RISC-V Open Source Compiler Performance

Is it Good Enough?

Jeremy Bennett, Craig Blackmore, Paolo Savini, Embecosm

How to measure compiler performance

Key benchmark criteria:

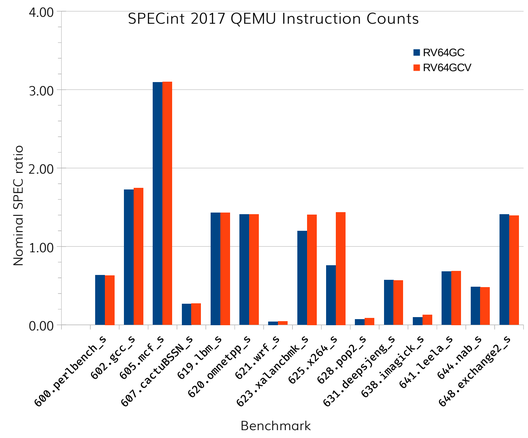
- based on multiple real programs, which are open source, or at least readily available
- able to benchmark code speed, code throughput (application cores) and code size (microcontrollers)
- an easily understood and reproducible score, fairly based on the performance of all programs
- updated regularly, to avoid being gamed by compilers.

Chosen benchmarks:

- Microcontrollers: Embench 1.0 and 2.0
- Application class cores: SPEC CPU 2006 and 2017

RISC-V Targets	Arm Targets	x86_64 Targets
CV32E40Pv2 on Nexys A4 FPGA @ 15MHz	STM32F407 @ 16MHz	AMD Threadripper 1950X @ 3.4GHz
HiFive Unmatched @1.4GHz	Apple M1 @ 3.8MHz	
MILK-V Pioneer @ 2.0GHz	QEMU User Mode	
QEMU User Mode		

The impact of ISA Extensions: RISC-V Vector (RVV)



Results obtained using GCC 14.0.1 of 24 April 2017. For both runs Glibc was built *without* RVV enabled.

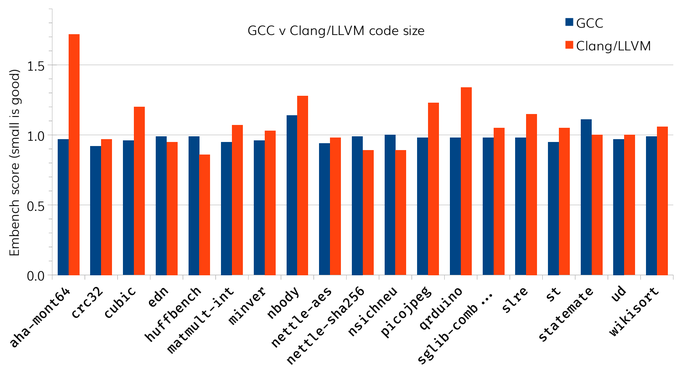
We use QEMU instruction counts as a (crude) proxy for code speed. SPEC CPU 2017 ratios are based on run times, so we convert instruction counts as though they were for a machine executing 10^9 instructions per second.

The runs were performed with LTO enabled in both cases, which is known to be buggy on RISC-V. We only show results for the 15 benchmarks which ran successfully both with and without RVV.

For six benchmarks, instruction counts increased marginally with RVV enabled (in all cases by < 1%). RVV is particularly effective with a small subset (3 benchmarks showing a >20% improvement).

Overall we see a 9.4% improvement.

GCC v Clang/LLVM

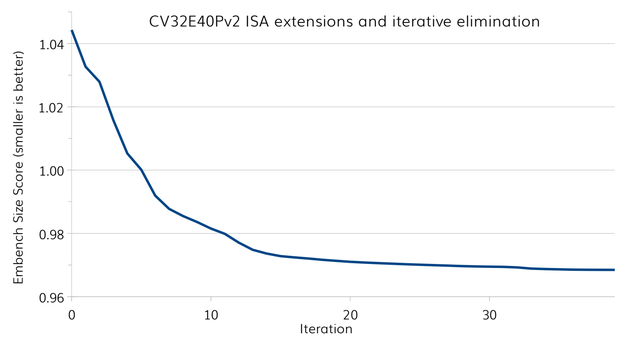


Embench 1.0 code size scores using GCC 10.0 and Clang/LLVM 9.0

SPEC 2006	*Dyn inst. count*
401.bzip2	-2.7%
403.gcc	-2.1%
429.mcf	-6.0%
445.gobmk	-9.1%
456.hmmer	-44.5%
458.sjeng	-10.6%
462.libquantum	148.2%
464.h264ref	-23.6%
471.omnetpp	-9.7%
473.astar	-20.2%
483.xalancbmk	-13.0%
Geometric mean	**-6.4%**

RISC-V QEMU instr count
red = more LLVM instructions

ISA extensions with iterative optimization

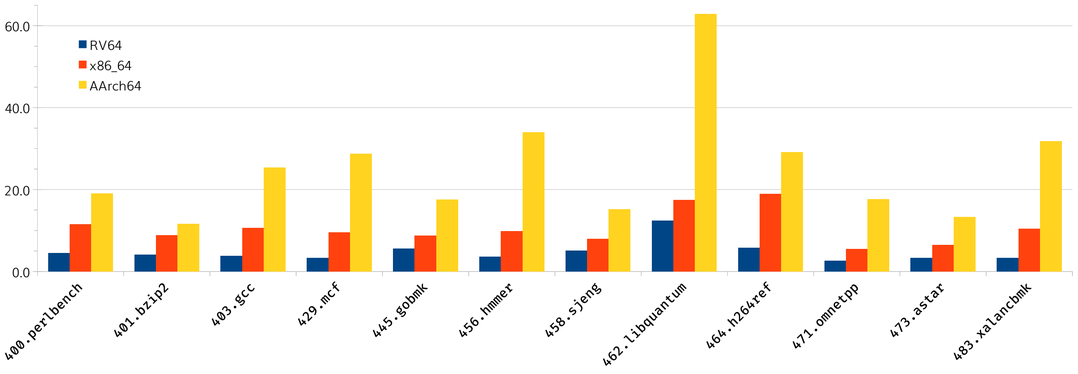


We combine two techniques to improve Embench code size on the CV32E40Pv2. First we select the memory (`xcvmem`) and multiply accumulate (`xcvmac`) ISA extensions. Then we apply iterative compilation to identify specific GCC optimizations to further improve code size.

After 37 iterations, we have improved the code size by 7.3%. The final set of compile line options are:

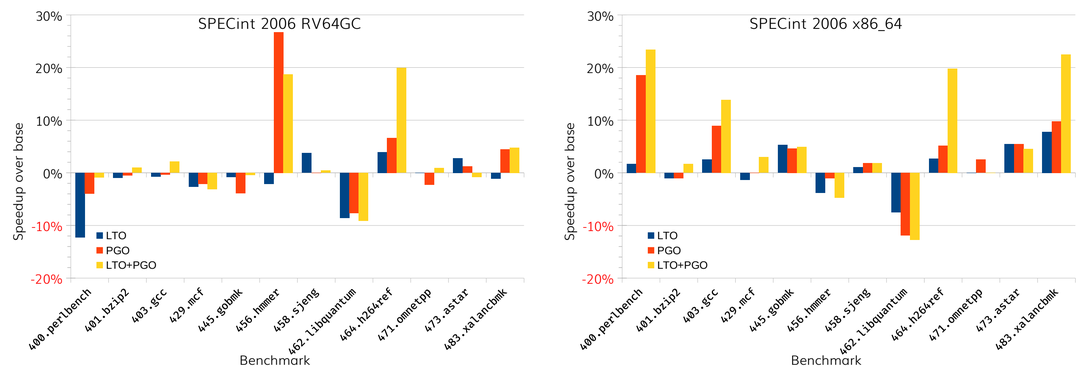
```
-march=rv32imac_xcvmac_xcvmem -ffunction-sections -msave-restore  -Os --param=gcse-unrestricted-cost=0 \
    --param=iv-consider-all-candidates-bound=48 --param=loop-invariant-max-bbs-in-loop=0 \
    --param=max-hoist-depth=0 --param=max-predicted-iterations=0 --param=sink-frequency-threshold=100 \
    -fno-caller-saves -fipa-pta -fno-ipa-reference-addressable -fno-math-errno -fno-reorder-functions \
    -fno-signed-zeros -fno-tree-forwprop -msmall-data-limit=2
```

RV64 v x86_64 v AArch64



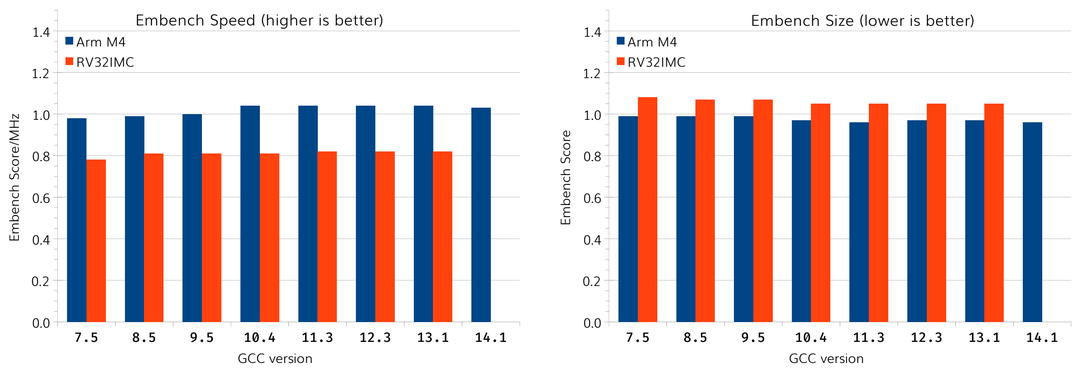
SPECint 2006 normalized to 1GHz clock for RV64 (MILK-V Pioneer), x86_64 (AMD Ryzen Threadripper 1950X) and AArch64 (Apple M1)

Impact of LTO and PGO



LTO and PGO are both powerful optimizations, although not always beneficial. RISC-V suffers from a buggy LTO implementation, while its PGO is hampered by the relatively short range of branches

Improvements over time

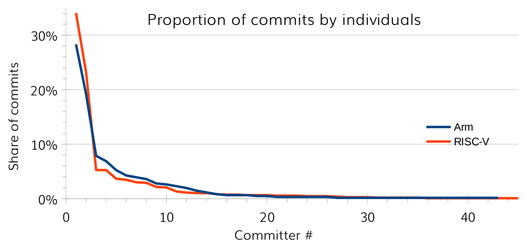


Embench 1.0 scores for Arm Cortex M1 (16MHz) and CV32E40Pv1 FPGA (10MHz)

The community

Central to development of the compiler tool chain is the community, and particularly the small band of contributors who make the majority of the changes. The table below looks at various metrics from the commit log for GCC 14.1. These are commits tagged as being either RISC-V or Arm/AArch64. One point of note is that in both cases more than half the commits come from two contributors for the same company. Perhaps not surprising for Arm, but for RISC-V?

Metric	RISC-V	Arm
Commits	1,058	611
Committers	45	43
Biggest contribution	363	173
Committers making 90% of commits	15	13
Corporate contributors	16	12



A huge thank you to our collaborators and supporters: OpenHW Group, the Embench Group and Dolphin Design.

www.embecosm.com