SFO17-409 TSC OSS Toolchain Discussion

David A Rusling
Ryan S. Arnold, Maxim Kuvyrkov
Overview

● Toolchain work in Linaro
  ○ GCC
    ■ ARM GNU funding to TCWG and the effect on Linaro TCWG’s roadmap
    ■ Transition of GNU toolchain release to ARM in 2018 (august)
    ■ ARMv8.2
    ■ SVE upstream progress
      ● GDB SVE enablement moving forward
  ○ LLVM
    ■ ARMv8.2
    ■ LLVM growth roadmap
    ■ SVE upstream progress
  ○ ILP32
    ■ ILP32 toolchain progress update
  ○ FDPIC Toolchain

● Discussion
  ○ Does this all fit together?
  ○ Is there anything that we’re missing?
Key GNU Deliverables

1. **TCWG-1232** Link Time Optimization tuning for AArch64
2. **TCWG-64** Sign/Zero-Extension Elimination optimizations
3. **TCWG-1233** Investigate scalability of libgomp on SPEC CPU2017
4. **TCWG-1207** ILP32 Toolchain
5. **TCWG-159** GDB Kernel Awareness
6. **TCWG-1035** GDB target description rework for SVE enablement
7. **TCWG-1160**, **TCWG-1161** OpenOCD AArch64 & GDB Remote debugging interoperability
8. **TCWG-935** Automated regression testing of upstream branches
9. **TCWG-1231** Automated benchmarking of upstream branches
ARM funding of GNU work & Need for LLVM

- High volume of LLVM work needed to be done (see LLVM Growth Roadmap slides). Linaro Exec Mgmt was planning to propose TCWG transition to LLVM in the future. This initial proposal was shared with ARM.
- ARM expressed concern as there is still important GNU work Linaro can do especially on behalf of ARM enterprise workloads.
- ARM has decided to fund three existing (full-time equivalent) TCWG engineers to continue to focus on GNU for at least the next year.
- Linaro is requesting additional LLVM engineers (assignees and / or member engineers) from membership
Impacts of ARM GNU Funding on TCWG

● Inevitability of Linaro TCWG transition to LLVM means ARM needs to eventually take over GNU Toolchain releases. This work will start immediately with a soft-transition.

● This will enable Linaro to focus more on upstream deliverables and to begin to transition more engineers toward LLVM development in the future.

● ARM will provide a reduced set of host-target combinations.

● Overlap between ARM customer interests and Linaro member interest is very high when it comes to toolchain.
  ○ The near term impact is ILP32 toolchain integration work being done to create a x86_64 to AArch64 ILP32 ‘snapshot’ cross toolchain.
Transition GNU Toolchain Releases to ARM

- ARM will commit two member engineers to take over the GNU Toolchain release process internally from Linaro employees and assignees in a soft-transition.
- ARM member engineers will start to implement release infrastructure within ARM for future releases.
- ARM will release GCC 7 (and the maintenance GCC 6 release) through August 2018 with Linaro’s release infrastructure.
- ARM will release GCC 8 (and the maintenance GCC 7 release) from ARM’s release infrastructure starting August 2018.
Toolchain Release Transition Schedule

Linaro Release Infrastructure

2017.08 ➔ 2017.11 ➔ 2018.02 2018.05

ARM Release Infrastructure

2018.08, 2018.11, 2019.02, 2019.05, ad infinitum

Transition to ARM Release Infrastructure

- GCC [6|7]-2017.08 - Linaro Employees/Assignees
  ○ Status quo process
  With Linaro Infrastructure

- GCC [6|7]-2017.11 - Linaro Employees/Assignees with ARM Member Engineers
  ○ Transition to ARM MEs, using Linaro process
  With Linaro Infrastructure

- GCC [6|7]-2018.02, 2018.05 - ARM Member Engineers
  ○ Training wheels off, using Linaro process
  With Linaro Infrastructure

- GCC [7|8]-2018.08, 2018.11, 2019.02, 2019.05, ad infinitum - ARM Engineers
  ○ All in-house in ARM, using ARM process
  With ARM Infrastructure
Key LLVM Deliverables

1. **TCWG-1236** Fix LLD for Android
   a. **TCWG-1210** Implement -fix-cortex-a53-843419 in lld

2. **TCWG-1183** FreeBSD linking with LLD (patches in upstream review)

3. Continue work on GlobalISel
   a. **TCWG-826** Implement GlobalISel support for AArch32
   b. **TCWG-1235** Monitor and benchmark GlobalISel support for AArch64 from Apple

4. **TCWG-1233** Investigate scalability of libomp on SPEC CPU2017

5. **TCWG-244, TCWG-668, TCWG-953, TCWG-1112** AArch32 and AArch64 LLVM buildbot hosting and maintenance

6. **TCWG-1083** AArch32 and AArch64 LLVM release management

7. **TCWG-1156** Compiler-rt build-system improvements
LLVM Growth Roadmap - GlobalISel

- Support vector data types and operations in GlobalISel
- MIR: Well defined, machine IR, with generic and target specific instr/regs
- Add links to IR snippets in MIR
- Validate AArch64 GlobalISel at -O0 before turning on as default
- Support ARM, Thumb, AArch32, hard-float, soft-float and other combinations in ARM 32-bit GlobalISel
LLVM Growth Roadmap - LLD

- Improve unit-testing and integration validation
- Benchmark link-time performance of AArch64 linker
- Enable LTO (Link Time Optimization)
- Features related to use as system linker in LLVM toolchain for Android
- Add missing linker script cases and command line options to ARM port.
- Add testing for embedded cases (\(\text{v}\{6-8\}\text{M}\))
- Support Build Attributes for interoperability with ARM linker object files
- Support Big-Endian
- Comparing to Gold/BFD in linking time, binary sizes, performance
LLVM Growth Roadmap - Vectorisation

- Loop and Scalar (SuperWord Level) - Pass the Instruction object to all cost functions, so we can do peephole
- VPlan vectorization framework to replace monolithic loop and scalar vectorisers to improve Single legalisation/cost/transformation phase
- outerloop vectorisation - ARM has identified “unroll-and-jam” as beneficial to CoreMark
- outerloop vectorisation - HPC workloads often benefit from outer-loop vectorisation
- outerloop vectorisation - At least unrolling the inner loop could provide benefit
LLVM Growth Roadmap - Lib C++ (and friends)

- System-wide validation to guarantee interoperability with GNU libs (libgcc, glibc)
- Used in FreeBSD for years in x86_64, has troubles on ARM - address known deficiencies.
- Fix locale trouble on ARMv7
- Implement validation of Lib C++ (and friends) on AArch64
LLVM Growth Roadmap - Cross-Compilation

- Sysroot detection - Clang’s header/lib/tools detection is based on poor heuristics
- Sysroot detection - Clang’s triple/env detection is based on poor heuristics
- Sysroot detection - Clang still defaults to “system header/lib/linker” on failure
- GNU compatibility - Most tests still rely on glibc, libgcc, binutils, but we can’t easily change versions (without changing the host’s OS)
- We should have cross-builds like GCC, we could start with QEMU
LLVM Growth Roadmap - Target Parser

- Most of the knowledge is duplicated in TableGen. Our target descriptions already have most of it, adding the rest would be simple.
- Create a new TableGen backend to produce the “string-tables”.
- Logic is duplicated between ARM and AArch64; design a “static class design” or investigate other alternatives.
- Merge target parser with Triple library - Triple class has some redundant logic/knowledge.
- Remove all parsing of architectural strings, even from Triple.
LLVM Growth Roadmap - LLVM Bugs

- Address existing bug backlog for AArch64 GlobalISel
- Address existing bug backlog for Chromium
- Address existing bug backlog for Android
- Address existing bug backlog for Linux/FreeBSD bugs
LLVM Growth Roadmap - Scheduler

- Overall improvement to generic ARM/AArch64 scheduler - completeness and accuracy.
LLVM Growth Roadmap - Target Descriptor

● "Cleanup targets have “features” which are usually abused for scheduling/isel poor choices
● We still have some “isLikeA9()” functions which are nothing “like A9” behaviour"
● Document target description strategies and best-practices
LLVM Growth Roadmap - Test-suite / Benchmarks

- Some tests are redundant, some are obsolete, needs a cleanup
- Some benchmarks are badly prepared/executed
- Would be nice to have more modern/HPC tests/benchmarks
LLVM Growth Roadmap - Benchmarking

- Comparing to GCC on ARM
- Comparing to LLVM on x86_64
- We need at least a monthly CI on LLVM for SPEC/EEMBC/HPC
- Address open performance analysis tasks in Linaro backlog
LLVM Growth Roadmap - Performance

• Address found performance analysis issues found in benchmarking tasks.
LLVM Growth Roadmap - Debug info enhancement

- fixing missing or wrong debug info for non-O0 optimization
- Improve debugging experience when debugging application with optimization
LLVM Growth Roadmap - Move the content of SLEEF into LLVM/parallel-libs

- [http://github.com/shibatch/sleef](http://github.com/shibatch/sleef), a target-independent library for vector math routines, with target-specific implementations
- The parallel-libs does not allow direct contribution from ARM, but the author of SLEEF has granted permission for SLEEF code to be ported to parallel-libs ([http://lists.llvm.org/pipermail/llvm-dev/2016-July/102254.html](http://lists.llvm.org/pipermail/llvm-dev/2016-July/102254.html))
  - Note that some of the non-ARM side of this work has already been started ([https://reviews.llvm.org/D24951](https://reviews.llvm.org/D24951)). Note also that ARM have a downstream implementation of SLEEF targeting SVE
LLVM Growth Roadmap - LoopVector Analysis

- Refactor LoopVectorAnalysis into a separate pass
  - Limited immediate benefit to the community, but helps future development of the vectorizer
TCWG preview-grade ILP32 toolchain

- Same process as for release-grade cross-toolchains
  - Automated build and test harness
  - Quarterly cadence
  - Hosted on x86_64-linux, i686-linux and i686-mingw

- Same components as in release toolchains
  - GCC 7.1, Binutils 2.28, GDB 8.0

- Community-supported branches for ILP32
  - Linux kernel: staging/ilp32-4.12
  - Glibc: arm/ilp32

- First upload
  - https://snapshots.linaro.org/components/toolchain/binaries/7.1-2017.08-rc1/aarch64-linux-gnu_ilp32/

- Next update in November 2017
Known ILP32 toolchain issues

- Status for 7.1-2017.08-rc1 upload
- Sanitizer support (ASAN, UBSAN, etc.)
  - Need porting to ILP32
  - Sanitizers do not build for ILP32
  - TCWG plans to estimate effort size on sanitizers, and, potentially, port them to ILP32
- Compiler testsuite failures
  - There are several dozen of failures compared to AArch64 LP64
  - TCWG will investigate and fix or file bugs upstream
- LTP failures in get*ent tests
  - Seems like a patch is missing on glibc/arm/ilp32 branch
  - Should be fixed in next upload
ILP32 Linux kernel branches

- ILP32 ABI is stable and will not change
  - Unless highly critical ABI problem is found

- Linux kernel ILP32 staging branches
  - git://git.kernel.org/pub/scm/linux/kernel/git/arm64/linux.git
  - staging/ilp32-4.12
  - staging/ilp32-4.13

- ARM64 kernel maintainer evaluates ILP32 adoption
  - every 6 months

- ILP32 Linux branch is planned to be either merged or abandoned within 2 years
ILP32 glibc branch

- ILP32 ABI is stable and will not change
  - Unless highly critical ABI problem is found

- Glibc ILP32 staging branch
  - [https://sourceware.org/git/?p=glibc.git;a=shortlog;h=refs/heads/arm/ilp32](https://sourceware.org/git/?p=glibc.git;a=shortlog;h=refs/heads/arm/ilp32)
  - arm/ilp32

- Glibc will merge ILP32 branch once kernel branch is merged

- Merging Glibc branch will NOT break compatibility
  - Symbol versions will not be updated
OpenEmbedded ILP32 Status

- [https://projects.linaro.org/browse/CTT-405](https://projects.linaro.org/browse/CTT-405)
- Minimal-ltp images for generic-armv8 target is available
- The rootfs has been validated on HiKey with upstream kernel
  - [https://git.kernel.org/pub/scm/linux/kernel/git/arm64/linux.git/log/?h=staging/ilp32-4.12](https://git.kernel.org/pub/scm/linux/kernel/git/arm64/linux.git/log/?h=staging/ilp32-4.12)
- Important packages are part of the image
  - Glibc-2.25
Debian ILP32 Bootstrap

- [https://projects.linaro.org/browse/CTT-157](https://projects.linaro.org/browse/CTT-157)
- Task to produce debootstrapable rootfs
- Current focus on debian stable to avoid moving-target hassles. Using ‘rebootstrap’ script for repeatable bootstrapping.
- All using updated triplet: aarch64-linux-gnu_ilp32
- Port status page: [https://wiki.debian.org/Arm64ilp32Port](https://wiki.debian.org/Arm64ilp32Port)
- Toolchains: debian-stable multiarch cross-toolchains for amd64 and arm64 in above repo. GCC-6, glibc2.24 based.
Debian ILP32 Bootstrap - 2

- Base packages: 141 out of 144 packages built. Almost enough to debootstrap filesystem.
- Patches for dpkg, binutils, gcc-6, glibc, linux, libatomic-ops, openssl, libgpg-error, gmp, nspr, libgc, findutils, gnuTLS28, systemd, libssh2, openssl1.0, curl, perl.
- Git repo containing patches at: https://anonscm.debian.org/cgit/users/wookey/rebootstrap.git
  Branches for stable and unstable.
- Bootstrap works exactly as well on arm64 build hardware as it does on amd64, which is nice.
FDPIC Toolchain

- FDPIC allows to use shared libraries on MMU-less systems
- Linux kernel patches for no-MMU are expected to land in 4.14
- GCC 4.7-based FDPIC toolchain by ST
  - [https://github.com/mickael-guene/fdpic_manifest](https://github.com/mickael-guene/fdpic_manifest)
  - Not upstreamed

- Should TCWG update and upstream GNU support for FDPIC?
- Should TCWG develop and upstream LLVM support for FDPIC?
Thank You

#SFO17

SFO17 keynotes and videos on: connect.linaro.org
For further information: www.linaro.org