RISCV 工具链数据报第一期：CodeSize 面面观

本期 RISCV 工具链数据报将对 GNU 和 LLVM-Clang toolchain，利用 CSiBE benchmark 进行 codesize 方面的对比。

本文 Toolchain 的源码和构建关键信息如下：

	GNU	LLVM-Clang
源码位置	https://github.com/riscv/riscv-gnu-toolchain.git	https://git.llvm.org/git/llvm.git https://git.llvm.org/git/clang.git https://git.llvm.org/git/lld.git
版本	commit id: 2c037e6 GCC9.2.0 Binutils2.32	commit id: llvm：2c4ca68 clang：65acf43 lld：64b024a LLVM:10.0.0
构建方式	Newlib cross-compiler	-DLLVM_EXPERIMENTAL_TARGETS_TO_BUILD="RISCV" -DLLVM_LINK_LLVM_DYLIB=ON

CSiBE 简介：CSiBE 是用于衡量编译器的代码尺寸指标的权威 benchmark（ http://szeged.github.io/csibe/ ）。我们利用 CSiBE 工具集中的 toolchain-files 指定工具链和编译选项，对 CSiBE-v2.1.1 测试集所包含的 16 个代码包进行了编译和测试。这 16 个代码包的详细情况可参考 CSiBE 官网，此不赘述。

CSiBE 的测试原理如下：以 bzip2-1.0.2 为例，它包含 9 个 c 文件，CSiBE 会将 9 个 c 文件分别编译生成 obj 文件，然后用 size 命令获得 obj 文件的 text 和 data 段字节数并相加。最后把所有代码包中所有的 obj 文件的结果以 csv 表格的形式报告。

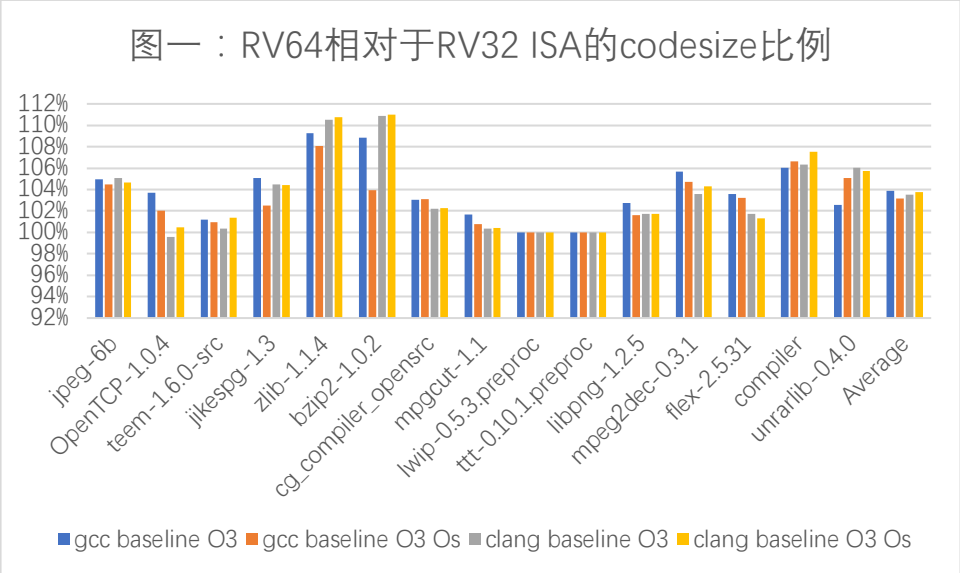
我们的测试目标是：对 GCC 和 LLVM-Clang，在 RISCV32 和 RISCV64 两种 ISA 下（以下简称 RV32 和 RV64），以-O3 为 baseline，-Os -O3 为 codesize 优化选项，得到 8 组数据。具体地，GCC 和 LLVM-Clang 的编译命令行分别如下：

	GCC	LLVM-Clang
RV32 O3 baseline	riscv32-unknown-elf-gcc/g++ -O3 -march=rv32imafdc	clang/clang++ --target=riscv32 -O3 --target=riscv32 -march=rv32imafdc -mabi=ilp32d –gcc-toolchain=~/foo/install32gcc/ -B~/foo/install32gcc/riscv32-unknown-elf/bin/
RV32 Os O3	riscv32-unknown-elf-gcc/g++ -O3 -Os -march=rv32imafdc	clang/clang++ --target=riscv32 -Os -O3 --target=riscv32 -march=rv32imafdc -mabi=ilp32d –gcc-toolchain=~/foo/install32gcc/ -B~/foo/install32gcc/riscv32-unknown-elf/bin/
RV64 O3 baseline	riscv64-unknown-elf-gcc/g++ -O3 -march=rv64imafdc	clang/clang++ --target=riscv64 -O3 --target=riscv64 -march=rv64imafdc -mabi=lp64d –gcc-

		toolchain=~/foo/install64gcc/ -B~/foo/install64gcc/riscv32-unknown-elf/bin/
RV64 Os O3	riscv64-unknown-elf-gcc/g++ -O3 -Os -march=rv64imafdc	clang/clang++ --target=riscv64 -Os -O3 --target=riscv64 -march=rv64imafdc -mabi=lp64d –gcc-toolchain=~/foo/install64gcc/ -B~/foo/install64gcc/riscv32-unknown-elf/bin/

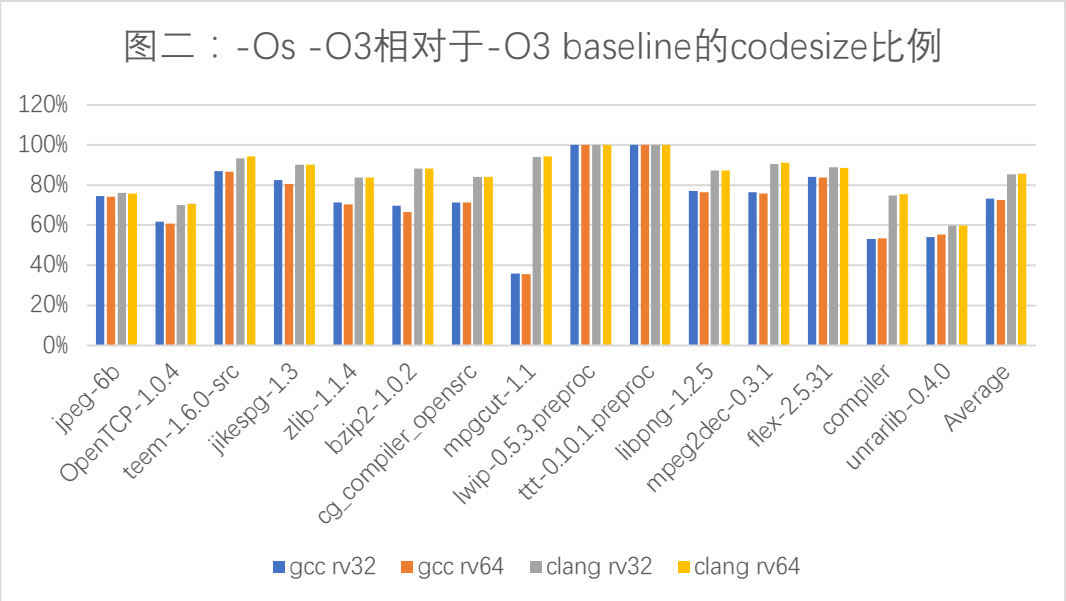
在对 csv 中 16 个代码包的原始 codesize 字节数据进行按各个代码包文件个数进行平均的处理以后，我们接下来将报告 3 个类别的对比数据：

一、RV32 和 RV64 在相同工具链和相同优化选项下的 codesize 对比，见图一：



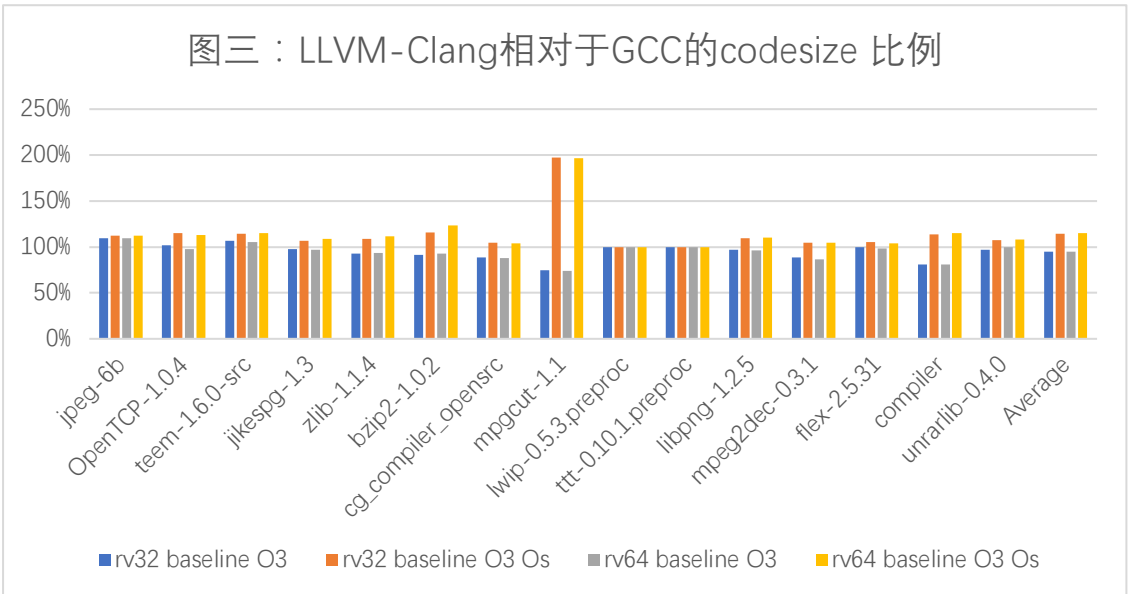
除了 OpenTCP 以外，对其他所有的 case，RV64 指令集下的 codesize 都大于或等于 RV32 指令集下的 codesize，在图一图例显示的 4 个对照组中，平均的代码膨胀比例依次是 3.9%，3.1%， 3.5%， 3.7%。

二、在相同指令集和相同工具链上，以-O3 为基准，关闭和开启-Os 时 codesize 的对比，见图二：



图二表明在-O3 的基础上再开启-Os 后，无论是 GCC 还是 LLVM-Clang，对绝大多数 case，codesize 都有明显降低，平均降低幅度在图二图例显示的 4 个对照组中分别是：26.7%， 27.3%， 14.6%， 14.4%，GCC 上得到的降低幅度大于 LLVM-Clang 上得到的降低幅度。在 mpgcut-1.1 上，GCC 开启-OS 获得了 65%的 codesize 降低，是所有 case 中最大的降幅，但对于 lwip-0.5.3.preproc 和 ttt-0.10.1preproc，codesize 完全没有变化。

三、在相同指令集和等价优化选项下，GNU 和 LLVM-Clang 工具链的 codesize 对比，见图三：



上图表明， 在 O3 baseline 的选项下，LLVM-Clang 所产生代码的 codesize 要略微小于 GCC 所产生代码的 codesize，在 RV32 指令集下，相对比例是 95.1%，在 RV64 指令集下，相对比例是 94.8%。与此相反的是，O3 baseline+Os 选项后，LLVM-Clang 在 codesize 上的表现，相比 GCC 具有明显劣势：在 RV32 指令集下，LLVM-Clang codesize 相对 GCC 平均增加 14%，最大增加 97%，该数据比例在 RV64 下基本一致。

总结：

在本期 RISCV 工具链数据报中，我们利用 CSiBE 对衡量编译器质量的重要指标之一：codesize 进行了测试和对比，结果表明，在 CSiBE 测试集上：

1、对同样编译器和本文所使用的测试选项，RV64 和 RV32 指令集的 codesize 相差 3～4 个百分点，RV64 的 codesize 稍大。

2、对同样的指令集和编译器，以-O3 为 baseline，添加 Os 选项后，在 GCC 上可以获得约 27%的 codesize 缩减，而在 LLVM-Clang 上仅可以获得约 14%的 codesize 缩减；

3、对同样的指令集和等价的编译选项，LLVM-Clang 在-O3 baseline 上，产生代码的 codesize 小于 GCC 5 个百分点；LLVM-Clang 在-O3 -Os 选项下，产生代码的 codesize 大于 GCC 14 个百分点。

第 2 和第 3 点共同说明，LLVM-Clang 在 codesize 的优化方面，相对 GCC 来说，还有很大潜能。