The ARM Compiler Tackles Open Source Software

Javier Orensanz
Objectives

- Understand the challenges of building ARM Linux open source software with commercial tools

- Learn about GCC emulation mode and other useful features for building ARM Linux software

- Learn how the ARM Compiler can be used together with GNU tools to build ARM Linux software
Background: Platforms and Compilers

- Most platform software is locked to the platform compiler by
  - SQA and validation concerns
  - Language extensions
  - Convenience

**Platforms and Compilers**
- GNU
- LLVM
- ARM Compiler
- Platform Builder
Open source software adoption

- Previous project
- Current project
- Next project

- Consortium obtained open source
- Publically obtained open source

Source: VDC, 2010
Linux Support for Multiple Segments

- Support Linux kernel and GNU tools as foundation for range of products and vertical SW integration
  - Many low-level components and tools are common to those verticals
  - Track and optimize support for ARM CPU roadmap and key IP blocks

- Enable UIs and frameworks developed by OEMs
  - Rich graphics frameworks with optimized HW acceleration
Linux and GCC

- Historical connection between GCC and Linux
  - 1983 - GNU Project aimed at creating free, open source software
  - 1985 - Stallman created the GNU Compiler Collection (GCC)
  - 1992 - The Linux kernel was added

- GCC and Linux have evolved together over 25 years under the philosophy of open source
ARM Open Source Tools

- Engagement with the developer community for GNU tools
  - GNU core tools (GCC, GAS, GLD, binutils, eglibc)
    - Thumb®-2 support for improved performance and code size
    - Cortex™-A8 and Cortex-A9 performance tuning
  - Sponsoring CodeSourcery releases
    - Pre-built, pre-packaged GCC binaries
  - Adding Linux profiling
    - Oprofile
    - LTTng, ftrace, perf-events

- Promote Linux ABI for the ARM architecture
  - Ensures GCC and RVCT interworking
  - Enable debuggers and performance analyzers to work with any compilation tools
“But I Want to Use the ARM Compiler”

- Open source tools do not meet everybody’s needs
  - Difficult to use
  - No technical support
  - No commitment to fixing bugs
  - No integration

- Many companies prefer to use professional tools
  - Known development environment
  - Efficient development and lower opportunity cost
  - Risk reduction
Benefits of ARMCC

- Industry standard ARM compiler for over 20 years
- Tuned for smallest possible code
  - Unit cost savings
  - Better usage of L1 and L2 cache for optimal performance and power
- Best-in-class compilation for Thumb, floating point and NEON™
  - Important for Android™ and new Cortex processor-based SoCs
- Professional support and maintenance
  - Technical support, documentation, FAQs, etc
  - Patches and bug fixes on old compiler versions released for years
- Early access: co-developed with new cores
Development Tool Incompatibility

- Areas of incompatibility between GCC and professional compilers
  - Compiler
  - Libraries
  - Build Environment
  - Linker
  - Assembler

- Over the years ARM has extended the ARM Compiler to introduce compatibility with the GNU compilation tools
Compiler

- Language extensions
  - Variable names starting with $ are forbidden in C standard
  - But are supported by GCC. Example: $my_var
  - De-referencing a null pointer is undefined in C standard
  - Common use in Linux/Android. Strict compilers throw an error

- Intrinsics and pre-defined macros
  - e.g. NEON parallel processor
    - `uint32x2_t vadd_u32 (uint32x2_t, uint32x2_t)`

- C++ exceptions
  - Linux expects that C++ exceptions are handled in a particular way
  - Compiled object files must work with C++ exception-handling code
Libraries

- Linux requires GNU C/C++ libraries: glibc, libstdc++
  - Enables POSIX, C++ exception handling, and others
  - Glibc is ~ 1.8M lines of code (!!!)

- Professional compilers often provide simple implementations of the C/C++ library: Std I/O, file management, time/date, etc

- ARM’s approach is to support linking glibc and libstdc++ instead of rewriting them
  - Enables reuse of years of development
  - Ensures compatibility with future versions of Linux
Build Environment

- Most ARM code is cross-developed
  - Development tools (including compiler) on desktop computer
  - ARM processor running code on a separate target
  - Code download via JTAG, SSH, NTFS, etc

- Scratchbox
  - Linux cross-compilation toolkit (open source)
  - Compiler wrapper emulates native compilation
  - ARM provides a scratchbox plug-in to build Linux software using RVCT

- Command line compatibility
  - Support GNU compiler and linker command line options to reuse makefiles
**Linker**

- Define the memory layout of the build image
  - This method is often an integral part of Linux build environments
  - ARM Compiler: scatter-loading files
  - GNU Compiler: linker scripts

- Prelink
  - GNU utility that speeds up the loading of shared libraries
  - It embeds shared library information in the object file

```c
LOAD 0x10000000 0x003FFFFF
{
  EXT_FLASH 0x10000000 0x003FFFFF
  {
    *.o (RESET, +First)
    .ANY (+RO)
  }
  EXT_SRAM 0x20000000 0x03FFFFFF
  {
    phy_init.o (+RW, +ZI)
  }
  INT_RAM 0x200000 0x000FFFFF
  {
    .ANY(+RW +ZI)
  }
}
```
Assembler

- ARM/Thumb assembler file syntax is not defined by the ARM architecture
  - ARM and GNU assembler syntax is fundamentally different
  - Examples: area definitions, macros, data, constants

- Assembler files written for GNU Assembler (GAS) need to be assembled with it

- The ARM Compiler still cannot cope with inline assembler in C/C++ source files
  - The only way to support inline assembler is to “fall back” to GCC
Current ARM/GNU Compatibility

- The ARM Compiler has achieved very high compatibility with GNU Compiler
  - The ARM Compiler can be used today to build
    - Over 11,000 Debian packages and 5,000 Android packages

- Running the ARM Compiler in GNU emulation mode
  - Use the `--translate_gcc` and `--arm_linux_configure` command line switches

- Object ABI compatibility enables collaborative builds
Scenarios

- ARM Compiler & GCC compatibility means you can
  - Retain explicit search paths on the command line
  - Use a standard configuration of system search paths and libraries
  - Reuse existing make-files and command-line options

- You can use the ARM Compiler to
  - Build a standalone Linux application
  - Build static and shared libraries
  - Migrate an existing Linux application build using the ARM Compiler

- Fallback (roadmap)
  - If a source file cannot be built with the ARM Compiler, automatically invoke the GNU Compiler to keep building the project
Summary

- GNU are the reference tools for Linux platforms, although professional software tools provide advantages.

- The ARM Compiler toolchain has new features to ease building of ARM Linux software.

- Fallback mode will enable seamless building of ARM Linux software.
Thank You

Please visit www.arm.com for ARM related technical details

For any queries contact < Salesinfo-IN@arm.com >