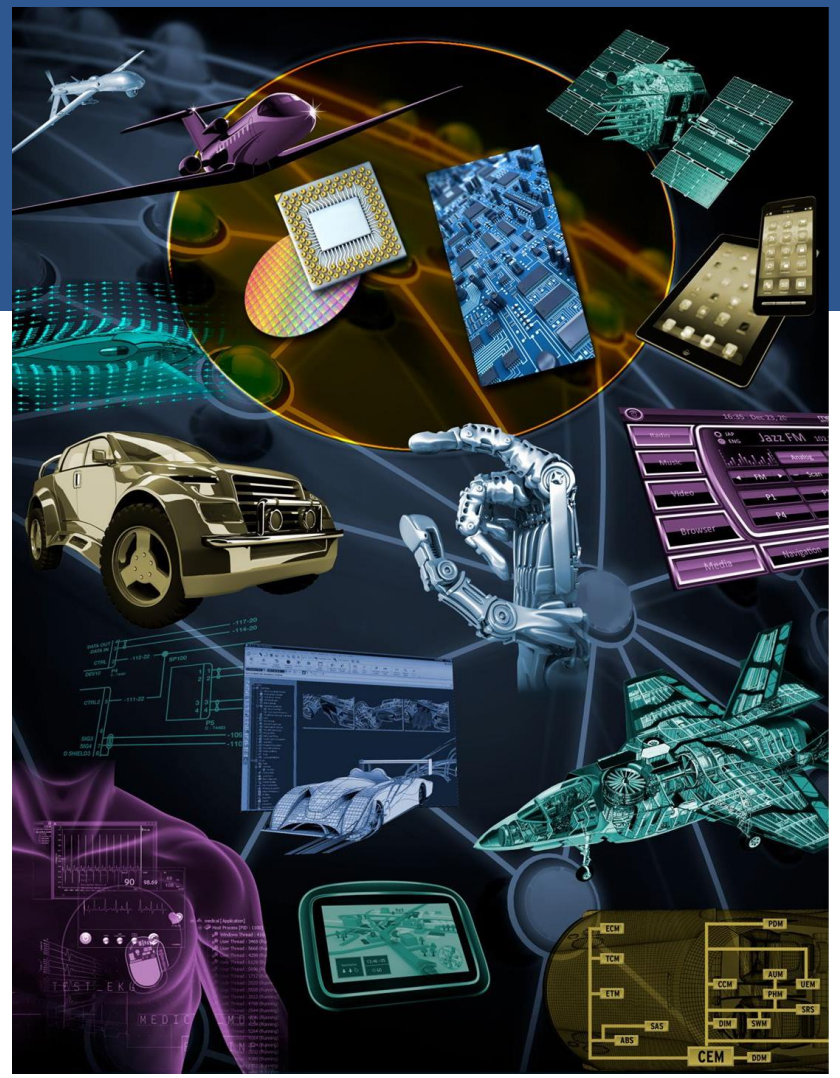


# LLVM in a Bare Metal Environment

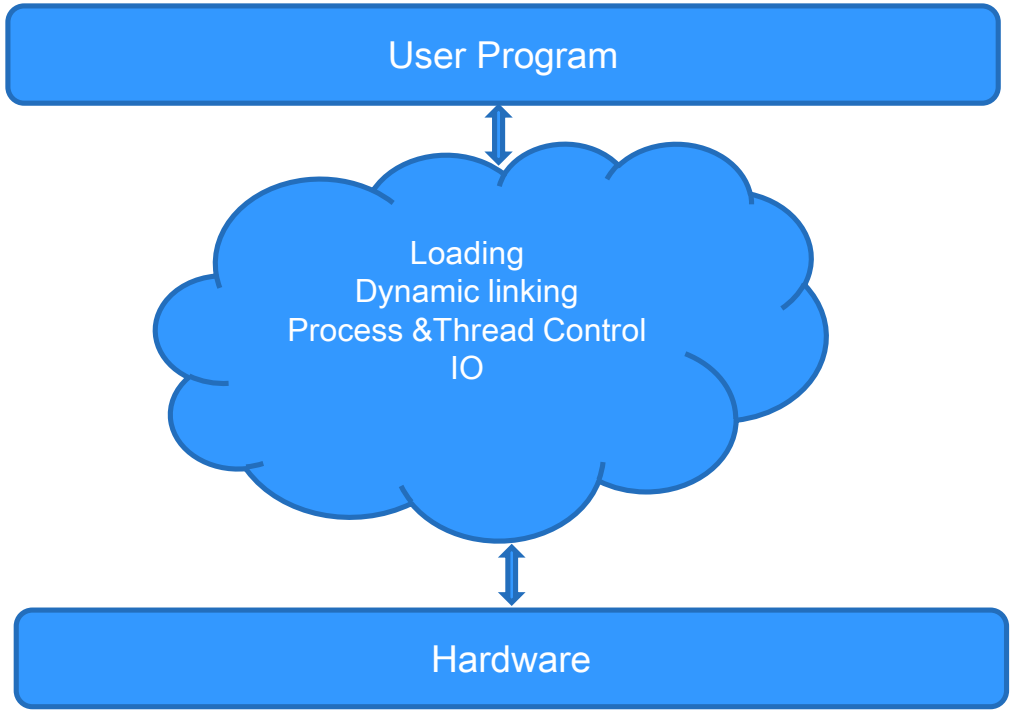
Hafiz Abid Qadeer

October 6, 2020

Mentor®
A Siemens Business

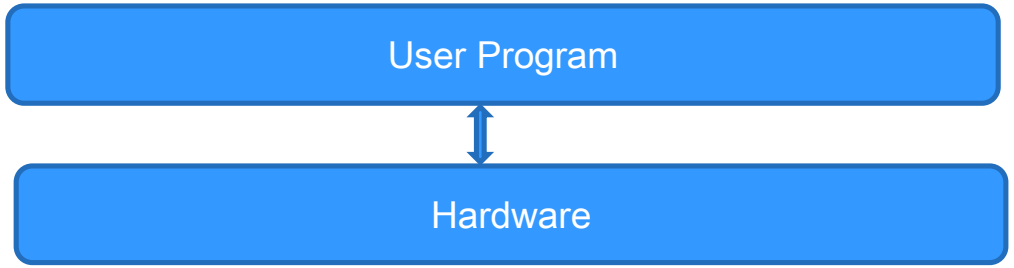
# Overview

# Why is Bare Metal Different



# Why is Bare Metal Different

- Static Libraries
- Customized memory layout
- No thread or process control
- No Filesystem or console IO



# Bare Metal in LLVM

- Not exercised enough

Baremetal.cpp (added in 2017)
("-lclang_rt.builtins-" + getArchName() + ".a"));

Baremetal.cpp (Fixed in 2020)
("-lclang_rt.builtins-" + getArchName()));

# Components

Compiler Driver

Compiler Proper

Runtime

Linker

# Clang Driver

- Responsibilities
- Baremetal.cpp
  - Only support ARM at the moment
  - Lacks Multilib support

```
bool BareMetal::handlesTarget(const llvm::Triple &Triple)
{ return isARMBareMetal(Triple);
}
```

# Runtime

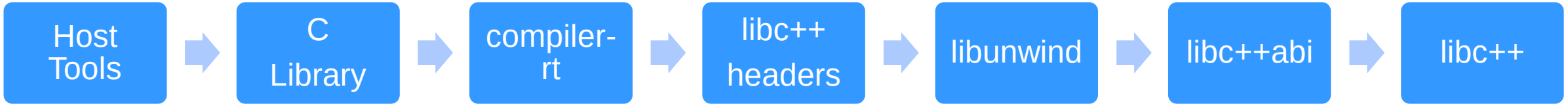
Runtime

libc++
libc++abi
libunwind

C library

compiler-rt

# Build Order



# Standalone Build

```
$ cd /path/to/build/
$ cmake …. $src/$library
$ make
$ make install
$ make check-xyz
```

# Building Runtime

CMAKE_C_COMPILER

CMAKE_ASM_COMPILER

CMAKE_CXX_COMPILER

CMAKE_C_FLAGS="--target=… -march=…"

CMAKE_CXX_FLAGS

CMAKE_ASM_FLAGS

CMAKE_TRY_COMPILE_TARGET_TYPE=STATIC_LIBRARY

# Building compiler-rt

cmake

-DCOMPILER_RT_BAREMETAL_BUILD=ON

-DCOMPILER_RT_BUILD_CRT=…

-DCOMPILER_RT_DEFAULT_TARGET_ONLY=On

-DCOMPILER_RT_BUILD_BUILTINS=ON

-DCOMPILER_RT_BUILD_SANITIZERS=OFF

-DCOMPILER_RT_BUILD_XRAY=OFF

-DCOMPILER_RT_BUILD_LIBFUZZER=OFF

-DCOMPILER_RT_BUILD_PROFILE=OFF

…

$SRC/compiler-rt

# Building libunwind

cmake

-DLIBUNWIND_ENABLE_SHARED=OFF

-DLIBCXX_ENABLE_SHARED=OFF

-DLIBUNWIND_IS_BAREMETAL=ON

-DLIBUNWIND_ENABLE_THREADS=OFF

-DLIBUNWIND_USE_COMPILER_RT=ON

….

$SRC/libunwind

# Building libc++abi

cmake

-DLIBCXXABI_BAREMETAL=ON

-DLIBCXXABI_ENABLE_THREADS=OFF

-DLIBCXXABI_ENABLE_SHARED=OFF

-DLIBCXX_ENABLE_SHARED=OFF

-DLIBCXXABI_USE_COMPILER_RT=ON

-DLIBCXXABI_ENABLE_EXCEPTIONS=ON

-DLIBCXXABI_LIBCXX_INCLUDES=...

-DLIBCXXABI_USE_LLVM_UNWINDER=ON

…

$SRC/libcxxabi

# Building libcxx

cmake

-DLIBCXX_BAREMETAL=ON

-DLIBCXX_USE_COMPILER_RT=ON

-DLIBCXX_ENABLE_SHARED=OFF

-DLIBCXX_ENABLE_EXCEPTIONS=ON

-DLIBCXX_ENABLE_THREADS=OFF

-DLIBCXX_ENABLE_MONOTONIC_CLOCK=OFF

-DLIBCXXABI_USE_LLVM_UNWINDER=ON

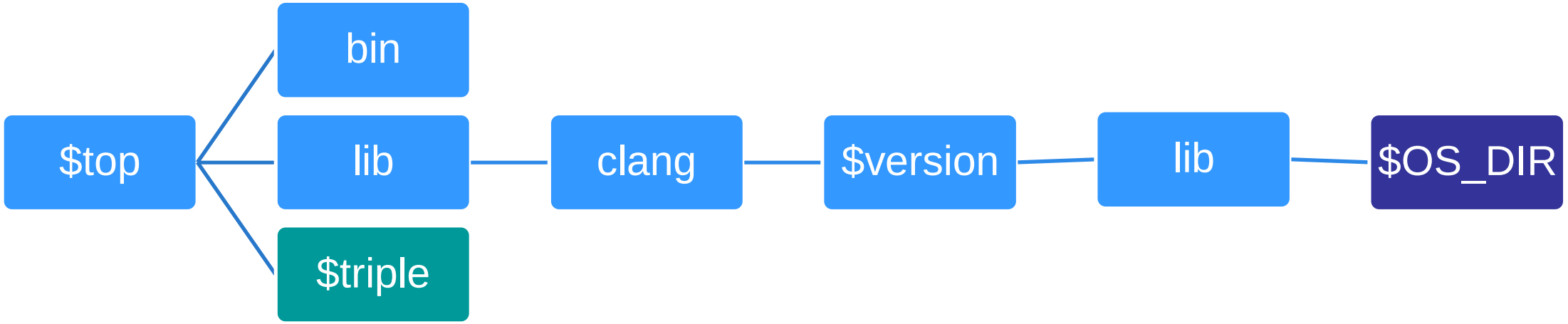
-DLIBCXX_CXX_ABI=libcxxabi

-DLIBCXX_ENABLE_FILESYSTEM=OFF

…

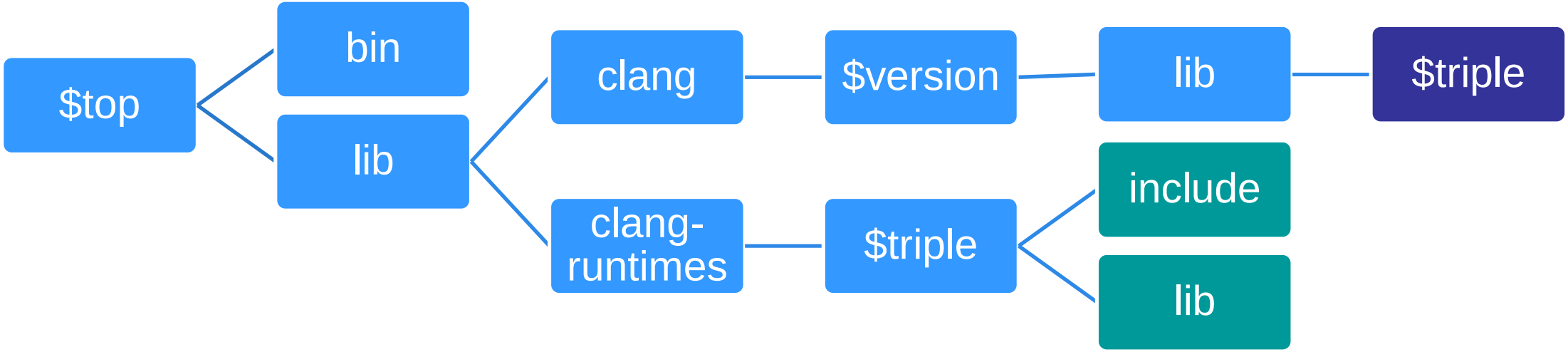
$SRC=libcxx

# Layout




- COMPILER_RT_OS_DIR
- LLVM_ENABLE_PER_TARGET_RUNTIME_DIR

# Layout



# Testing Runtimes

- The host tools are tested somewhat like this:

*RUN: Use a command to produce something*

*RUN: Dump using a tool | FileCheck …*

*…*

*CHECK: some pattern to match ..*

# Testing Runtime

- Low Level Library
  - Initialization
  - exit
- Linker Scripts
  - __eh_frame_start
  - __eh_frame_end
  - __eh_frame_hdr_start
  - __eh_frame_hdr_end
- Execution Support
  - Executor
  - Emulator

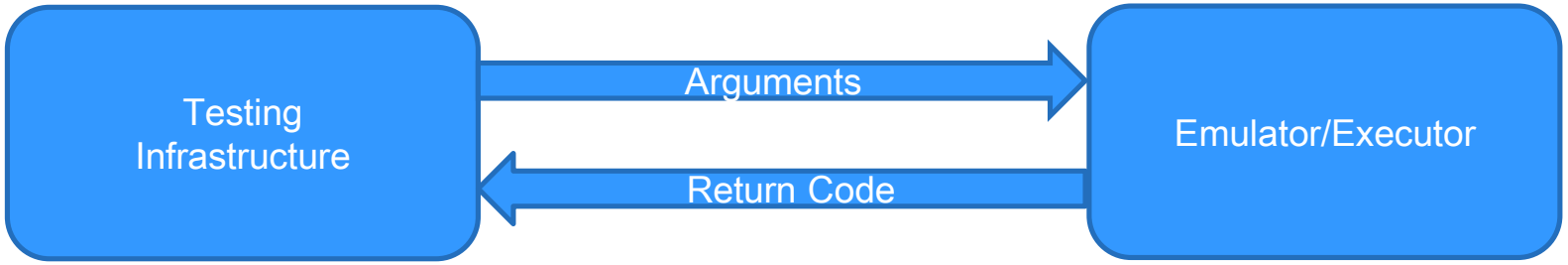
# Emulator/Executor

- Emulator is invoked like this:

*$emulator the_test_executable*

- Executor is invoked like this:

*$executor --execdir %T --codesign_identity "" --env  -- the_test_executable*



# Testing compiler-rt

- It looks like

// RUN: %clang_builtins %s %librt -lm -o %t && %run %t

- COMPILER_RT_EMULATOR=...
- COMPILER_RT_TEST_COMPILER_CFLAGS=…

# Testing libc++

```
Build: '%{cxx} %s %{flags} %{compile_flags} %{link_flags} -o %t.exe'
Run: '%{exec} %t.exe'
```

# Testing libc++

Build: '%{cxx} %s %{flags} %{compile_flags} %{link_flags} -o %t.exe'
Run: '%{exec} %t.exe'

FOO.pass.cpp
FOO.compile.fail.cpp

# Testing libc++

Build: '%{cxx} %s %{flags} %{compile_flags} %{link_flags} -o %t.exe'
Run: '%{exec} %t.exe'

FOO.pass.cpp
FOO.compile.fail.cpp

CMake Variables

# Testing libc++

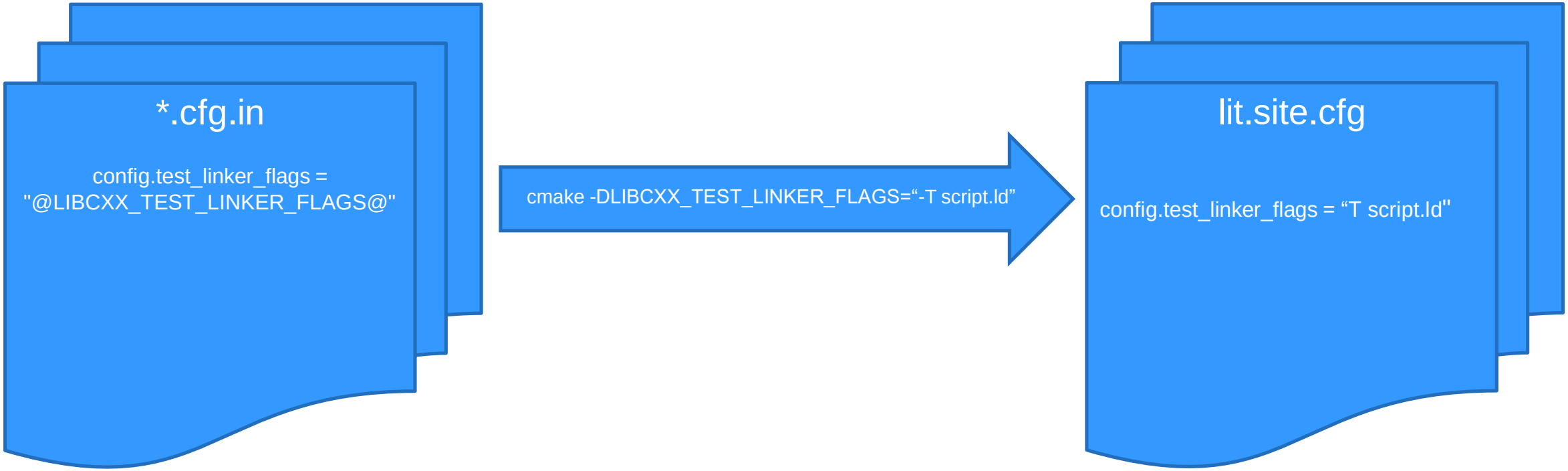
Build: '%{cxx} %s %{flags} %{compile_flags} %{link_flags} -o %t.exe'
Run: '%{exec} %t.exe'

FOO.pass.cpp
FOO.compile.fail.cpp

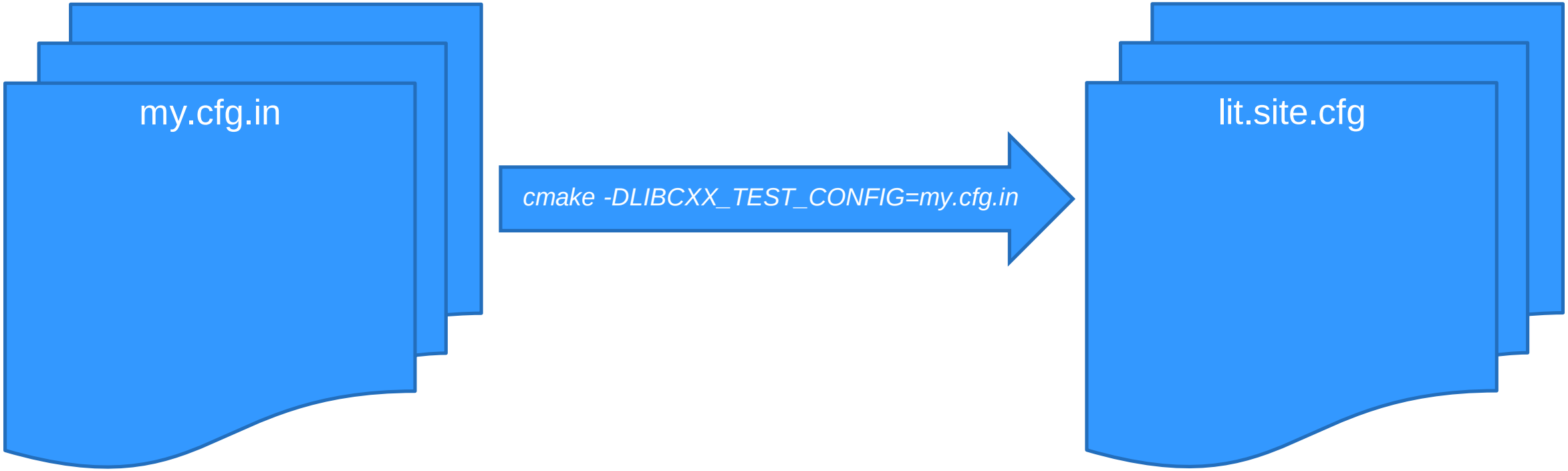
CMake Variables

ADDITIONAL_COMPILE_FLAGS
FILE_DEPENDENCIES.

# Testing libc++



# Testing libc++



# Testing libc++

- Overriding values at test time

*$llvm-lit --param=test_linker_flags="script2.ld" …*

# Testing libc++

LIBCXXABI_TEST_LINKER_FLAGS

LIBCXXABI_TEST_COMPILER_FLAGS

LIBCXXABI_EXECUTOR

LIBCXX_TEST_LINKER_FLAGS

LIBCXX_TEST_COMPILER_FLAGS

LIBCXX_EXECUTOR

LIBUNWIND_TEST_LINKER_FLAGS

LIBUNWIND_TEST_COMPILER_FLAGS

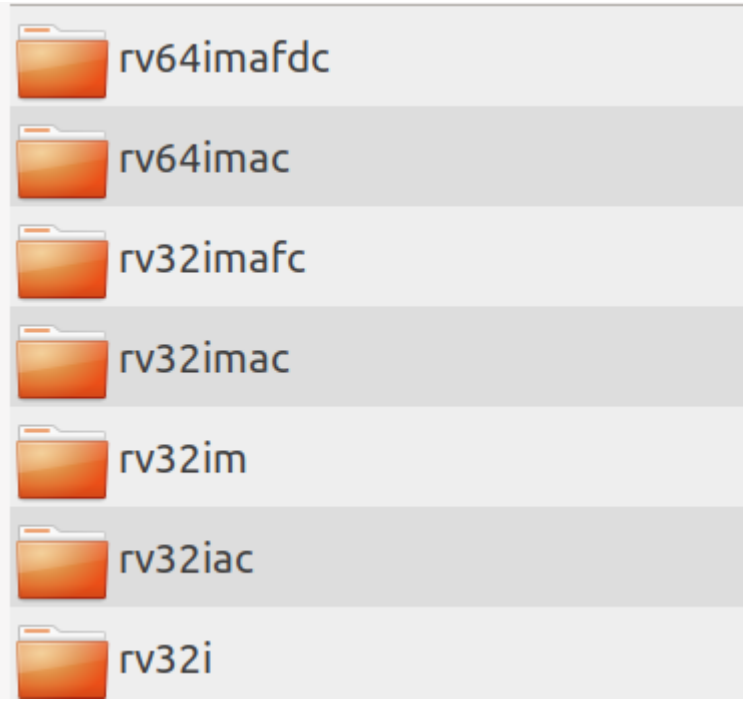
LIBUNWIND_EXECUTOR

# Unsupported Tests

- File streams
- Process Control
- C library limitations
  - locales
  - wide characters
  - strtold

# Multilibs

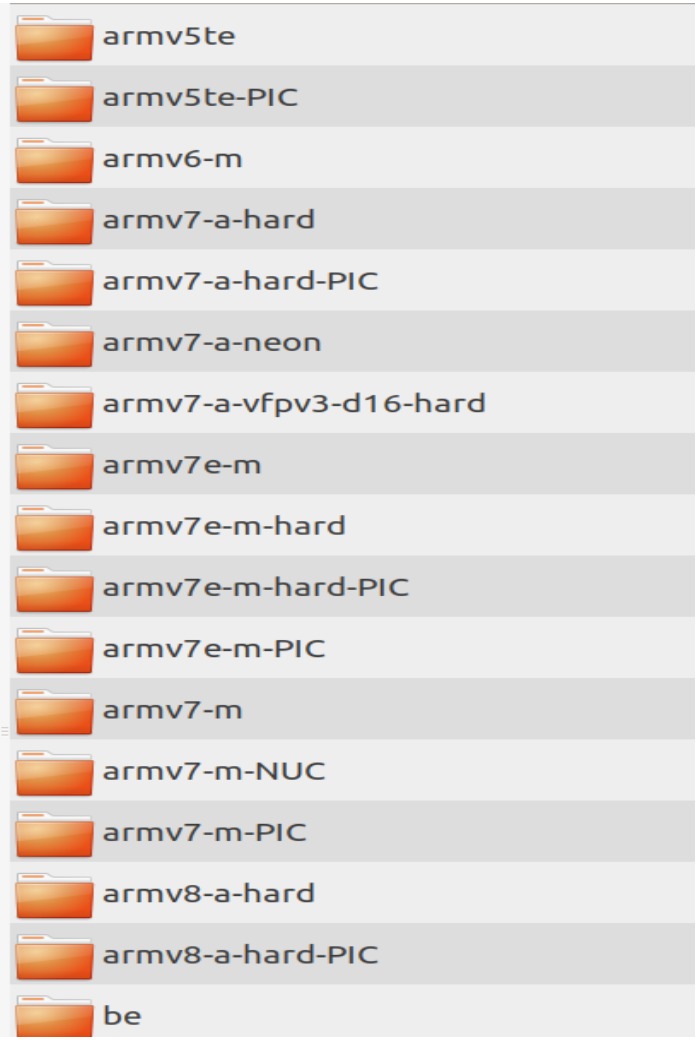
- What are multilibs




```
$ riscv64-unknown-elf-gcc -print-multi-lib
.;
rv32i/ilp32;@march=rv32i@mabi=ilp32
rv32im/ilp32;@march=rv32im@mabi=ilp32
rv32iac/ilp32;@march=rv32iac@mabi=ilp32
rv32imac/ilp32;@march=rv32imac@mabi=ilp32
rv32imafc/ilp32f;@march=rv32imafc@mabi=ilp32f
rv64imac/lp64;@march=rv64imac@mabi=lp64
rv64imafdc/lp64d;@march=rv64imafdc@mabi=lp64d
```

# Multilibs

- What are multilibs




```
$arm-none-eabi-gcc -print-multi-lib
.;
thumb;@mthumb
armv5te;@march=armv5te
be;@mbig-endian
armv8-a-hard;@march=armv8-a@mfloat-abi=hard@mfpu=neon-fp-armv8
armv8-a-hard-PIC;@march=armv8-a@mfloat-abi=hard@mfpu=neon-fp-
armv8@fPIC
armv7-a-neon;@march=armv7-a@mfloat-abi=softfp@mfpu=neon
armv7-a-hard;@march=armv7-a@mfloat-abi=hard@mfpu=neon
armv7-a-vfpv3-d16-hard;@march=armv7-a@mfloat-abi=hard@mfpu=vfpv3-d16
armv7-a-hard-PIC;@march=armv7-a@mfloat-abi=hard@mfpu=neon@fPIC
vfp-hard;@march=armv5te@mfloat-abi=hard
armv5te-PIC;@march=armv5te@fPIC
vfp-hard-PIC;@march=armv5te@mfloat-abi=hard@fPIC
armv7-m;@mthumb@march=armv7-m
armv7e-m;@mthumb@march=armv7e-m
armv6-m;@mthumb@march=armv6-m
thumb2;@mthumb@march=armv7@mfix-cortex-m3-ldrd
…
```

# Multilibs

- What are multilibs
- Multilibs in clang

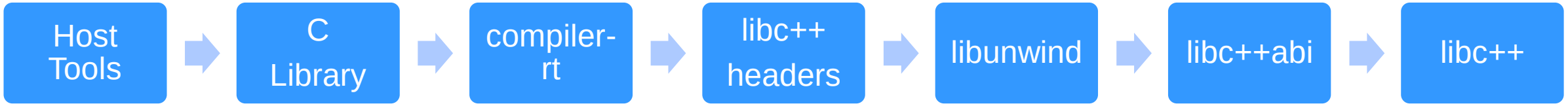
```
Multilib IMAC = Multilib("/rv64imac").flag("+march=rv64imac");
Multilib IMAFDC = Multilib("/rv64imafdc").flag("+march=rv64imafdc");

Multilib LP64D = Multilib("/lp64d").flag("+mabi=lp64d");
Multilib LP64 = Multilib("/lp64").flag("+mabi=lp64");
MultilibSet RISCVMultilibs = MultilibSet()
        .Either(IMAC, IMAFDC)
        .Either(LP64, LP64D)
        .FilterOut("/rv64imac/lp64d");
```

```
$bin/clang --target=riscv64-unknown-elf  -print-multi-lib
 rv64imac/lp64;@march=rv64imac@mabi=lp64
 rv64imafdc/lp64;@march=rv64imafdc@mabi=lp64
 rv64imafdc/lp64d;@march=rv64imafdc@mabi=lp64d
```

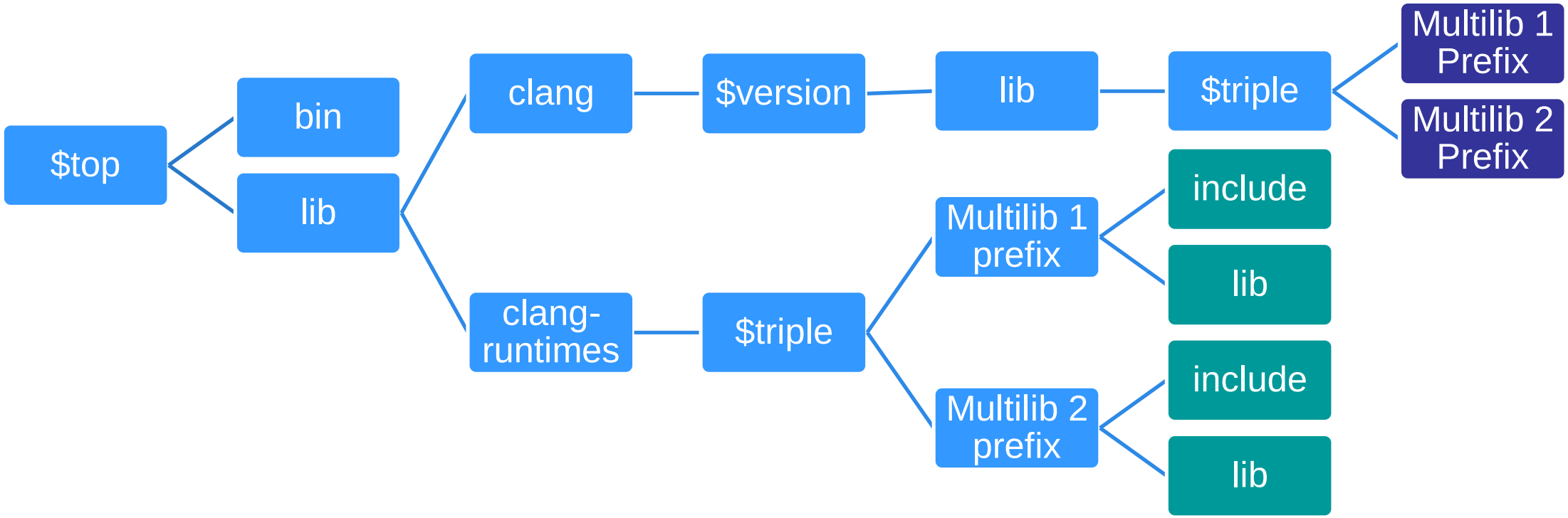
# Standalone Build

```
$ for multilib in all_multilibs; do
  $ cd /path/to/build/$multilib
  $ cmake …. $src/$library
  $ make
  $ make install
  $ make check-xyz
done
```

# Build Order Revisited



# Layout



# Linker

- LLD
  - Linker Script Issues

# Looking Ahead …

Freely available LLVM toolchain for RISCV.

https://www.mentor.com/embedded-software/toolchain-services/codebench-lite-downloads

Review upcoming patches.

*Thank you!*

Mentor®

A Siemens Business

www.mentor.com