RISCV 工具链数据报第二期：

CodeSize ARM vs RISCV 和 RISCV IDE 评测

本期 RISCV 工具链数据报分为两部分，第一部分基于 GNU toolchain 和 CSiBE 对 ARM 和 RISCV 的 codesize 进行测试和对比；第二部分，我们将对目前公开网络上可下载的一款 RISCV toolchain 进行 codesize 的评测。

Part1 ARM 和 RISCV 的 codesize 测试和对比

延续第一期的方法，使用 CSiBE（http://szeged.github.io/csibe/） 的 CSiBE-V2.1.1 进行测试，结果去掉了两个编译不过的 case（jepg-6b 和 libmspack）和两个 codesize 不受编译影响的 case（lwip-0.5.3.preproc 和 ttt-0.10.1.preproc）。

评测使用的 ARM toolchain 来自 https://www.linaro.org/downloads/ 页面的 Latest Linux Targeted Binary Toolchain Releases 的 Binaries，如下图所示：

Latest Linux Targeted Binary Toolchain Releases

arm-linux-gnueabihf	*32-bit Armv7 Cortex-A, hard-float, little-endian*	Release-Notes	Binaries	Source
armv8l-linux-gnueabihf	*32-bit Armv8 Cortex-A, hard-float, little-endian*	Release-Notes	Binaries	Source
aarch64-linux-gnu	*64-bit Armv8 Cortex-A, little-endian*	Release-Notes	Binaries	Source

解压后目录列表如下：

gcc-linaro-7.4.1-2019.02-x86_64_armv8l-linux-gnueabihf
gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabihf
gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu

评测使用的 RISCV32 和 RISCV64 的 toolchain，构建信息如下：

源码位置	https://github.com/riscv/riscv-gnu-toolchain.git
版本	commit id: 2c037e6 GCC9.2.0 Binutils2.32
构建方式	Newlib cross-compiler （参考 README.md 中的 RISCV32 和 RISCV64 的构建步骤）

测试对照组和编译选项如下表：

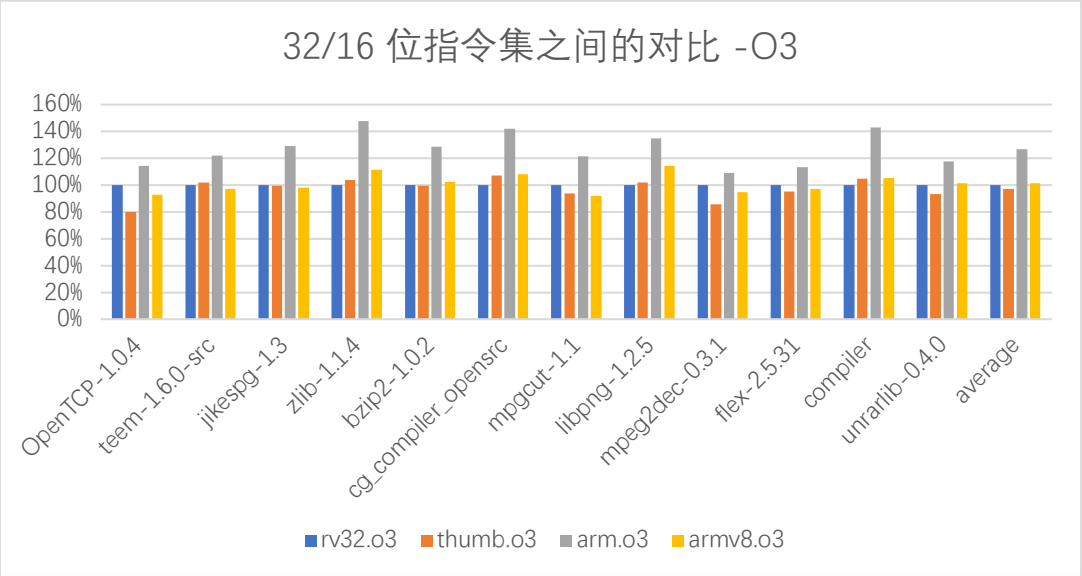
	工具链名称	O3	O3Os
RV32	riscv32-unknown-elf-gcc/g++	-O3 -march=rv32imafdc 【rv32.o3】	-O3 -Os -march=rv32imafdc 【rv32.o3os】
RV64	riscv64-unknown-elf-gcc/g++	-O3 -march=rv64imafdc 【rv64.o3】	-O3 -Os -march=rv64imafdc 【rv64.o3os】
armv7a-marm	arm-linux-gnueabihf-gcc/g++	-O3 -mfpu=vfpv3-d16 --machine=arm -mtune=cortex-a9 -march=armv7-a 【arm.o3】	-O3 -Os -mfpu=vfpv3-d16 --machine=arm -mtune=cortex-a9 -march=armv7-a 【arm.o3os】
armv7a-mthumb	arm-linux-gnueabihf-gcc/g++	-O3 -mfpu=vfpv3-d16 --machine=thumb -mtune=cortex-a9 -march=armv7-a 【thumb.o3】	-O3 -Os -mfpu=vfpv3-d16 --machine=thumb -mtune=cortex-a9 -march=armv7-a 【thumb.o3os】
armv8l-marm	armv8l-linux-gnueabihf	-O3 -mcpu=cortex-a53 【armv8.o3】	-O3 -Os -mcpu=cortex-a53 【armv8.o3os】
aarch64	aarch64-linux-gnu	-O3 -mcpu=cortex-a53 【a64.o3】	-O3 -Os -mcpu=cortex-a53 【a64.o3os】

codesize 的测试结果如下：

1. 下表显示了所有 12 个测试用例平均每个文件的 text 段字节数绝对值

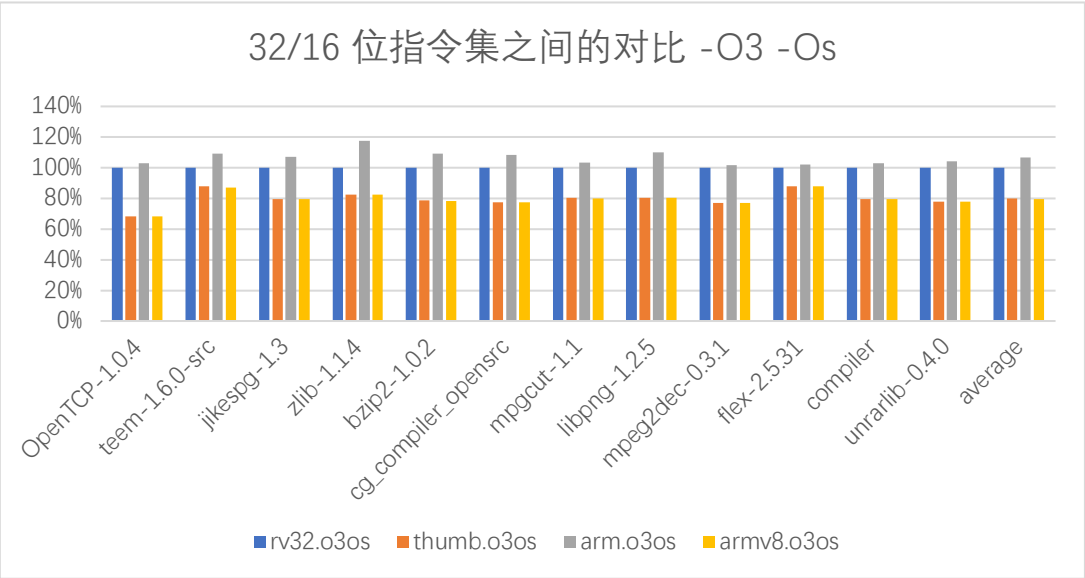
	arm.o3os	arm.o3	armv8.o3os	armv8.o3	thumb.o3os	thumb.o3	a64.o3os	a64.o3	rv32.o3os	rv32 o3	rv64.o3os	rv64.o3
OpenTCP-1.0.4	1491	2675	986	2168	987	1871	1539	3138	1448	2340	1478	2427
teem-1.6.0-src	2802	3600	2228	2865	2254	3004	2670	3529	2561	2947	2585	2982
jikespg-1.3	13126	19097	9739	14503	9739	14727	13297	18838	12233	14805	12536	15552
zlib-1.1.4	2418	4273	1701	3225	1698	3008	2507	4550	2061	2895	2227	3164
bzip2-1.0.2	6549	11073	4708	8813	4713	8557	6471	11229	6002	8593	6238	9354
cg_compiler_opensrc	5561	10213	3970	7789	3968	7703	5623	10507	5124	7191	5282	7409
mpgcut-1.1	8804	29004	6804	21948	6872	22336	8760	28284	8532	23878	8598	24282
libpng-1.2.5	5574	8845	4073	7497	4075	6680	5561	10019	5065	6562	5147	6742
mpeg2dec-0.3.1	1576	2206	1193	1919	1193	1735	1637	2510	1550	2026	1623	2142
flex-2.5.31	9719	12865	8368	11011	8365	10799	9626	13192	9521	11328	9829	11732
compiler	4044	10566	3125	7792	3127	7741	4194	10949	3927	7402	4186	7849
unrarlib-0.4.0	4731	9907	3542	8517	3535	7845	4801	10181	4546	8417	4776	8635

2. 下图显示了所有 32/16 位指令集的 O3 选项，相对于 RISCV32 O3 选项的比例：



平均情况下，上图所示四个对照组的相对比例是：100%，97%，127%，101%，可以看到，thumb.o3 和 armv8.o3 对应的 codesize 与 rv32.o3 的 codesize 差别不大，但 arm.o3 选项下，codesize 相比 rv32.o3 大 27%。

3. 下图显示了所有 32/16 位指令集的 O3Os 选项，相对于 RISCV32 O3Os 选项的比例：



平均情况下，上图所示四个对照组的相对比例是：100%，80%，107%，80%，可以看到， thumb.o3os 和 armv8.o3os 对应的 codesize 相对

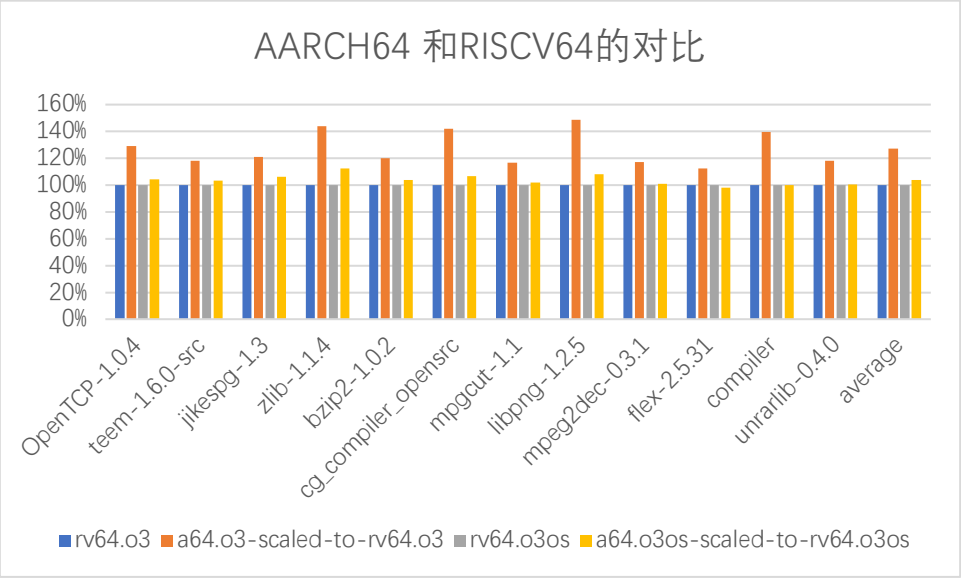
rv32.o3os 的 codesize 有较大优势（前者仅为后者的 80%大小），但 arm.o3os 的 codesize 相比 rv32.o3os 的 codesize 大 7%。

为了对数据进行解释，我们查看了测试使用的 ARM 编译器 build configure 信息中与 ISA/ABI 相关的内容，信息如下：

工具链名称	ISA/ABI info
gcc-linaro-7.4.1-2019.02-x86_64_armv8l-linux-gnueabihf	--with-float=hard --with-fpu=neon-fp-armv8 --with-mode=thumb --with-arch=armv8-a
gcc-linaro-7.4.1-2019.02-x86_64_arm-linux-gnueabihf	--with-float=hard --with-fpu=vfpv3-d16 --with-mode=thumb --with-tune=cortex-a9 --with-arch=armv7-a
gcc-linaro-7.4.1-2019.02-x86_64_aarch64-linux-gnu	--with-arch=armv8-a

可以看到，由于编译器有默认的 ISA 配置，因此，在测试对照组中的 thumb.o3/thumb.o3os/armv8.o3/armv8.o3os 都默认开启了-mthumb，这可能是 codesize 上比较接近的原因之一；当在 arm-linux-gnueabihf 下显式开启-marm 选项时，codesize 相对-mthumb 会有明显增加(约 30%)。

4. 下图显示了 AARCH64 和 RISCV64 位指令集下 codesize 的对比，图中分别以 rv64.o3 和 rv64.o3os 为参照（100%），将 AARCH64 的对应值进行了 scale。在-O3 选项下，AARCH64 的 codesize 平均比 RISCV64 大 27%，在-O3 -Os 选项下，AARCH64 的 codesize 平均比 RISCV64 大 3%。



PART2 RISCV IDE 的评测

卡姆派乐公司的 RISCV IDE 是目前网络上唯一的一款免费的、包含 RISCV toolchain 的 IDE(https://code.ihub.org.cn/projects/790/repository/riscv-ide)。我们从 http://bggit.ihub.org.cn/p47082315/riscv-ide.git

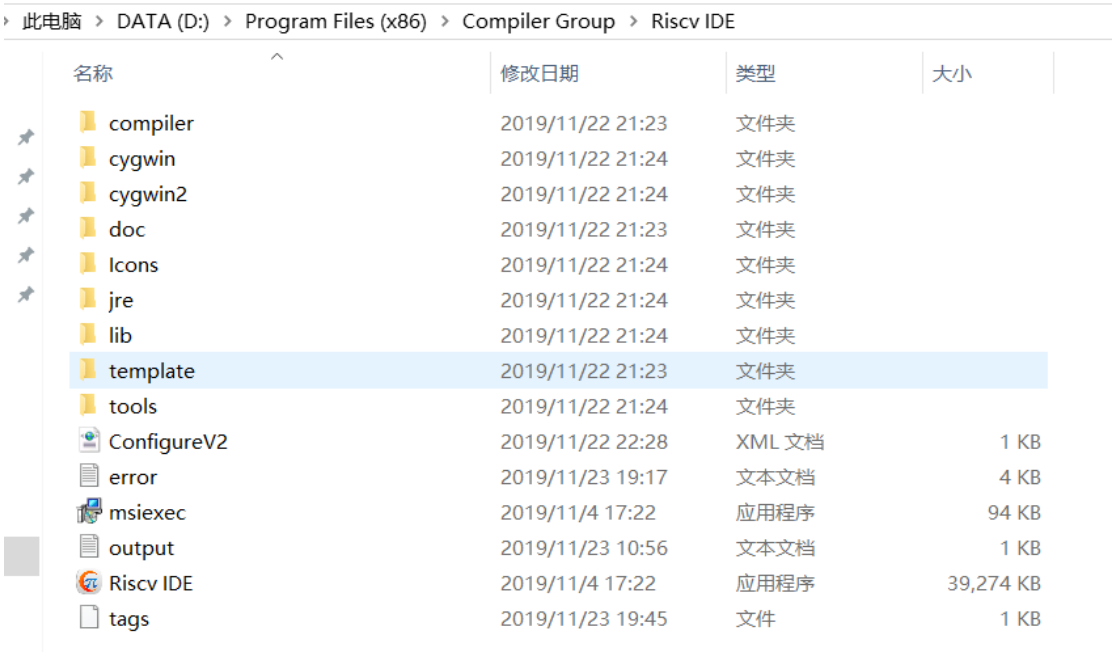
下载了最新版本的 windows msi 安装包（ commit-id 8360abac7d3b98d25debdb8373cbb0eff6d0077a），这个 IDE 的介绍如下图:

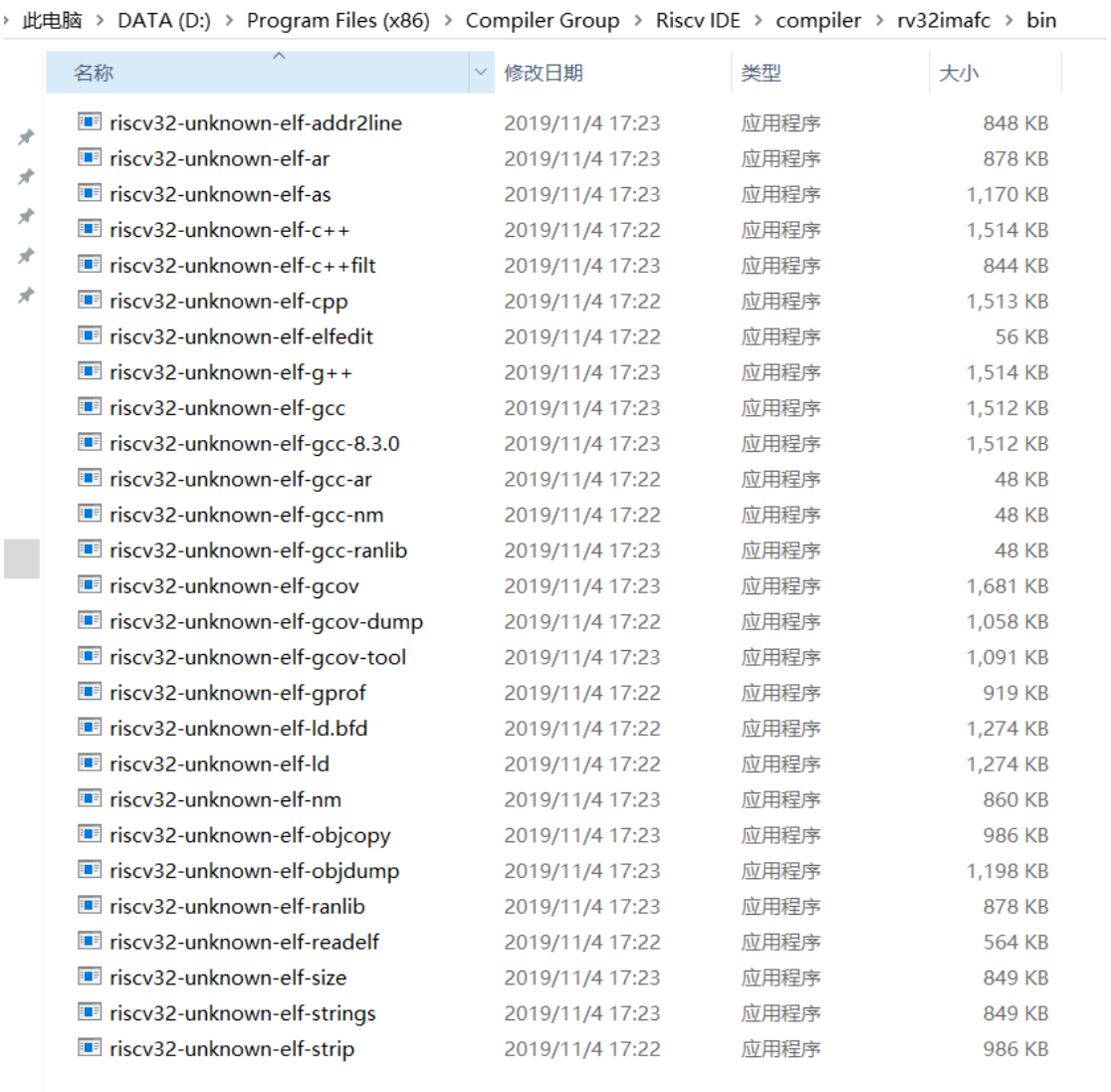
首款国产RISCV集成开发环境，基于图形化界面，一键式安装，主要特点：
l 编译器支持代码长度优化：二进制代码长度比公版优化10-30%；
l 启动速度快、功能强大、界面简洁清晰；
l 集成SPIKE模拟器；
l 提供中英文两个版本；
l 功能可定制：可以根据用户体系结构的需求，提供编译器、调试器等定制服务；
l 支持Windows和Linux操作系统。
l 支持GD32VF103-START

RISCV IDE | 项目　卡姆派乐

创建时间：2019-10-28 16:12

安装完成后，我们先对安装目录和用户手册进行查看，可以发现内嵌的 toolchain 是 Cygwin host 方式编译的 baremetal riscv32-unknown-elf-gcc，核心目录结构见下面三个图：

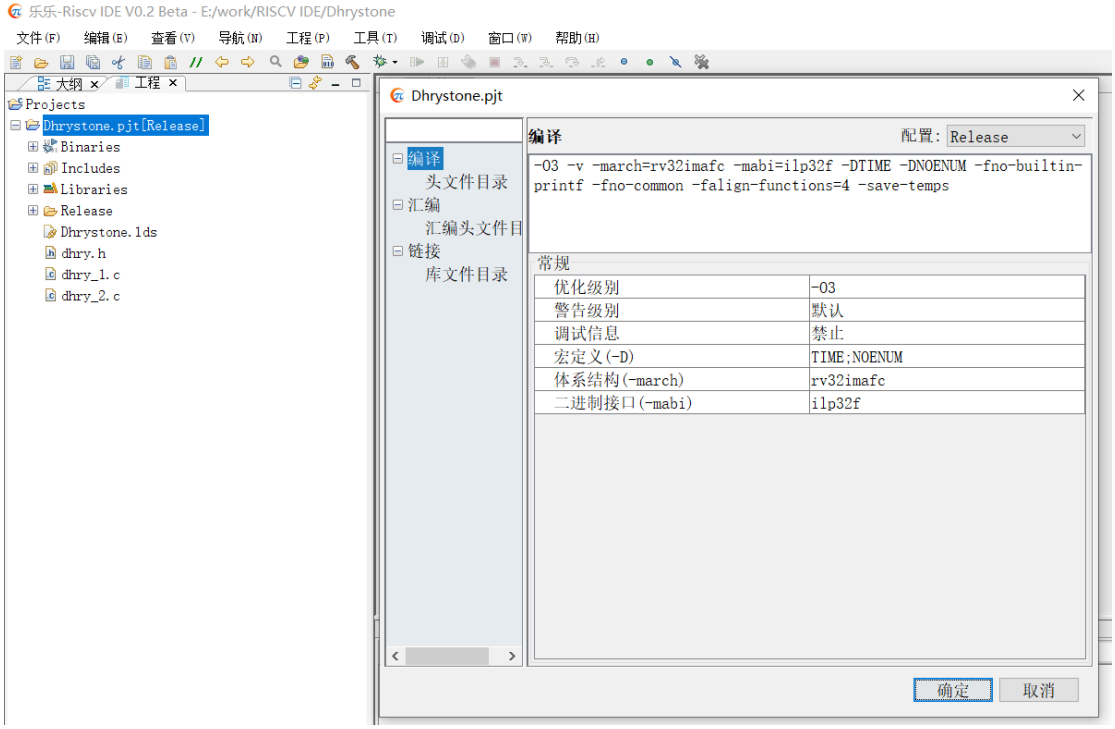




通过阅读其用户文档，获知可以建立两类工程，分别是 spike 和 gd32vf103c-start。我们首先建立了一个 spike 工程，将

https://github.com/sifive/benchmark-dhrystone.git 下载的 dhrystone 源

代码的 .c 和.h 头文件加入该工程，然后添加 dhrystone 相关的编译选项，添加方式和选项如下图：



点击确定按钮后，再点击编译按钮，程序能够成功编译和链接，第一步评测完成。

于此同时，通过-v 选项的 verbose 输出，可以获得该 IDE 中内嵌编译器的各个组件的版本和 configure 选项信息，如下：

```
Target: riscv32-unknown-elf
Configured with: /cygdrive/e/src/riscv-gnu-toolchain/build_rv32imafc/../riscv-gcc/configure --target=riscv32-unknown-elf --prefix=/opt/rv32imafc --disable-shared --disable-threads --disable-tls --enable-languages=c,c++ --with-system-zlib --with-newlib --with-sysroot=/opt/rv32imafc/riscv32-unknown-elf --disable-libmudflap --disable-libssp --disable-libquadmath --disable-libgomp --disable-nls --src=../../riscv-gcc --enable-checking=yes --disable-multilib --with-abi=ilp32f --with-arch=rv32imafc --with-tune=rocket 'CFLAGS_FOR_TARGET=-Os -mcmodel=medlow' 'CXXFLAGS_FOR_TARGET=-Os -mcmodel=medlow'
Thread model: single
gcc version 8.3.0 (GCC)
GNU assembler version 2.32 (riscv32-unknown-elf) using BFD version (GNU Binutils) 2.32
```

上述信息说明，RISCV IDE 中内嵌的 GCC 基础版本是 8.3.0，binutils 基础版本是 2.32。我们将从开源社区下载该基础版本的 GNU 官方源码，然后构建一套 RISCV32 baremetal 工具链，来进行对比测试。具体方法如下：

1. 使用 https://github.com/riscv/riscv-gnu-toolchain.git 提供的构建脚本和目录，将 riscv-gcc 目录（当前是 9.2.0 版本）的代码替换成从 https://ftp.gnu.org/gnu/gcc/gcc-8.3.0/ 下载的 gcc-8.3.0.tgz，构建一个 x86_64 Linux host 的 baremetal RISCV32 交叉工具链（注：riscv-binutils-gdb 已经是 2.32 版本此处没有替换）；

2.使用相同的编译选项来对比 RISCV IDE 内嵌的 toolchain 与 GNU 官方相同基础版本的 codesize 指标。

评测的目标程序方面，除了前述步骤使用的 dhrystone，还有从 https://github.com/eembc/coremark.git 下载的 coremark，以及 RISCV IDE 中提供的 gd32vf103c 的 Running_Led 工程。评测方法是，分别在 IDE 和 x86_64 Linux host 上，用同样的编译、链接选项来构建上述测试用例，最后用 size 命令来获得目标文件或可执行文件的 text 段大小，进而对比数据。

具体的编译选项如下：

表 1: dhrystone 编译选项（不做链接）

dhry.o2	-O2 -DTIME -DNOENUM -fno-builtin-printf -fno-common -falign-functions=4 -march=rv32imafc -mabi=ilp32f
dhry.o2os	-O2 -Os -DTIME -DNOENUM -fno-builtin-printf -fno-common -falign-functions=4 -march=rv32imafc -mabi=ilp32f
dhry.o3	-O3 -DTIME -DNOENUM -fno-builtin-printf -fno-common -falign-functions=4 -march=rv32imafc -mabi=ilp32f
dhry.o3os	-O3 -Os -DTIME -DNOENUM -fno-builtin-printf -fno-common -falign-functions=4 -march=rv32imafc -mabi=ilp32f

表 2：Coremark 编译选项（不做链接）

cm.o2	-O2 -DPERFORMANCE_RUN=1 -DITERATIONS=100 -DFLAGS_STR=0 -I${PROJECT_SOURCE_DIR} -I${PROJECT_SOURCE_DIR}/linux/ -march=rv32imafc -mabi=ilp32f
cm.o2os	-O2 -Os -DPERFORMANCE_RUN=1 -DITERATIONS=100 -DFLAGS_STR=0 -I${PROJECT_SOURCE_DIR} -I${PROJECT_SOURCE_DIR}/linux/ -march=rv32imafc -mabi=ilp32f
cm.o3	-O3 -DPERFORMANCE_RUN=1 -DITERATIONS=100 -DFLAGS_STR=0 -I${PROJECT_SOURCE_DIR} -I${PROJECT_SOURCE_DIR}/linux/ -march=rv32imafc -mabi=ilp32f
cm.o3os	-O3 -Os -DPERFORMANCE_RUN=1 -DITERATIONS=100 -DFLAGS_STR=0 -I${PROJECT_SOURCE_DIR} -I${PROJECT_SOURCE_DIR}/linux/ -march=rv32imafc -mabi=ilp32f

表 3: gd32vf103c-Running_Led 的编译选项（均采用 RISCV IDE 中的默认选项）

编译选项	-march=rv32imac -mabi=ilp32 -mcmodel=medlow -msmall-data-limit=8 -O0 -g -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -fno-common
链接选项	--start-group -lstdc++_nano -lm -lg_nano -lgcc -lc_nano -lperipherals --end-group
移植到 Linux 上的完整编译命令	riscv32-unknown-elf-gcc -march=rv32imac -mabi=ilp32 -mcmodel=medlow -msmall-data-limit=8 -O0 -g -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -fno-common -I./Application/ -I"./Utilities/LCD_common" -I"./Peripherals/Include" -I"./Peripherals" -I"./RISCV/drivers" -I"./Utilities" -I"./Application" -I"./" ./Application/main.c -c -o ./Application/main.o riscv32-unknown-elf-gcc -march=rv32imac -mabi=ilp32 -mcmodel=medlow -msmall-data-limit=8 -O0 -g -fmessage-length=0 -fsigned-char -ffunction-sections -fdata-sections -fno-common -I./RISCV/env_Eclipse -I"../Utilities/LCD_common" -I"./Peripherals/Include" -I"./Peripherals" -I"./RISCV/drivers" -I"./Utilities" -I"./Application" -I"../" -c "./RISCV/env_Eclipse/your_printf.c" -o "RISCV/env_Eclipse/your_printf.o" riscv32-unknown-elf-ld "Application/main.o" "RISCV/env_Eclipse/your_printf.o" -melf32lriscv --start-group -lstdc++_nano -lm -lg_nano -lgcc -lc_nano -lperipherals --end-group --gc-sections -o"Running_Led.out" -T"./Running_Led.lds" -L/opt/rv32imac/lib1/rv32imac/ilp32/ -L/opt/rv32imac/lib2/rv32imac/ilp32/ -L. -L. size --format=berkeley ./Running_Led.out

测试结果如下：

表 4　dhrystone codesize 数据

	RISCV IDE				GNU 官方			
	o2	o2os	o3	o3os	o2	o2os	o3	o3os
dhry_1.c	3831	3439	3851	3439	3831	3439	3851	3439
dhry_2.c	256	242	236	242	256	242	236	242

表 5 coremark codesize 数据

	RISCV IDE				GNU 官方			
	o2	o2os	o3	o3os	o2	o2os	o3	o3os
core_list_join.c	1472	1328	2800	1328	1472	1328	2800	1328
core_main.c	3172	3078	3320	3078	3172	3078	3320	3078
core_matrix.c	1502	860	1798	860	1502	860	1798	860
core_portme.c	158	160	158	160	158	160	158	160
core_state.c	1367	1045	1499	1045	1367	1045	1499	1045
core_util.c	398	316	2046	316	398	316	2046	316

表 6　gd32vf103c-Running_Led CodeSize 对比

	RISCV IDE	GNU 官方
Running_Led.out	8068	8068

结论：对比表格中数据，在此次所使用的评测方法、编译和链接选项下，RISCV IDE 和官方相同基础版本源代码所构建 RISCV32 baremetal 工具链的 codesize 完全一致。