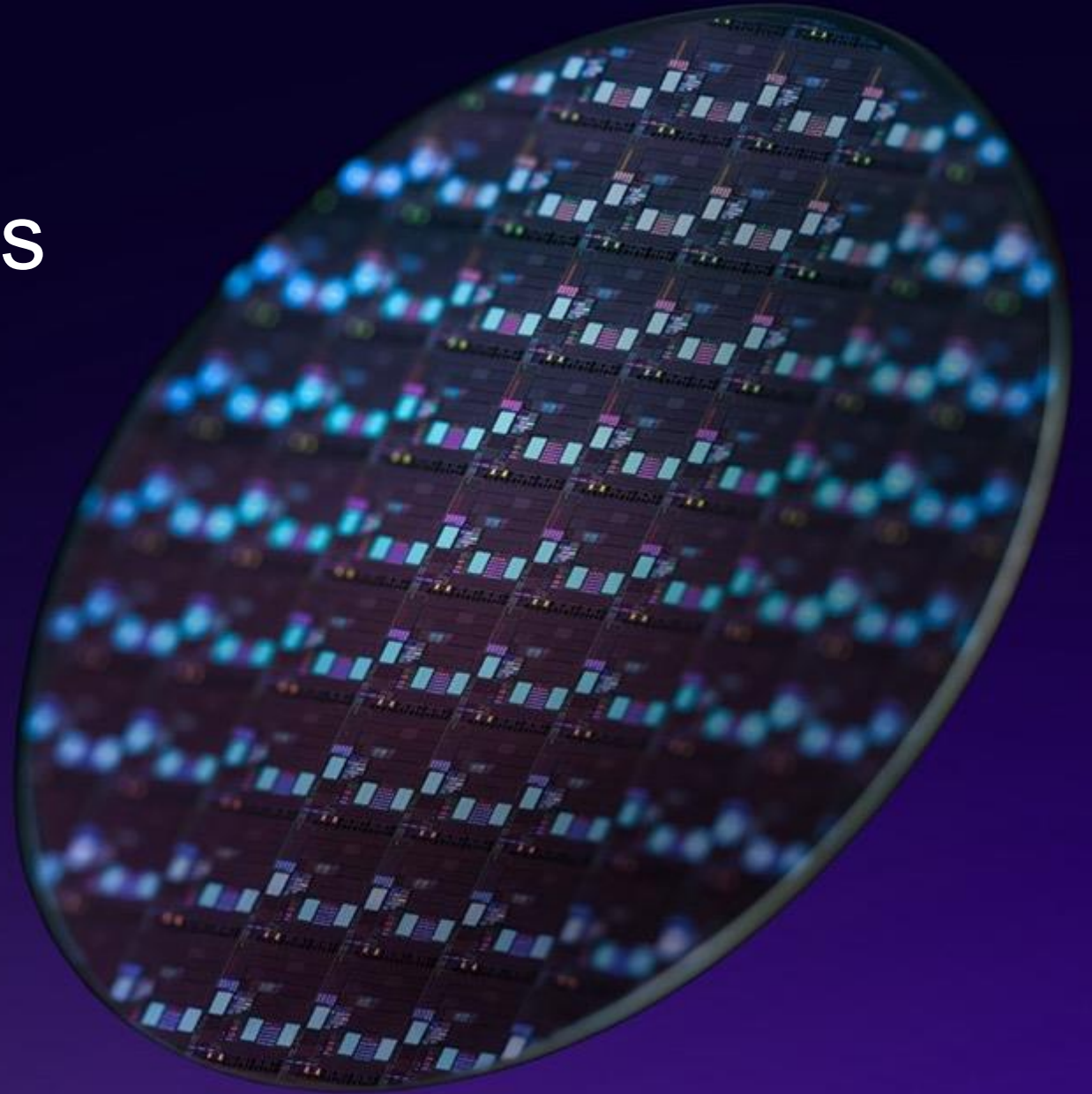


# arm

Can LLVM-libc be used as  
the library for embedded  
toolchains?



William Huynh  
15/09/25

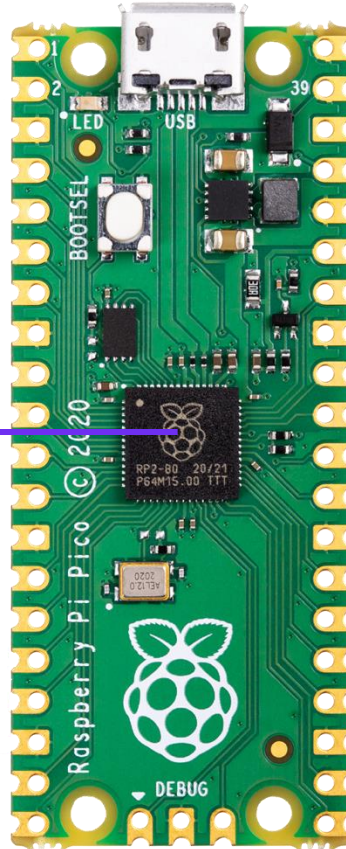
# arm

## Background

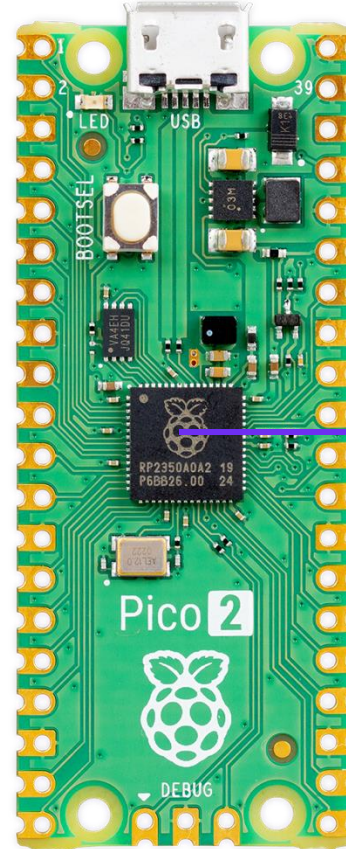


# We develop a toolchain for embedded Arm platforms

RP2040  
Cortex-M0+  
Arm v6-M



RP2350  
Cortex-M33  
Arm v8-M



# Current toolchain setup

C++ library  
(libc++)

C library  
(picolibc)

Compiler  
(clang)

# Ideal toolchain setup

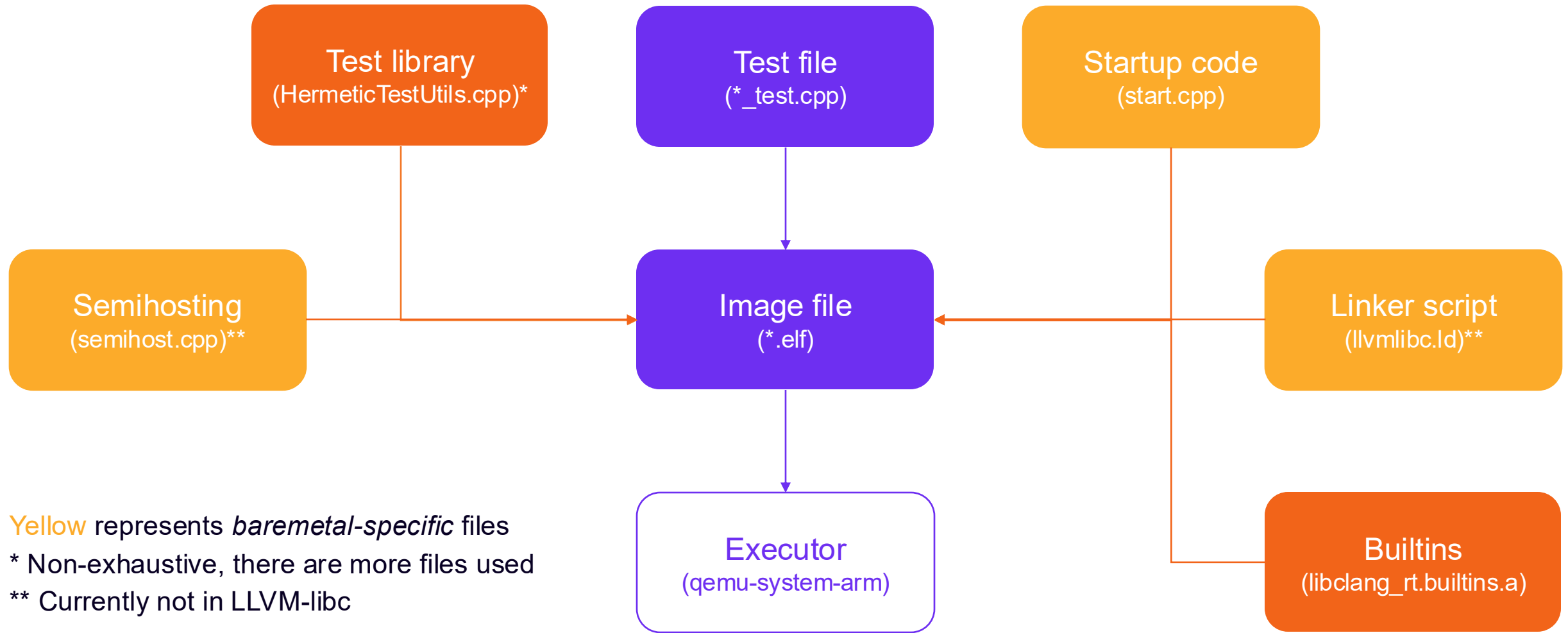
C++ library  
(libc++)

C library  
(LLVM-libc)

Compiler  
(clang)

Is LLVM-libc correct and performant?

# Basic testing setup for each (variant, test)



Reference: <https://github.com/arm/arm-toolchain/blob/arm-software/arm-software/embedded/arm-runtimes/CMakeLists.txt>

# Testing in Action

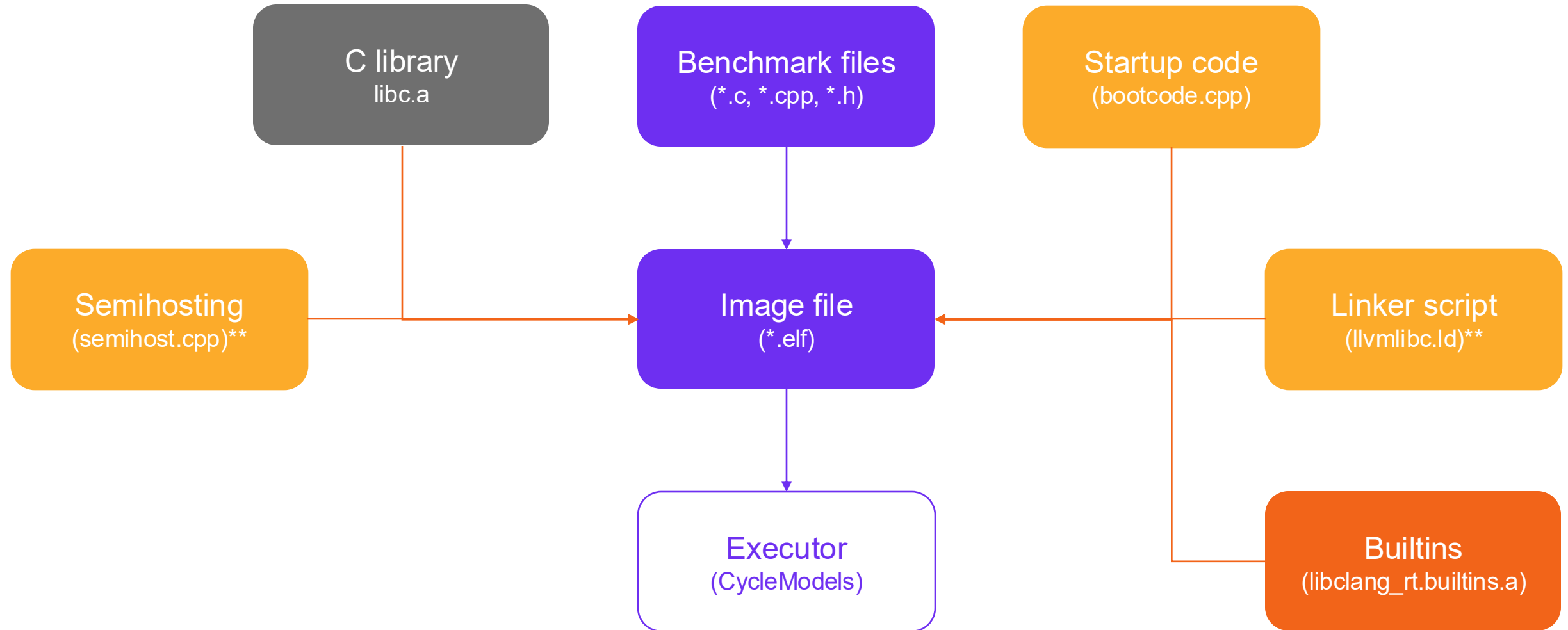
- Looks identical to upstream tests but internally uses our setup.
- Passes most, but not all of the tests.

```
[1069/3255] Running hermetic test libc.test.src.math.ldexp_test.__hermetic__  
[=====] Running 6 tests from 1 test suite.  
[ RUN      ] LlvmLibcLdExpTest.SpecialNumbers  
[          OK ] LlvmLibcLdExpTest.SpecialNumbers (0 ns)  
[ RUN      ] LlvmLibcLdExpTest.PowersOfTwo  
[          OK ] LlvmLibcLdExpTest.PowersOfTwo (0 ns)  
[ RUN      ] LlvmLibcLdExpTest.Overflow  
[          OK ] LlvmLibcLdExpTest.Overflow (0 ns)  
[ RUN      ] LlvmLibcLdExpTest.UnderflowToZeroOnNormal  
[          OK ] LlvmLibcLdExpTest.UnderflowToZeroOnNormal (0 ns)  
[ RUN      ] LlvmLibcLdExpTest.UnderflowToZeroOnSubnormal  
[          OK ] LlvmLibcLdExpTest.UnderflowToZeroOnSubnormal (0 ns)  
[ RUN      ] LlvmLibcLdExpTest.NormalOperation  
[          OK ] LlvmLibcLdExpTest.NormalOperation (10 ms)  
Ran 6 tests.  PASS: 6  FAIL: 0
```

Reference: <https://github.com/arm/arm-toolchain/actions/runs/18703574536/job/53336991030#step:9:1>



# Basic benchmarking setup for each variant



\*\* Currently not in LLVM-libc

# LLVM-libc vs picolibc

Tested on Cortex-M55 (v8.1-M), accurate as of 21/10/25

**+52%**

Performance on  
distance math  
functions

**+0%**

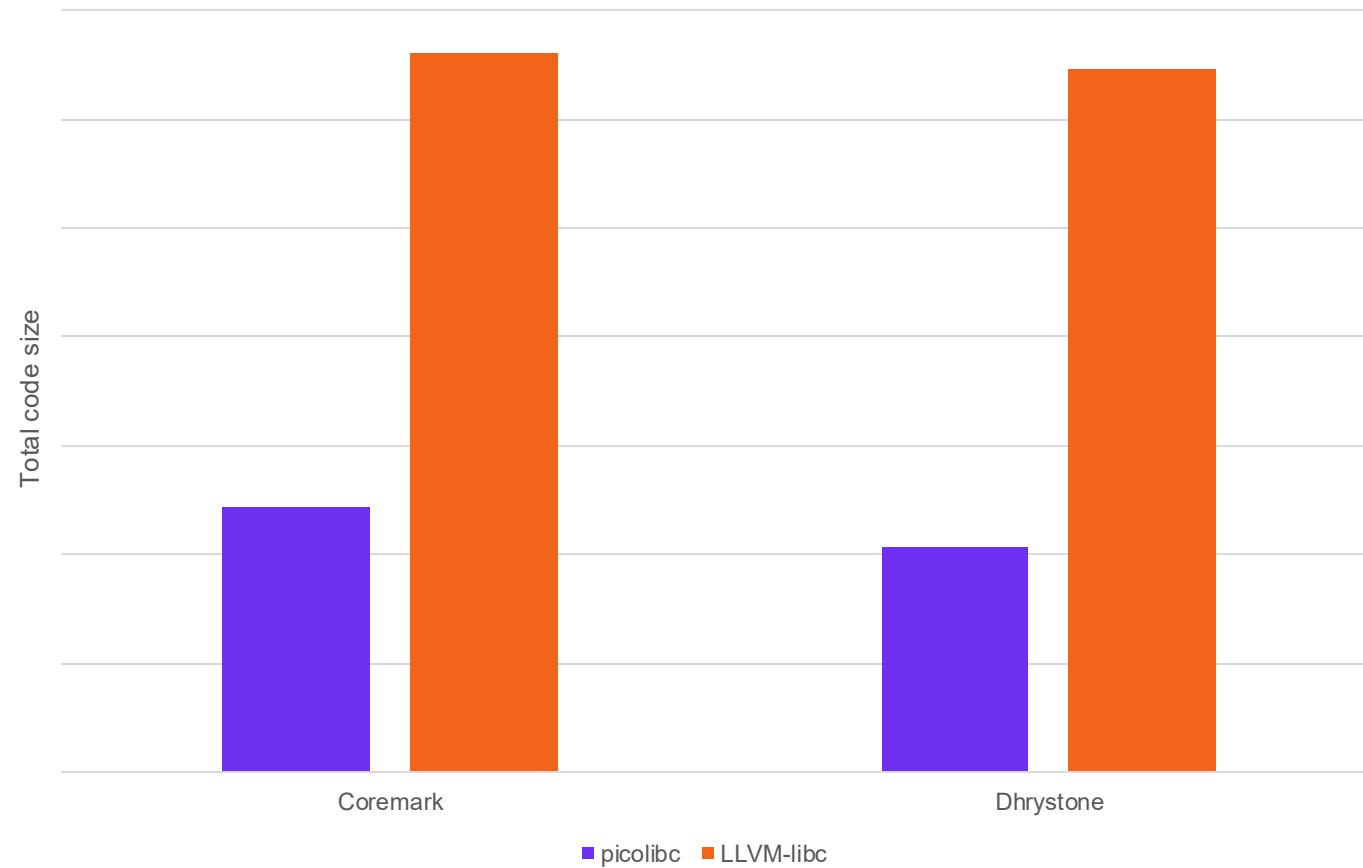
Performance on a  
generic benchmark  
suite

**-15%**

Performance on  
Dhrystone

# The figures look worse on code size...

We are seeing increased code size



Reference: <https://discourse.llvm.org/t/rfc-printf-code-size-optimization/83146>

There is still a lot of work to do

# References

- Toolchain: <https://github.com/arm/arm-toolchain>
- Tracking issue: <https://github.com/llvm/llvm-project/issues/145349>
- Inspired by: <https://llvm.org/devmtg/2024-10/slides/techtalk/Hosek-ModernEmbeddedDevelopment-with-LLVM.pdf>
- Introduction to LLVM-libc: <https://libc.llvm.org/>
- QEMU: <https://www.qemu.org/docs/master/system/target-arm.html>

arm

Merci

Danke

Gracias

Grazie

谢谢

ありがとう

Asante

Thank You

감사합니다

धन्यवाद

Kiitos

شكراً

ধন্যবাদ

תודה

ధన్యవాదములు

Köszönöm

The ARM logo, consisting of the lowercase letters 'arm' in a white, sans-serif font, is positioned on the left side of the slide. The background is a solid blue color with a subtle gradient that transitions from a lighter blue at the top to a darker blue at the bottom.

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

[www.arm.com/company/policies/trademarks](http://www.arm.com/company/policies/trademarks)