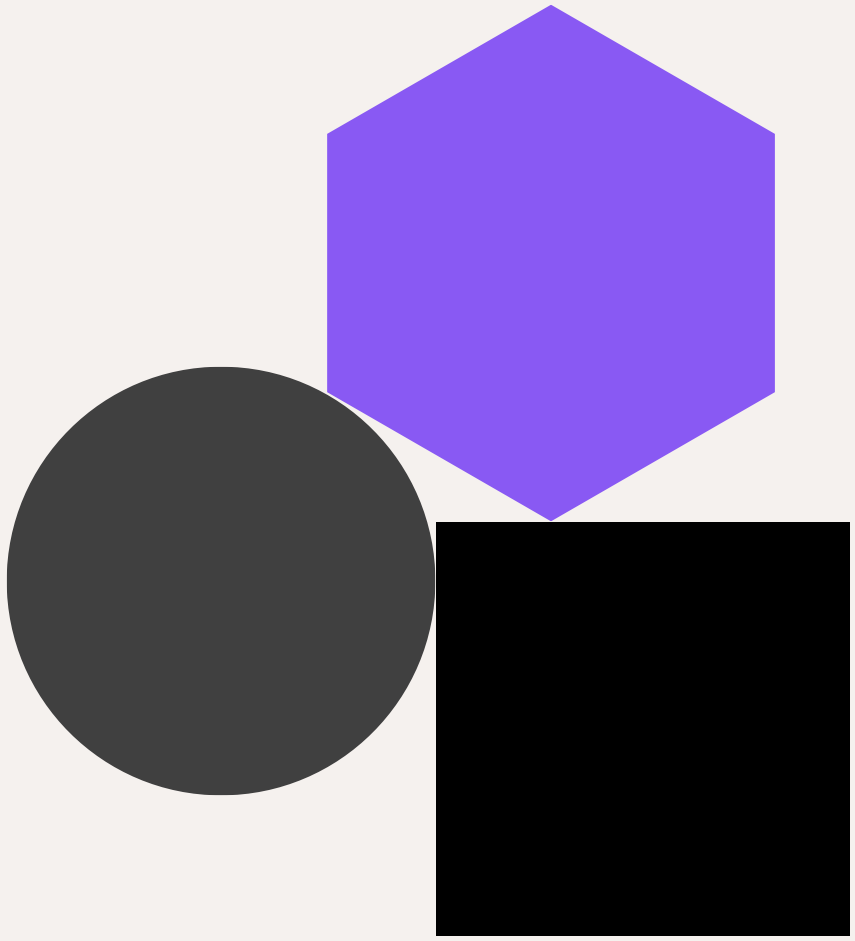
CHERI performance (continued...)



Dr Carl Shaw

CHERI Blossoms, March 2026

→ CHERI performance

Codasip

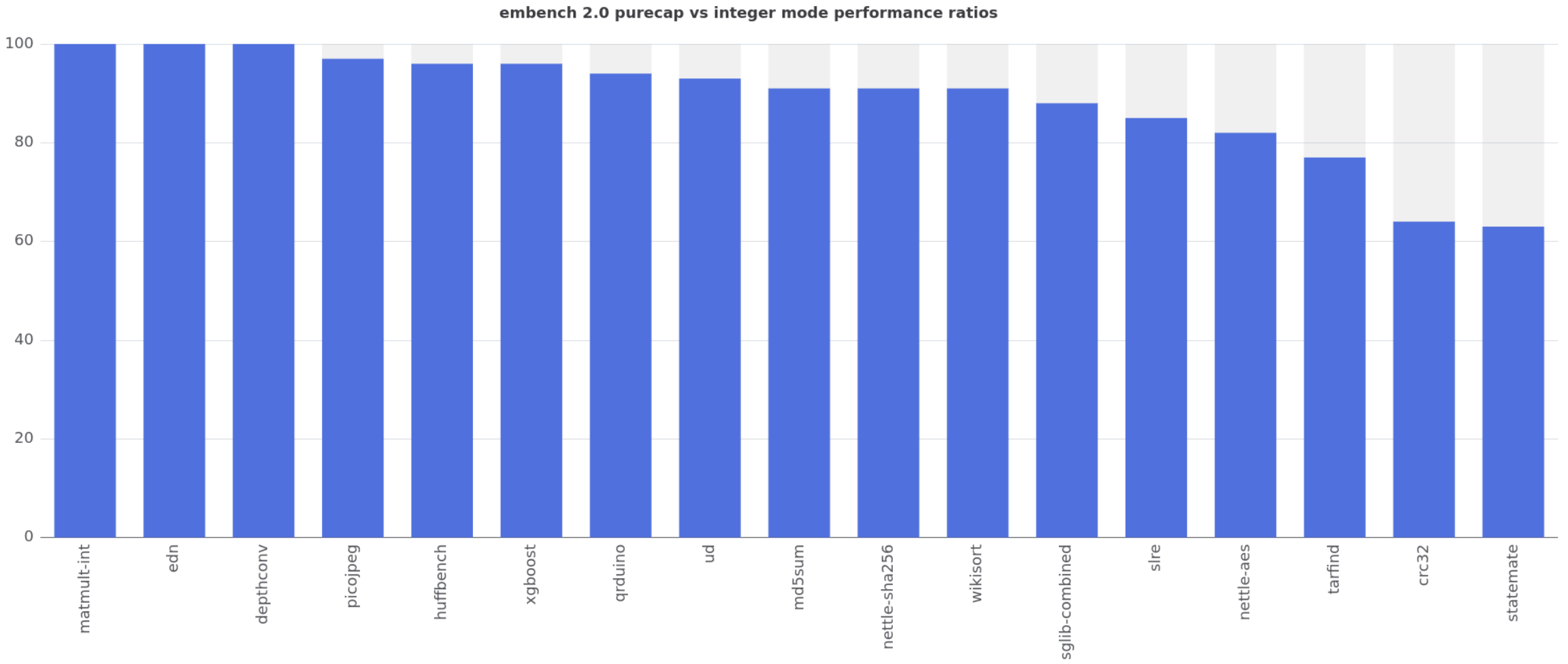
Things to note:

- Work is focused on Linux and/or Codasip-cores
  - But should be generally applicable (in most cases!)
  - Software is (or will be soon) open sourced

- Benchmark results (CHERI and non-CHERI) have been obtained by running on a Codasip X730-MP4 64-bit CHERI application CPU on the Codasip Prime FPGA platform
  - Your mileage may vary

→ Microbenchmarks (embench 2.0)

Codasip



- CHERI purecap vs non-CHERI results shown at CHERItech Nov 2025

→ Analysing the microbenchmark results (crc32)

Codasip

RISC-V purecap code:

```
<crc32pseudo>:
…
17 09 00 00   auipc  cs2, 0
0f 49 09 00   lc     cs2, 0(cs2)
…
```

- Called many times
- In CHERI case there is an indirect load of the global variable containing the address of a look-up table

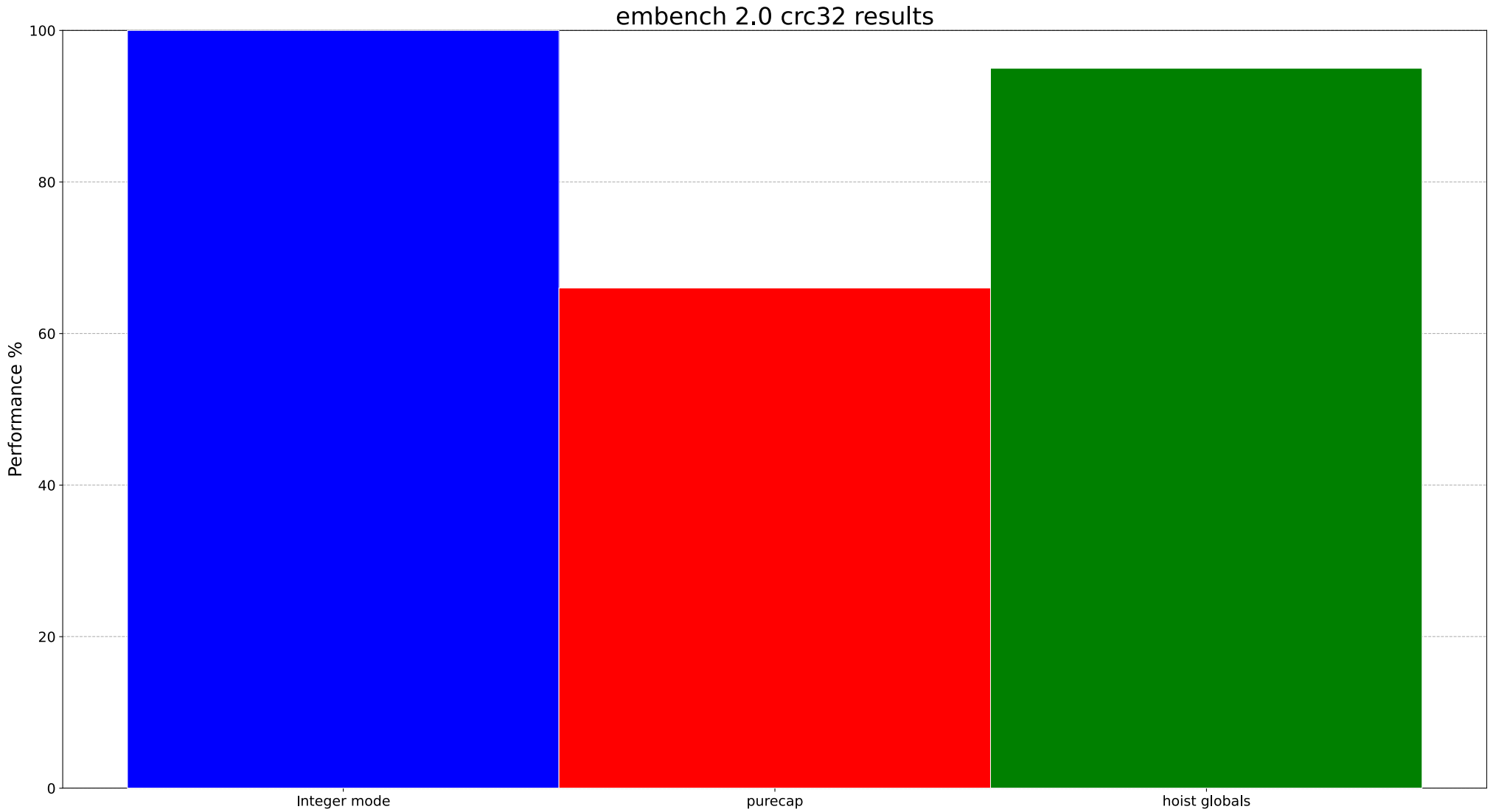
RISC-V integer code:

```
<crc32pseudo>:
…
17 05 00 00   auipc   a0, 0
13 09 05 00   mv      s2, a0
…
```

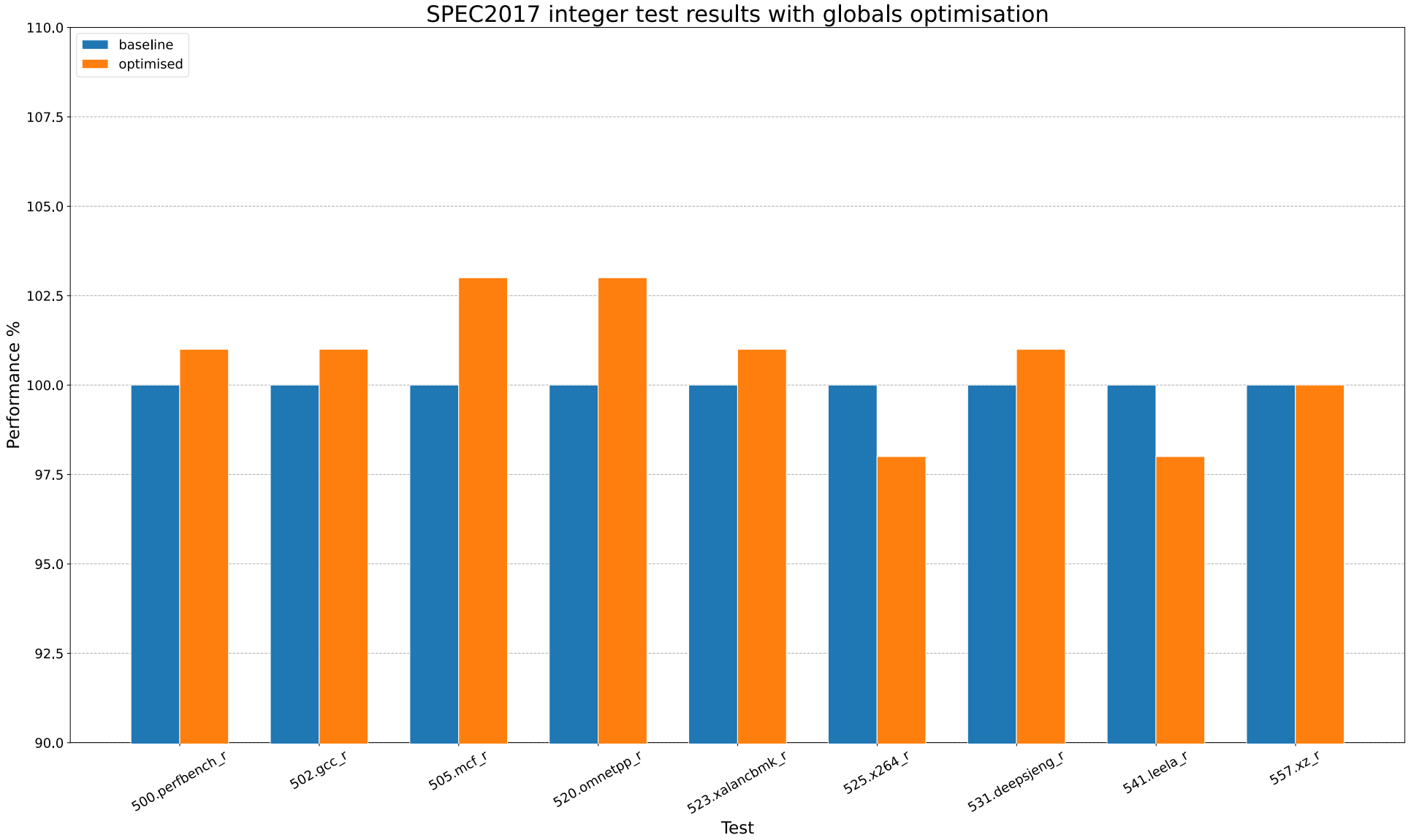
- In the integer-mode code the address of the look-up table global is used directly

→ Analysing the microbenchmark results (crc32)

Codasip

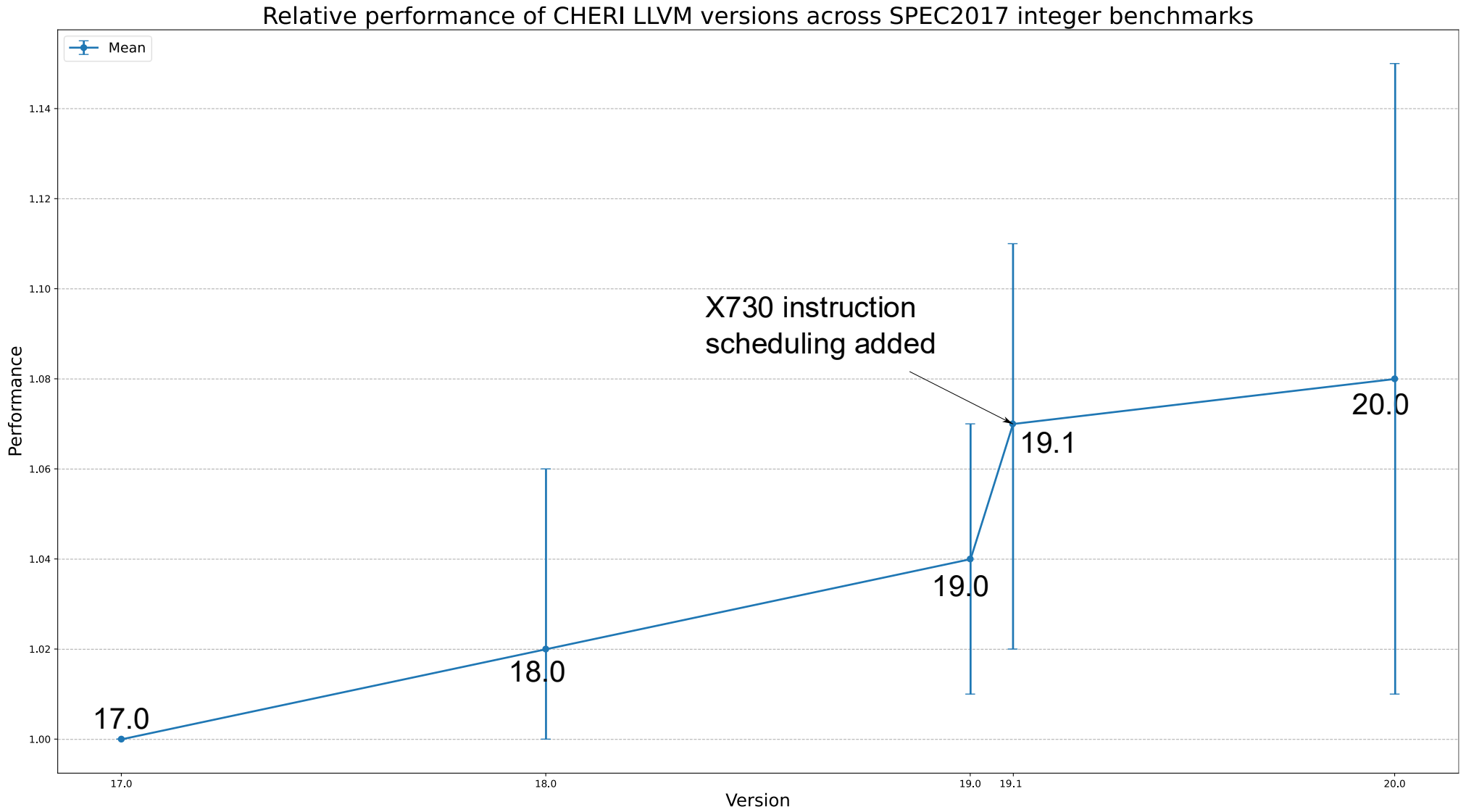


→ Globals optimisation (early work in progress)



→ Compiler improvements

Codasip



→ Optimised string functions

Codasip

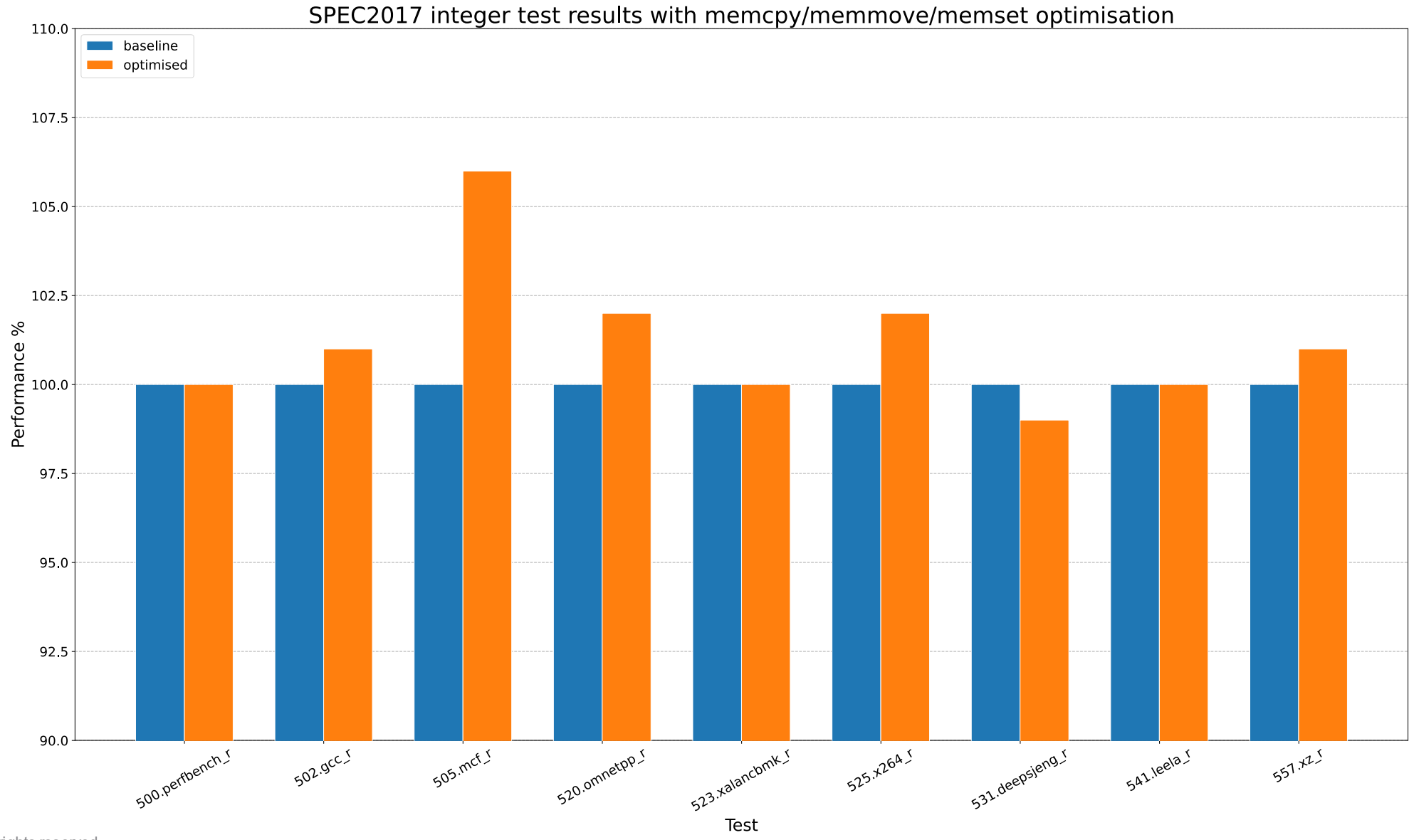
Results (measured on Codasip Prime X730@100MHz)

Benchmark	MiB/s current	MiB/s improved	MiB/s ratio
strchr_short	20.42	23.48	115%
strchr_mid	26.70	55.55	208%
strchr_long	28.55	101.70	**356%**
strcmp_short_aligned	17.67	19.20	109%
strcmp_mid_aligned	23.86	45.36	190%
strcmp_long_aligned	24.69	71.86	**291%**

Benchmark	MiB/s current	MiB/s improved	MiB/s ratio
strlenbench_short	16.84	24.80	147%
strlenbench_mid	21.06	63.60	302%
strlenbench_long	21.84	132.73	**608%**
strcmp_short_unaligned	16.29	13.68	84%
strcmp_mid_unaligned	21.95	21.58	98%
strcmp_long_unaligned	0.78	0.83	106%

Note: Uses Zbb and supports CHERI (Zbb commonly used, part of RVA22 and 23)

→ Memcpy/memmove/memset optimisation impact



→ Linux perf

Codasip

- We now have "perf" running under Linux with access to Hardware Performance Counters
  - Uses recent work on implementing tracing in the Linux kernel
  - Includes support for HW breakpoints
  - Also adding in support for our Capability Management Unit (DDR tag controller) HW performance counters

→ Linux perf

Codasip

```
# perf stat -e cycles,instructions,load_commited,cap_load_commited,store_commited,cap_store_commited
-- ./mcf_r_base.target inp.in

Performance counter stats for './mcf_r_base.target inp.in':

       1101416923      cycles
        464574975      instructions                     #    0.42  insn per cycle
         50357557      load_commited
         62124969      cap_load_commited
         11996774      store_commited
         38582162      cap_store_commited

     11.156775960 seconds time elapsed

     10.610582000 seconds user
      0.539521000 seconds sys
```

→ Optimising the musl memory allocator

Codasip

- Implemented improved hash function
- Showed promise under QEMU
- Hasn't shown expected improvements on real FPGA HW yet
  - Trying to work out why...

→ Where next?

Codasip

- Better analysis using perf on a wider range of applications (e.g. pgbench)
- Analysing spatial and temporal properties of capabilities
  - Cache impact
  - Looking at efficient revocation
  - New Linux kernel module to snapshot capabilities in memory
- Continue to update LLVM – expect more gains from 21 and 22
- Continue data driven optimisation

Thank you!

carl.shaw@codasip.com