The LLVM Embedded Toolchain for Arm

Peter Smith (peter.smith@arm.com)

16.09.2021

Embedded Toolchain?

- Embedded toolchain targets bare-metal systems with no assumption of an underlying OS.
- Cross-compilation.
- Static linking.
- Complicated linker scripts.
- Lots of library configurations.
- Optimization for code-size.
- Code debugged and tested on emulators, often with semihosting.
  - Semihosting uses the emulator/debugger to do IO.

The LLVM Embedded toolchain for Arm

- Modelled on the GNU Arm Embedded Toolchain.
- Compiler, LLVM binutils, compiler-rt and C/C++ libraries for supported platforms.
- Sample Linker scripts and example projects.
- Use tools and libraries from the LLVM project wherever possible.
- Open source components buildable by open source tools.
- Goal is to provide binary releases, currently a set of build scripts.

Project structure

The LLVM Embedded Toolchain for Arm includes:

- Build scripts written in Python.
- A file defining LLVM and newlib revisions to checkout.
- Temporary patches for LLVM and newlib.
- Linker scripts.
- Tests.
- Source code samples.
- Documentation.

Using an Embedded Toolchain

- Example from the GNU Arm Embedded Toolchain
  - Cortex-M0 with semihosting library.
  - specs files add additional configuration, semihosting and newlib-nano.
  - Multilib support selects Cortex-M0 compatible libraries.

```
$ arm-none-eabi-gcc semihost.c \
../../startup/startup_ARMCM0.s \
-mcpu=cortex-m0 \
--specs=nano.specs \
--specs=rdimon.specs \
-L ../../ldscripts –T gcc.ld \
-o semihost-CM0.axf
```

Multilib

- Selection of library path based on compilation options.
- Architecture, Arm/Thumb, Floating point, Floating point calling convention.

```
$ arm-none-eabi-gcc –print-multi-lib
arm/v5te/softfp;@marm@march=armv5te+fp@mfloat-abi=softfp
…
thumb/v7/nofp;@mthumb@march=armv7@mfloat-abi=soft
thumb/v7+fp/softfp;@mthumb@march=armv7+fp@mfloat-abi=softfp
thumb/v7+fp/hard;@mthumb@march=armv7+fp@mfloat-abi=hard
…
thumb/v6-m/nofp;@mthumb@march=armv6s-m@mfloat-abi=soft
…
thumb/v8-m.main+dp/hard;@mthumb@march=armv8-m.main+fp.dp@mfloat-abi=hard
```

Specs files

- Configuration files with limited conditionality
  - Semihosting and newlib nano.

```
$ cat lib/thumb/v6-m/nofp/rdimon.specs
%rename link_gcc_c_sequence                rdimon_link_gcc_c_sequence

*rdimon_libc:
%{!specs=nano.specs:-lc} %{specs=nano.specs:-lc_nano}

*rdimon_libgloss:
%{!specs=nano.specs:-lrdimon} %{specs=nano.specs:-lrdimon_nano}

*link_gcc_c_sequence:
%(rdimon_link_gcc_c_sequence) --start-group %G %(rdimon_libc)
%(rdimon_libgloss) --end-group

*startfile:
crti%O%s crtbegin%O%s %{!pg:rdimon-crt0%O%s} %{pg:rdimon-crt0%O%s}
```

Challenges in LLVM embedded Toolchain

No bare-metal Multilib or specs file support in LLVM

- Current solution: config files with downstream addition of $@ for current path.

```
$ clang --config armv8m.main_hard_fp_rdimon test.c -o test
```

armv8m.main_hard_fp_rdimon.cfg:

```
--target=armv8m.main-none-eabi -mfloat-abi=hard -march=armv8m.main+fp
-L$@/../lib/clang-runtimes/armv8m.main_hard_fp/lib
-isystem $@/../lib/clang-runtimes/armv8m.main_hard_fp/include
-Wl,-T$@/../lib/clang-runtimes/armv8m.main_hard_fp/base.ld
```

…

- Desired solution: multilib support in the Clang driver.

```
$ clang --target armv8m-none-eabi -march=armv8m.main+fp \
-mfloat-abi=hard test.c -o test
```

Challenges

- Compiling newlib with Clang
  - Some assembly accepted by GNU assembler but not clang integrated assembly.
  - An __attribute__((naked)) function incompatible with clang.
- Integrating libc++ and newlib
  - Missing aligned heap allocation functions in newlib (no C++17 aligned allocation).
- Libc++ configurations
  - Building without exceptions/RTTI and locale for small code-size.
- Windows builds
  - Currently using MinGW GCC to cross-compile the toolchain. Adds some dependencies on MinGW DLLs.
- Testing
  - Internal using Arm Fast Models.
  - QEMU can be used if it has a system model.

Where we are now and where do we want to go?

Current roadmap

Feature	Current status	Planned
Release types	Source only	Source and precompiled binaries
Released versions	LLVM 12, LLVM trunk	Each future LLVM major and minor release, LLVM trunk
Host support	x86-64 Linux and Windows	x86-64 Linux, Windows, Mac AArch64 Linux, Windows, Mac
C++ support	No exceptions, no RTTI No iostreams, no TLS	
Testing	Github-based CI for the build scripts. Internal nightly testing	CI for the build scripts and public build bots
Benchmarks	Internal, manual runs	

Working with the community

- Multilib support in bare-metal toolchains
  - GCC does this at build-time using a configuration file.
  - Clang multilib support outside bare-metal seems to hard-code paths.
  - Is there a better way to solve the problem?
- Potential for expanding config files to support specs file like features
  - Jury is out as to whether this is necessary.
- Buildbots that build and test compiler-rt for bare-metal platforms
  - Coverage of M and R profile upstream limited.
- Continuous integration reporting failures to upstream
  - Programs in LLVM Test suite may not be appropriate for embedded devices, well defined subset?
  - Something akin to Linaro's clang Linux kernel build bisection CI job?
- Docker containers to help reproduce failures
  - Contain toolchain and emulators.

References

- Questions on the talk https://github.com/ClangBuiltLinux/llvm-distributors-conf-2021/issues/17
- Link to repository https://github.com/ARM-software/LLVM-embedded-toolchain-for-Arm
  - Please use github issues for bug reports or any other technical feedback on the toolchain.
  - Code contributions welcome via pull requests.
- Linaro Connect presentation “The LLVM Embedded Toolchain for Arm, a new open source toolchain https://connect.linaro.org/resources/lvc21f/lvc21f-321/
  - A more general introduction for Linaro developers familiar with the GNU Arm Embedded Toolchain
- GNU Arm Embedded Toolchain https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm

arm

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה