riscv-gnu-toolchain简介与开发入门

riscv-gnu-toolchain 前言

RISC-V GNU toolchain 从字面拆开来看即是 RISC-V + GNU + toolchain，我们分别来介绍一些这几个关键字

RISC-V指RISC-V指令集架构，目前它包含的指令可分为四类：

1. 标准非特权指令集 —— g（imafd）通用指令扩展
2. 标准特权指令集 —— 特权寄存器
3. 专用扩展指令集 —— 位操作扩展（B），密码学扩展（K），向量扩展(V)等
4. 厂商自定义指令集 —— 面向C90*系列芯片的xthead扩展，OpenHW的CV扩展等

RISC-V指令集手册：https://github.com/riscv/riscv-isa-manual

riscv-gnu-toolchain 前言

GNU工程是自由软件活动的集大成者，包含了丰富的软件工具，例如 gcc, gas, gdb 等等，遵循 GPLv3 开源协议，自由软件的用户拥有四项基本自由：

(0)自由运行软件

(1)自由学习和修改软件源代码

(2)自由再发布软件拷贝

(3)自由发布修改后的软件版本。

自由软件的介绍：自由软件现在更加重要

GNU软件列表

riscv-gnu-toolchain 前言

Toolchain 在这里更为准确的说明是 compiler toolchain，由于一个高级程序从编译到运行期间往往需要多种工具的合作，我们把这些使用到的工具集合统称为 toolchain：


```c
#include<stdio.h>
int main() {
    printf("Hello world!");
    return 0;
}
```

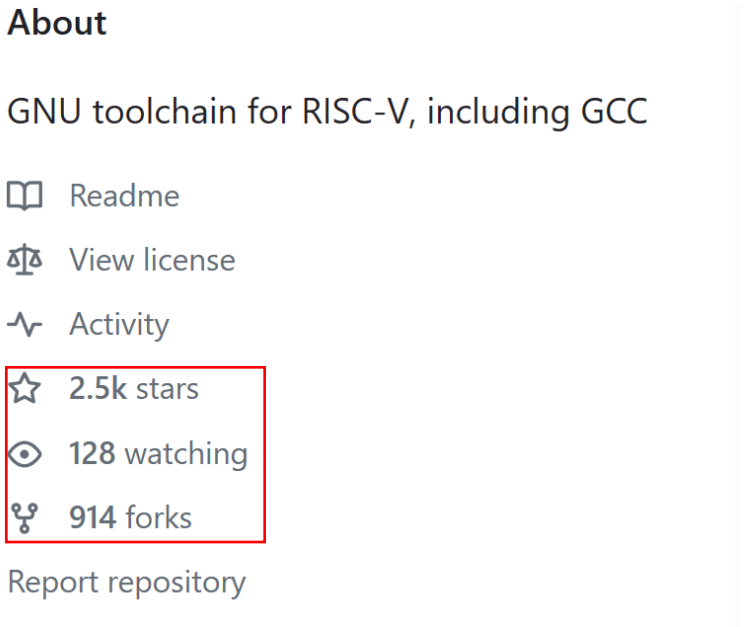
```
.LC0:
        .string "Hello world!"
main:
        addi    sp,sp,-16
        sd      ra,8(sp)
        sd      s0,0(sp)
        addi    s0,sp,16
        lui     a5,%hi(.LC0)
        addi    a0,a5,%lo(.LC0)
        call    printf
        li      a5,0
        mv      a0,a5
        ld      ra,8(sp)
        ld      s0,0(sp)
        addi    sp,sp,16
        jr      ra
```

gdb

riscv-gnu-toolchain简介——

riscv-gnu-toolchain 是由 RVI 管理的官方工具链仓库，包含支持 RISC-V 指令集架构的一些列工具（编译器，汇编器，链接器，调试器，模拟器，c库，测试工具）

仓库地址：https://github.com/riscv-collab/riscv-gnu-toolchain



riscv-gnu-toolchain简介——

这里我们重点介绍一下 riscv-gnu-toolchain 中一些常用的模块:

Binutils(bin-utils)是一系列二进制工具的集合，包含链接器(linker), 汇编器(assembler),目标文件查看工具(objdump),性能分析工具(gprof)等

Binutils官网：https://sourceware.org/binutils

仓库地址：https://sourceware.org/git/binutils-gdb.git

riscv-gnu-toolchain简介——

GCC(GNU compiler collection)是使用最为广泛的编译器之一，这里的gcc仓库支持多种前后端，是用来生成特定目标编译器的一个编译器框架

gcc官网：https://gcc.gnu.org

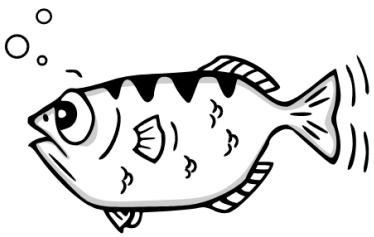
仓库地址：https://gcc.gnu.org/git/gcc.git

riscv-gnu-toolchain简介——

GDB是一个常用的二进制调试工具，它可以帮助了解程序正在发生什么（往往程序运行崩溃了你就会想到它。。）

gdb官网：https://sourceware.org/gdb

仓库地址：https://sourceware.org/git/binutils-gdb.git (和binutils在同一个仓库)



riscv-gnu-toolchain简介——

Glibc是标准c库，提供了 linux 中最核心的库文件支持，例如 ISO C11, POSIX, BSD 等标准制定的所有库函数，以及常用函数的 API 支持（ printf, fopen/close, read/write ）

glibc官网：https://www.gnu.org/software/libc/libc.html

仓库地址：https://sourceware.org/git/glibc.git

riscv-gnu-toolchain的构建——

首先阅读README，简单了解仓库，接着根据指示配置依赖环境，设置构建参数，然后进行构建，
所有构建参数在configure文件中保存，利用configure参数进行配置,构建过程中如果报错，可以根据错误信息尝试修正，或者记录log寻求帮助

```
It was created by riscv-toolchain configure 1.0, which was
generated by GNU Autoconf 2.69.  Invocation command line was

  $ ./configure --prefix=/opt/riscv/ --with-arch=rv64gc

## --------- ##
## Platform. ##
## --------- ##

hostname = plct
uname -m = x86_64
uname -r = 5.4.0-73-generic
uname -s = Linux
uname -v = #82-Ubuntu SMP Wed Apr 14 17:39:42 UTC 2021
```

riscv-gnu-toolchain的使用——

构建完成后，我们就可以直接使用生成的一系列二进制工具了，使用过程中如果发现错误或者有任何疑问，可以在 riscv-gnu-toolchain 的 issue 中进行提问，社区会及时响应解答

```
root@plct:/opt/riscv/bin# ls
riscv64-unknown-linux-gnu-addr2line    riscv64-unknown-linux-gnu-gcc-ranlib     riscv64-unknown-linux-gnu-nm
riscv64-unknown-linux-gnu-ar           riscv64-unknown-linux-gnu-gcov           riscv64-unknown-linux-gnu-objcopy
riscv64-unknown-linux-gnu-as           riscv64-unknown-linux-gnu-gcov-dump      riscv64-unknown-linux-gnu-objdump
riscv64-unknown-linux-gnu-c++filt      riscv64-unknown-linux-gnu-gcov-tool      riscv64-unknown-linux-gnu-ranlib
riscv64-unknown-linux-gnu-cpp          riscv64-unknown-linux-gnu-gdb            riscv64-unknown-linux-gnu-readelf
riscv64-unknown-linux-gnu-elfedit      riscv64-unknown-linux-gnu-gdb-add-index  riscv64-unknown-linux-gnu-run
riscv64-unknown-linux-gnu-gcc          riscv64-unknown-linux-gnu-gprof          riscv64-unknown-linux-gnu-size
riscv64-unknown-linux-gnu-gcc-10.4.0   riscv64-unknown-linux-gnu-ld             riscv64-unknown-linux-gnu-strings
riscv64-unknown-linux-gnu-gcc-ar       riscv64-unknown-linux-gnu-ld.bfd         riscv64-unknown-linux-gnu-strip
riscv64-unknown-linux-gnu-gcc-nm       riscv64-unknown-linux-gnu-lto-dump
```

开发准备——

开发使用C语言，所以需要熟悉C语言的语法和使用

Linux环境：本地，服务器都可以

注意代码规范：GNU Coding Standards 注释，缩进风格

学会灵活使用IDE，git工具

社区公约：https://gcc.gnu.org/conduct.html
如何向社区提交代码：https://github.com/plctlab/riscv-gcc/wiki

开发示例——如何添加一条RISC-V汇编指令

这里我们以一条RISC-V单精度浮点加法汇编指令 fadd.s 为例，看看它在工具链中是如何添加的

1. 首先查询RISC-V指令集手册，了解熟悉指令定义
2. 在binutils中添加指令
3. 编写指令测试用例
4. 重新构建工具链，测试是否能够正确识别指令

开发示例——如何添加一条RISC-V汇编指令

GCC中的fadd.s定义：

```
(define_insn "add<mode>3"
  [(set (match_operand:ANYF            0 "register_operand" "=f")
        (plus:ANYF (match_operand:ANYF 1 "register_operand" " f")
                   (match_operand:ANYF 2 "register_operand" " f")))]
  "TARGET_HARD_FLOAT || TARGET_ZFINX"
  "fadd.<fmt>\t%0,%1,%2"
  [(set_attr "type" "fadd")
   (set_attr "mode" "<UNITMODE>")])
```

gcc开发手册：https://gcc.gnu.org/onlinedocs/gccint/index.html#toc-Machine-Descriptions

Q&A

Q：如何快速入门？

A：多看其他人提交的代码，尤其是过去的一些提交记录，大部分代码支持都非常相似，可以学习很多

Q：代码太多，看不懂怎么办？

A：关键在于对框架的理解，有时代码实现的细节较为复杂，不必纠结其中

Q：现在入坑选GCC还是LLVM？

A：两面各有优劣，GCC是GNU的指定编译器，随着Linux还会活跃很长时间
LLVM社区活跃程度高，教程丰富，更适合新人