SFO15-110: Toolchain Collaboration

Ryan S. Arnold - Linaro
Matthew Gretton-Dann - ARM
High Level Roadmap

- **SFO15**
  - Sanitizers for Aarch64
  - LLDB Enablement
  - Benchmark Automation
  - GCC Auto-vectorization Improvements

- **BKK16**
  - Multiple GDB watchpoint support for ARM
  - GDB Linux Kernel Awareness
  - GDB Testsuite Coverage Improvements

- **LCU16**
  - Further LLDB enhancements
  - LLD for Aarch64
  - LLVM Reference Toolchain Integration
  - GDB Feature Parity w/ x86_64

**ONGOING**: Linaro Toolchain Quarterly Binary Releases, Monthly Linaro GCC Source releases (including backports), Periodic GNU Toolchain Package Releases (glibc, binutils, GDB), GNU Toolchain Maintenance (bug fixing), Monthly GCC Benchmarking, GCC Performance Development, System Library Performance, LLVM Maintainership, Toolchain Validation Automation Improvements, Release Validation, GCC Modularization
Delays

- Still discussing methods for supporting 39-bit, 42-bit, and 48-bit virtual memory addressibility in the Sanitizers by dynamic selection at runtime.
- 48-bit virtual memory addressibility will wait until Linux Kernel support is available.
- LLD for ARM/Aarch64 delayed due to community refactoring of linker backend.
- LTO broken in trunk.

Risks

- LLVM community maintenance taking significantly more time. ARM/Aarch64 bots are broken too often.
- Aarch64 systems of all types struggle under Toolchain validation, jeopardizing TCWG snapshots and releases.
- GCC 6 stage 1 won’t be open much longer.
glibc

- Buildbots for arm/Aarch64
- Reboot malloc improvements?
- General maintainership
- glibc release management
- micro-architecture optimization - best practices?
- Test out of tree to enable cross-testing?

binutils

- As-needed
GCC Performance & Features

- Continue with auto-vectorization improvements.
- Continue with fixing bugs found in Linaro Toolchains.
- Analyze large applications with LTO to look for missing opportunities.
- Permute through optimization flags with benchmarking using TACT in order to identify optimization opportunities.
- Intrinsics testing improvements are mostly upstream. Expand coverage?
- Restart modularization work now that GCC 6 Stage 1 is nearing the close of the development window. Restructuring work will cause less impact to backporting efforts.
GCC Features

- Restart modularization work now that GCC 6 Stage 1 is nearing the close of the development window. Restructuring work will cause less impact to backporting efforts.
- Addressing Linaro GCC bug backlog.
GDB

- Finishing up Multi-arch support in GDB for AArch64.
- Trace-points enabled for Aarch64 GDB.
- Fast trace-point support enabled for Aarch64 GDB.

- Work on non-stop debugging for Aarch64
- GDB Kernel Awareness is progressing. Community has been discussing best technical solution which happens to be lowest cost option.
- Will work on displaced Stepping support in Aarch64.
- Continue Multi-arch support in GDB for AArch64
- Improve ARM frame unwinding
- GDB commit level testing?
Backflip can easily automate high volumes of patch backports.

Our validation infrastructure structure can’t keep up.

We’re trying to scale up.

ARMv8.1. This will be either back-ported to FSF 5.2 or we’ll backport to Linaro branches.

Release scripting is in development to improve consistency.
Linaro Binary Releases

- Release-Candidate process is finding problems internally before release.
- Latest RC added support for:
  - bare-metal multilib support.
  - fixed library paths
  - armv8l-linux-gnueabihf targeted cross toolchains.
  - Python support in GDB for mingw32 and Linux.
- Plan is to make release candidate process more efficient in order to better hit quarterly dates.
- Linaro will hold back a release rather than release a known buggy toolchain.
- Turn on GDB & Glibc tests in release testing framework.
- Grow extended-validation framework.
Toolchain Benchmarking Automation

- This is a hard problem (see our Benchmarking 101 presentation)
- Two main Goals:
  - Release benchmarking
  - Backport benchmarking
- Secondary Goals:
  - Aide compiler developer benchmarking
  - Comparison/Parity Benchmarking
LLVM

- Continue maintainership on ARM/Aarch64 and keep buildbots green.
- Work on integration of LLVM hosted components (libraries, linkers, etc) into default release toolchains.
- Test combinations of components in continuous integration with buildbots.
- Start to look at LLVM performance improvements.

Can ARM consider an LLVM ‘member engineer’ working with Linaro to help with maintainership? General community contribution to arm/Aarch64 has increased and bots are breaking too often.
Sanitizers

- TSAN, ASAN, MSAN - base enablement nearly complete for 39-bit and 42-bit virtual memory addressibility.
- Buildbots are green for sanitizers.
- Enable dynamic selection of virtual memory addressibility (39-bit, 42-bit, and 48-bit).
- Procedure is to unify macro usage first across all sanitizers.
- KASAN? Do they need help?
**LLD**

- Initial porting for Aarch64 was making good progress. This was stopped by backend refactoring.
- Backend refactoring has now completed. We will resume LLD work on ARM/Aarch64.
- Previous work will serve as a reference point to check the correctness of the new backend.

**LLDB**

- Basic ARM/Aarch64 remote debugging enabled and demonstrated.
- This includes register print, breakpoint, single-step. Next step is to enable further features.
Agenda

- Introduction
- LLVM Update
  - Progress since June 2015
  - Plans for 4Q 2015
- GNU Toolchain
  - Progress since June 2015
  - Plans for 4Q 2015
Purpose of this Presentation

- Explain what ARM plans to work on, and what its current priorities are:
  - However, things are likely to change – so:
  - We will not achieve all this in the next six months,
  - And there will be other things we do do.
  - This is an update of the presentation given at the end of the 1H2015
  - If your plans include the same topics, or work in the same areas
  - Come and talk to us – we should work together
  - Preferably this conversation should happen in the appropriate upstream communities.

- If you feel that we’re doing the wrong thing
  - Come and talk to us – we’re happy to work out a better way forward

- If possible use the public mailing lists & bug databases to report issues
  - This is the best place to have the conversation about best ways forward.
Overview of Goals for 2015

- Support the Architecture & Cores
  - Teams are involved in development of new cores and architecture extensions
  - We will not discuss those here
  - However, we plan to upstream functionality as soon as possible after public announcements

- Support the Community

- Improve Performance:
  - Focusing on the whole range of architecture profiles.
  - Driven by a range of benchmarks, including industry standard CPU benchmarks.
  - We analyze benchmarks both:
    - for improvements we can make to the toolchains; and
    - to note any regressions and get them fixed in co-operation with the community
LLVM Toolchain
Progress in 3Q 2015 – features/architecture support/infrastructure

- **ACLE:**
  - Completed special register intrinsics support in clang
  - Added AArch32 fp16 Neon intrinsics
  - Allow __fp16 as function argument and return type on AArch32 and AArch64.

- Various bug fixes.

- Improvements to LNT:
  - Various improvements to daily report page
  - Recording hashes of binaries to know when binary didn’t change during noise analysis – in progress.
LLVM Toolchain
Progress in 3Q 2015 - optimizations

▪ Improved recognition of various min/max idioms & improved code generation for those – work coming to completion.
▪ Improvements to recognizing absolute difference idioms – In progress.
▪ Vectorization:
  ▪ Enabled interleaved access vectorization – shows big speedups on some image filters & other codes with similar access patterns – Done.
  ▪ Optimized run-time alias checking so it becomes worthwhile to vectorize in more cases – Done.
  ▪ Improving SCEV analysis of mixed-type expressions, resulting in e.g. loop trip counts being analyzed better and more vectorization in some cases – In progress.
  ▪ New Loop Vectorization library under development – cleans up the design of the vectorizer & should allow faster progress on vectorization in the future – In progress
▪ Improved Thumb-1 code generation for:
  ▪ loads and stores with offsets;
  ▪ array indexing using multiplies in the index calculation.
▪ Improving alias analysis for static global variables:
  ▪ various approaches analysed – an implementation under code review at the moment.
▪ Fixed a long compilation-time issue seen when compiling some files in AOSP ART.
LLVM Toolchain

Next to be looked at

- Completing optimizations in progress.
- Start work on a few more optimizations, including:
  - Improve vectorization factor for loops working on 8- & 16-bit integer data types.
GNU-A Toolchain
Progress in 3Q 2015

- Helped community release glibc 2.22
  - No known serious AArch64 bugs
- ABI Support:
  - Improved support for different Memory Models & Thread Local Storage
- Over-aligned parameter passing issue fixed (PR65956)
- AArch64 target attribute enabled
  - Improves LTO Support
- ARMv8.1-A enabled in binutils
- TLS for all AArch64 memory models supported in binutils
- Induction Variable Optimisations enhancement – overflow detection
GNU-RM Toolchain

Progress in 3Q 2015

- Released 4.9 Q2 toolchain.
  - Updated some inline assembly code in Newlib to work with old targets
    https://www.sourceware.org/ml/newlib/2015/msg00386.html
  - Avoid wastage caused by section size promotion
    https://answers.launchpad.net/gcc-arm-embedded/+question/262160
  - Local register variables don't work correctly with inline asm operands
    https://bugs.launchpad.net/gcc-arm-embedded/+bug/1411655

- Bug fixes
  - Newlib format specifier mismatch fix
    https://answers.launchpad.net/gcc-arm-embedded/+question/269083
  - Newlib macro mismatch fix
    https://answers.launchpad.net/gcc-arm-embedded/+question/269177
  - Assembler encoding bug fix.
    https://sourceware.org/ml/binutils/2015-08/msg00028.html
GNU Toolchain

Next Steps

- Glibc correctness for 2.23 release
- Complete ARMv8.1-A support across the toolchain
- fp16 support in GCC – Completing current Advanced SIMD intrinsics support for AArch64
- Cost model improvements for Cortex-A53 / Cortex-A57 (AArch64)
- Enhancing GCC vectorizer
- Enhancing GCC loop invariant optimisations
- Enable ARMv7-M execute-only
- Thumb-1 Unified syntax
- Improved ABI/ACLE compliance
- Improved use of tree matching
  - In particular to optimize CRC functions
- Improved selection of FP divide & multiply on Cortex-M
ARM Math Libraries

- In Q4 2015, ARM will offer a commercially supported set of 64-bit ARMv8 numerical libraries for scientific computing.
- Enable ARM partners’ computational kernels tuned for their SOC implementation.
  - Unified, validated framework
  - Optimized for ARMv8 microarchitectures, including A57, A72 and custom designs.
  - Development model allows for the production of ARM Math Libraries that are highly tuned for specific system configurations.
  - All implementations hosted on arm.com
- By the end of 2015, an HPC-specific ARM microsite will offer downloads, technical reference material, how-to-guides and third-party software recommendations for the scientific computing community.

2015 Focus:
- BLAS
- LAPACK
- FFT