RISE QEMU function benchmarks

These are hand-written versions of common memory and string library functions provided by SiFive Inc. For each benchmarked function, we are comparing a "baseline" QEMU against a "latest" QEMU. By default, we look at three versions of the code.

1. the standard library "scalar" implementation of the function
2. the hand-written vector code for VLEN=128 and LMUL=1 ("small vector")
3. the hand-written vector code for VLEN=1024 and LMUL=8 ("large vector")

We thus have a total of 6 datasets.

There are four graphs. The first looks at the average number of instructions executed per iteration, and is a sanity check. Although we plot all 6 datasets, only 3 should show on the graph - the "latest" versions of "scalar", "small vector" and "large vector", since the version of QEMU should have no impact on the number of instructions being executed. This graph is useful to developers of the vector implementations to see how their code behaves for different sizes of data, but this is outside the scope of this project.

The remaining three graphs capture QEMU performance for each of the three versions of the code. They use the metric of nanoseconds per instruction, to measure the efficiency of QEMU. Each graph shows this metric, plotted against problem size, one line for the "baseline" version of QEMU, the other the "latest" QEMU.

This report is generated entirely automatically using the command

```
./run-all-benchmarks.py
```

This takes less than 20 minutes to run on a 40 thread AMD Threadripper 1950X at 3.4GHz. The code is in the `strmem-benchmarks` directory of the rise-rvv-tcg-qemu-tooling repository. It is intended to be portable, but to date has only been tested on Ubuntu 22.04 LTS.

Document details

- Datestamp: 2025-01-08-12-58-14
- User: paolo

Functions to be benchmarked

Any functions which failed to benchmark are noted.

- memcpy

QEMU versions

- 6528013b5f5ba6bb3934b7f5fe57a3110680530f
- db95037b428e28b084ce550872406da9ba4217bf

Tool chain configuration

GCC configuration

```
Using built-in specs. COLLECT_GCC=riscv64-unknown-linux-gnu-gcc
COLLECT_LTO_WRAPPER=/home/paolo/QEMU/install/libexec/gcc/riscv64-unknown-linux-gnu/15.0.0/lto-wrapper
Target: riscv64-unknown-linux-gnu Configured with: /home/paolo/QEMU/gcc/configure
--target=riscv64-unknown-linux-gnu --prefix=/home/paolo/QEMU/install
--with-sysroot=/home/paolo/QEMU/install/sysroot --with-pkgversion=g24689b84b8e-dirty --with-system-zlib
--enable-shared --enable-tls --enable-languages=c,c++,fortran --disable-libmudflap --disable-libssp
--disable-libquadmath --disable-libsanitizer --disable-nls --disable-bootstrap --src=/home/paolo/QEMU/gcc
--enable-multilib --with-abi=lp64d --with-arch=rv64gc --with-tune= --with-isa-spec=20191213
'CFLAGS_FOR_TARGET=-O2    -mcmodel=medany' 'CXXFLAGS_FOR_TARGET=-O2    -mcmodel=medany' Thread model:
posix Supported LTO compression algorithms: zlib gcc version 15.0.0 20240717 (experimental)
(g24689b84b8e-dirty)
```

Assembler version

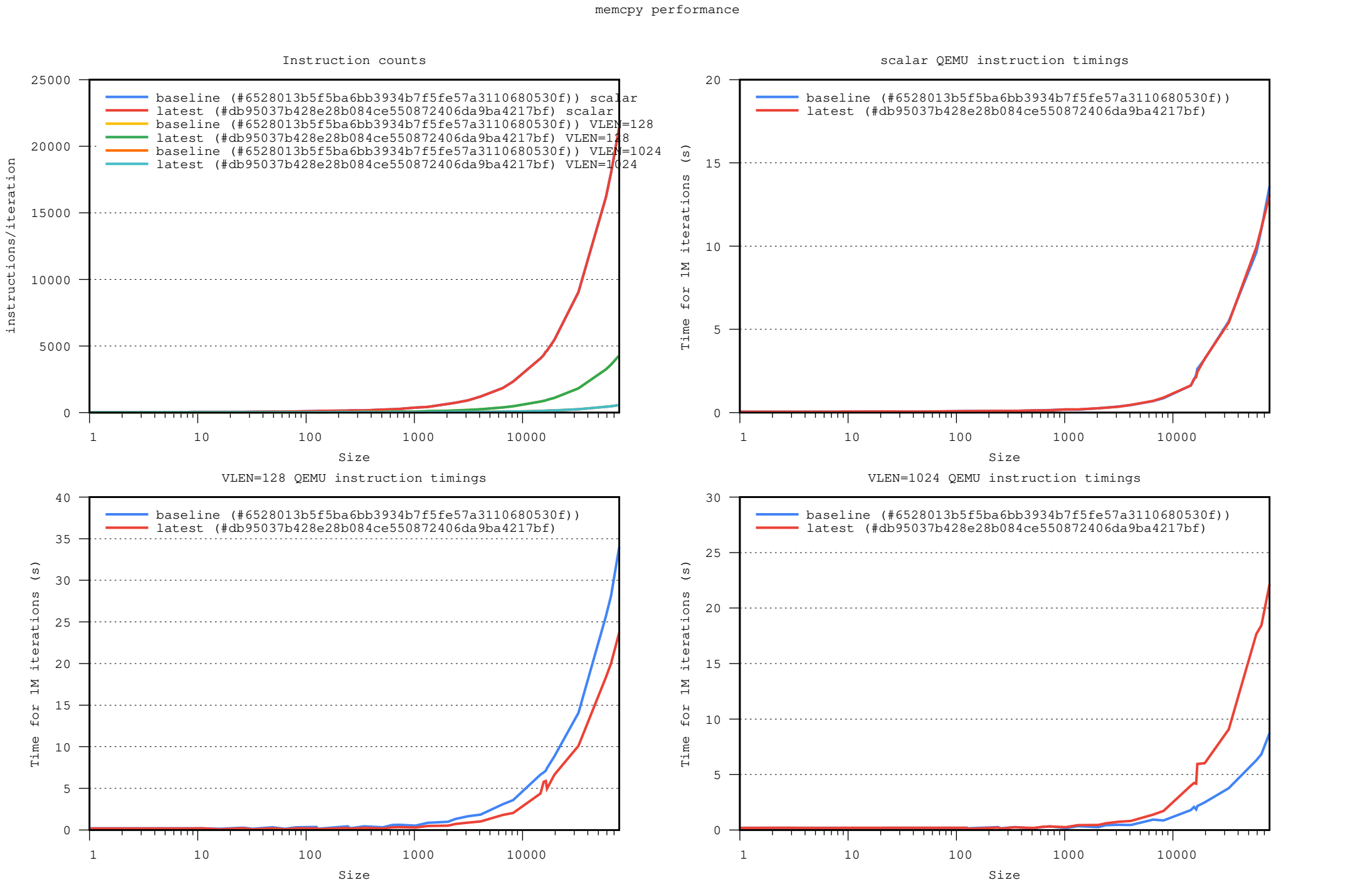
```
GNU assembler version 2.42.50 (riscv64-unknown-linux-gnu) using BFD version (GNU Binutils)
2.42.50.20240717
```

Linker version

```
GNU ld (GNU Binutils) 2.42.50.20240717
```

Glibc version

```
ldd (GNU libc) 2.39.9000 Copyright (C) 2024 Free Software Foundation, Inc. This is free software; see the
source for copying conditions.  There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE. Written by Roland McGrath and Ulrich Drepper.
```


memcpy performance
Instruction counts
instructions/iteration
25000
20000
15000
10000
5000
0
baseline (#6528013b5f5ba6bb3934b7f5fe57a3110680530f)) scalar
latest (#db95037b428e28b084ce550872406da9ba4217bf) scalar
baseline (#6528013b5f5ba6bb3934b7f5fe57a3110680530f)) VLEN=128
latest (#db95037b428e28b084ce550872406da9ba4217bf) VLEN=128
baseline (#6528013b5f5ba6bb3934b7f5fe57a3110680530f)) VLEN=1024
latest (#db95037b428e28b084ce550872406da9ba4217bf) VLEN=1024
1
10
100
1000
10000
Size
scalar QEMU instruction timings
Time for 1M iterations (s)
20
15
10
5
0
baseline (#6528013b5f5ba6bb3934b7f5fe57a3110680530f))
latest (#db95037b428e28b084ce550872406da9ba4217bf)
1
10
100
1000
10000
Size
VLEN=128 QEMU instruction timings
Time for 1M iterations (s)
40
35
30
25
20
15
10
5
0
baseline (#6528013b5f5ba6bb3934b7f5fe57a3110680530f))
latest (#db95037b428e28b084ce550872406da9ba4217bf)
1
10
100
1000
10000
Size
VLEN=1024 QEMU instruction timings
Time for 1M iterations (s)
30
25
20
15
10
5
0
baseline (#6528013b5f5ba6bb3934b7f5fe57a3110680530f))
latest (#db95037b428e28b084ce550872406da9ba4217bf)
1
10
100
1000
10000
Size