJDK 11，它基于OpenJDK 11对RV64G进行了实现，目前模版解释器和C1/C2都可以工作。
- BishengJDK 11目前可以在X86机器上进行交叉编译，并运行在QEMU RISCV64的用户模式和D1开发板，已经可以通过17000+的jreg测试用例。
- BishengJDK 11项目库位于gitee，地址为：
<https://gitee.com/openeuler/bishengjdk-11/tree/risc-v>

OpenJDK/HotSpot的RISC-V支持

[About RISC-V](#) ▾[Membership](#) ▾[RISC-V Exchange](#) ▾[Technical](#) ▾[News & Events](#) ▾[Community](#) ▾

This page is a collection of available software in the RISC-V ecosystem. This list is curated by the community – which includes you! Add software to the list by filing a pull request on the [GitHub repository](#). If you have any questions about this process, [contact us](#) for help.

Simulators	Object Toolchain	Debugging	C Compilers & Libraries	Bootloaders & Monitors	Hypervisors
OS & Kernels	Non-C Compilers/Runtimes	IDEs & SDKs	Security	Machine Learning & AI	Configuration
Verification Tools	Accelerated Libraries				

BishengJDK/HotSpot (Java Virtual Machine)	Upstream	GPLv2 with Classpath Exception	Yadong Wang
--	----------	--------------------------------	-------------

<https://riscv.org/exchange/software/#tab-1607480869844-2>

OpenJDK/HotSpot的RV32项目启动

- 基于PLCT实验室的愿景以及OpenJDK/HotSpot for RISC-V的支持情况，PLCT实验室开始启动OpenJDK/HotSpot for RV32G项目。
- OpenJDK/HotSpot for RV32G项目于2021年1月份正式启动，于2021年3月份走入正轨。
- 项目启动时候，项目团队有一名正式员工和一名实习生。
- 项目工作过程和工作产出都在github公开：
<https://github.com/openjdk-riscv/jdk11u>
- 项目过程中会产出技术文章，这些文章都公开在：
<https://github.com/openjdk-riscv/jdk11u/wiki>

目录

- 项目背景
- 项目进展
- 项目过程中遇到的问题
- 2022年目标

项目进展——团队建设

- 团队成员已经扩展为4个人的小队：史宁宁、张定立、章翔、曹贵。



项目进展——开发进度

- 目前模版解释器已经可以编译成功，正在调试JVM加载中的错误。

```
/home/shining/jdk11u/src/hotspot/os/linux/os_linux.cpp:1356:6: warning: #warning "SYS_clock_getres not defined for this platform, disabling thread_cpu_time" [-Wcpp]
1356 |         #warning "SYS_clock_getres not defined for this platform, disabling fast_thread_cpu_time"
      |         ^~~~~~
Finished building target 'default (exploded-image)' in configuration 'linux-riscv32-normal-core-slowdebug'
```

```
[847161] 220516 57 anewarray java/lang/Class
[847161] 220517 60 astore #6
[847161] 220518 62 iload_1
[847161] 220519 63 ifle 78
[847161] 220520 66 aload_0
[847161] 220521 67 getfield 4 <java/lang/invoke/MethodType.ptypes/[Ljava/lang/Class;>
[847161] 220522 70 iconst_0
[847161] 220523 71 aload #6
[847161] 220524 73 iconst_0
[847161] 220525 74 iload_1
[847161] 220526 75 invokestatic 33 <java/lang/System.arraycopy(Ljava/lang/Object;ILjava/lang/Object;II)V>
# To suppress the following error report, specify this argument
# after -XX: or in .hotspotrc: SuppressErrorAt=/copy_linux_riscv32.inline.hpp:90
#
# A fatal error has been detected by the Java Runtime Environment:
#
# Internal Error (/home/shining/jdk11u/src/hotspot/os_cpu/linux_riscv32/copy_linux_riscv32.inline.hpp:90), pid=847159, tid=847161
# assert(BytesPerLong == BytesPerOop) failed: jlongs and oops must be the same size.
#
# JRE version: OpenJDK Runtime Environment (11.0.9) (slowdebug build 11.0.9-internal+0-adhoc.shining.jdk11u)
# Java VM: OpenJDK Core VM (slowdebug 11.0.9-internal+0-adhoc.shining.jdk11u, interpreted mode, serial gc, linux-riscv32)
# No core dump will be written. Core dumps have been disabled. To enable core dumping, try "ulimit -c unlimited" before starting Java again
#
# An error report file with more information is saved as:
# /home/shining/jdk11u/build/linux-riscv32-normal-core-slowdebug/jdk/bin/hs_err_pid847159.log
#
# If you would like to submit a bug report, please visit:
# https://bugreport.java.com/bugreport/crash.jsp
#
Current thread is 847161
Dumping core ...
Aborted
```

项目进展——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>

专栏

Java on RISC-V

Java on RISC-V

让RISC-V生态可以用上工业级的Java应用



Bamboo · 26 篇内容

推荐文章



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

项目进展——社区贡献

RISC-V 2021中国峰会
主会场poster



项目进展——社区贡献（续）

RISC-V 2021中国峰会 海报展示



项目进展——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》

项目进展——社区贡献（续）

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

The Introduction of Porting
OpenJDK to RV32G

项目进展——部署和验证

1. [交叉编译OpenJDK11 for RV32G \(ZERO VM\)](#)
2. [萌新的交叉编译OpenJDK11 for RV32G的踩坑之路](#)
3. [SPECjvm2008基准测试](#)
4. [毕昇JDK 11 for RV64GC在D1开发板构建过程](#)
5. [在 QEMU 上运行 RISC-V 32 位版本的 Linux](#)
6. [在RISCV-yocto上运行 RV32G的OpenJDK11\(ZERO\)](#)
7. [HiFive Unleashed原生系统与Fedora写入及毕昇JDK的GDB调试](#)
8. [毕昇JDK 11 for RICSV64构建及HiFive Unleashed测试](#)
9. [在ubuntu i386中编译OpenJDK11](#)

目录

- 项目背景
- 项目进展
- 项目过程中遇到的问题
- 2022年目标

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->slli

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
lduw->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(sc_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)。

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。

目录

- 项目背景
- 项目进展
- 项目过程中遇到的问题
- 2022年目标

2022年目标

技术目标：解释器正常运行；C1、C2正常运行；开始将RV32G向OpenJDK for RISC-V官方库进行提交。

团队扩充：建设人才梯队，招聘和培养实习生，从中选拔优秀人才进入团队。

技术积累：编写一本OpenJDK for RISC-V的技术书籍。

Thanks~