



Release Note for Arm GNU Toolchain

11.3.Rel1

Non-Confidential

Copyright © 2022 Arm Limited (or its affiliates).
All rights reserved.

Issue

109845_11.3.Rel1_en



Release Note for Arm GNU Toolchain

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
11.3.Rel1	8 August 2022	Non-Confidential	11.3.Rel1 Release
11.2-2022.02	15 February 2022	Non-Confidential	11.2-2022.02 Release

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm

makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of this document shall prevail.

The validity, construction and performance of this notice shall be governed by English Law.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. Please follow Arm's trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on <https://support.developer.arm.com>

To provide feedback on the document, fill the following survey: <https://developer.arm.com/documentation-feedback-survey>.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Release Note for Arm GNU Toolchain 11.3.Rel1.....	6
--	---

1. Release Note for Arm GNU Toolchain 11.3.Rel1

Release Date: 8 August 2022

This is release 11.3.Rel1 of Arm GNU Toolchain.

Arm GNU Toolchain releases package pre-built binaries of GNU Toolchain for various Arm targets. These are community supported and come with no warranty. For more information, please visit the [arm Developer](#) page.

This release includes bare-metal and Linux toolchains for various hosts, as described in the [Host Support](#) section.

Changes since Arm GNU Toolchain 11.2-2022.02

Changes since Arm GNU Toolchain 11.2-2022.02:

- Updated GCC to source code based on version 11.3.
- Updated Binutils to source code based on version 2.38.
- Updated Glibc to version 2.35.
- Updated GDB to source code based on version 12.
- Fixed an issue where the compiler might generate an internal compiler error due to microarchitecture-specific instructions that might be unavailable on certain CPUs. This has been fixed for the bare-metal target toolchains.
- In the Windows hosted toolchains, GDB is provided without Python support.
- In the Linux hosted toolchains, GDB's Python support has been updated to Python 3.8.
- Changed the way Arm GNU Toolchain release versions are named. From this release onwards, the release version name is derived from GCC version it is based on, the quality level and the revision number. The format of the version name is as follows:

```
Arm GNU Toolchain <GCC Major Version>.<GCC Minor Version>.[<Feature>-]{Alp|Bet|  
Rel}<Revision>
```

Known Issues

Known issues:

1. In the Linux hosted toolchains, GDB is provided with Python support. In the Windows and macOS hosted toolchains, GDB is provided without Python support.
2. For the Linux target toolchains, executing the compiler might generate an internal compiler error due to certain microarchitecture-specific instructions that might be unavailable on certain CPUs.
3. When you decompress the windows packages, the decompression requests permission to overwrite certain files. This is because the files have similar names with different case, which

are treated as identical names on a Windows host. You can choose to overwrite the files with identical names.

4. Doing IPA on CMSE generates a linker error: The linker will error out when resulting object file contains a symbol for the clone function with the `__acle_se` prefix that has a non-local binding. Issue occurs when compiling binaries for M-profile Secure Extensions where the compiler may decide to clone a function with the `cmse_nonsecure_entry` attribute. Although cloning nonsecure entry functions is legal, as long as the clone is only used inside the secure application, the clone function itself should not be seen as a secure entry point and so it should not have the `__acle_se` prefix. A possible workaround for this is to add a 'noclone' attribute to functions with the 'cmse_nonsecure_entry'. This will prevent GCC from cloning such functions.
5. GCC can hang or crash if the input source code uses MVE Intrinsics polymorphic variants in a nested form. The depth of nesting that triggers this issue might vary depending on the host machine. This behaviour is observed when nesting 7 times or more on a high-end workstation. On less powerful machines, this behaviour might be observed with fewer levels of nesting. This issue is reported in https://gcc.gnu.org/bugzilla/show_bug.cgi?id=91937

Ask Questions

For any questions, please use the [Arm Community forums](#)

Report Bugs

Please report any bugs via the [Linaro Bugzilla](#) under “GNU Binary Toolchain” product.

Host support

Host	Host Identifier (package name)	Toolchain Target
Windows on IA-32 or x86_64 Windows 10 or later	mingw-w64-i686	AArch64 Bare-metal AArch64 Linux AArch32 Bare-metal AArch32 Linux hard-float
Linux on AArch64 These toolchains are built on and for Ubuntu 18.04 on AArch64, and will likely also be useable on OS versions: <ul style="list-style-type: none"> • later than Ubuntu 18.04 • RHEL8 	aarch64	AArch64 Bare-metal AArch32 Bare-metal AArch32 Linux hard-float

Host	Host Identifier (package name)	Toolchain Target
Linux on x86_64 These toolchains are built on and for RHEL7 on x86_64, and will likely also be useable on OS versions: <ul style="list-style-type: none"> RHEL8 Ubuntu 18.04 or later 	x86_64	AArch64 Bare-metal AArch64 Linux AArch64 Linux big-endian AArch32 Bare-metal AArch32 Linux hard-float
Mac OS X on x86_64 Mac OS X 10.15 or later Note: Toolchains described as [BETA] have not had the same level of testing as the other toolchains.	darwin-x86_64	[BETA] AArch64 Bare-metal AArch32 Bare-metal

Included Toolchains

The package names of the released GNU toolchain binaries have the following naming convention:

`arm-gnu-toolchain-<Release Version>-<Host>-<Target Triple>.tar.xz`

- In the following table, <Target Triple> is listed in parentheses in the second column as part of target description.
- For Windows, the binaries are provided in zip files and with installers.
- For Linux, the binaries are provided as tarball files.
- For Mac OS X, the binaries are provided as tarball files and pkg files.

Toolchain Package Name	Host OS / Target Description
arm-gnu-toolchain-11.3.rel1-aarch64-aarch64-none-elf.tar.xz	Host: AArch64 Linux Target: AArch64 bare-metal (aarch64-none-elf)
arm-gnu-toolchain-11.3.rel1-aarch64-arm-none-eabi.tar.xz	Host: AArch64 Linux Target: Arch32 bare-metal (arm-none-eabi)
arm-gnu-toolchain-11.3.rel1-aarch64-arm-none-linux-gnueabi.tar.xz	Host: AArch64 Linux Target: AArch32 GNU/Linux target with hard float. (arm-none-linux-gnueabi)

Toolchain Package Name	Host OS / Target Description
arm-gnu-toolchain-11.3.rel1-mingw-w64-i686-arm-none-eabi.zip	Host: Windows Target: AArch32 bare-metal (arm-none-eabi)
arm-gnu-toolchain-11.3.rel1-mingw-w64-i686-arm-none-eabi.exe	Host: Windows Target: AArch32 bare-metal (arm-none-eabi)
arm-gnu-toolchain-11.3.rel1-mingw-w64-i686-aarch64-none-elf.zip	Host: Windows Target: AArch64 bare-metal (aarch64-none-elf)
arm-gnu-toolchain-11.3.rel1-mingw-w64-i686-aarch64-none-elf.exe	Host: Windows Target: AArch64 bare-metal (aarch64-none-elf)
arm-gnu-toolchain-11.3.rel1-mingw-w64-i686-arm-none-linux-gnueabi.zip	Host: Windows Target: AArch32 GNU/Linux with hard float (arm-none-linux-gnueabi)
arm-gnu-toolchain-11.3.rel1-mingw-w64-i686-arm-none-linux-gnueabi.exe	Host: Windows Target: AArch32 GNU/Linux with hard float (arm-none-linux-gnueabi)
arm-gnu-toolchain-11.3.rel1-mingw-w64-i686-aarch64-none-linux-gnu.zip	Host: Windows Target: AArch64 GNU/Linux (aarch64-none-linux-gnu)
arm-gnu-toolchain-11.3.rel1-mingw-w64-i686-aarch64-none-linux-gnu.exe	Host: Windows Target: AArch64 GNU/Linux (aarch64-none-linux-gnu)
arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-elf.tar.xz	Host: x86_64 Linux Target: AArch64 bare-metal triple: aarch64-none-elf
arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-linux-gnu.tar.xz	Host: x86_64 Linux Target: AArch64 GNU/Linux (aarch64-none-linux-gnu)

Toolchain Package Name	Host OS / Target Description
arm-gnu-toolchain-11.3.rel1-x86_64-aarch64_be-none-linux-gnu.tar.xz	Host: x86_64 Linux Target: AArch64 GNU/Linux big-endian (aarch64_be-none-linux-gnu)
arm-gnu-toolchain-11.3.rel1-x86_64-arm-none-eabi.tar.xz	Host: x86_64 Linux Target: AArch32 bare-metal (arm-none-eabi)
arm-gnu-toolchain-11.3.rel1-x86_64-arm-none-linux-gnueabi.tar.xz	Host: x86_64 Linux Target: AArch32 GNU/Linux with hard float (arm-none-linux-gnueabi)
arm-gnu-toolchain-11.3.rel1-darwin-x86_64-aarch64-none-elf.tar.xz	Host: x86_64 Darwin Target: [BETA] AArch64 bare-metal triple: aarch64-none-elf (aarch64-none-elf)
arm-gnu-toolchain-11.3.rel1-darwin-x86_64-aarch64-none-elf.pkg	Host: x86_64 Darwin Target: [BETA] AArch64 bare-metal triple: aarch64-none-elf (aarch64-none-elf)
arm-gnu-toolchain-11.3.rel1-darwin-x86_64-arm-none-eabi.tar.xz	Host: x86_64 Darwin Target: AArch32 bare-metal (arm-none-eabi)
arm-gnu-toolchain-11.3.rel1-darwin-x86_64-arm-none-eabi.pkg	Host: x86_64 Darwin Target: AArch32 bare-metal (arm-none-eabi)

Released Files

Released files:

arm-gnu-toolchain-*.tar.xz	Toolchain binaries
arm-gnu-toolchain-*.zip	Zipped toolchain binaries for Windows
arm-gnu-toolchain-*.exe	Toolchain installer for Windows
arm-gnu-toolchain-*.pkg	Toolchain installer for Mac
arm-gnu-toolchain--src-snapshot-.tar.xz	Toolchain sources
arm-gnu-toolchain-*.src-manifest.txt	List of remote repositories and the revisions of the source code used for building the toolchain

arm-gnu-toolchain-*-abe-manifest.txt	Input files for the Linaro ABE build system.
*.asc	MD5 checksum files for sources and binaries
*.sha256asc	SHA256 checksum files for sources and binaries

Source Code

The sources for this release are provided in the source tar ball, arm-gnu-toolchain-src-snapshot-11.3.rel1.tar.xz, and includes the following items:

Project	Version	Repository/Branch/Revision
GCC	based on 11.3	git://gcc.gnu.org/git/gcc.git branch: vendors/ARM/heads/arm-11 revision: 4249a65c814287af667aa78789436d3fc618e80a
glibc	2.35	git://sourceware.org/git/glibc.git branch: master-2.35 revision: 0e5b239f45992e4b54c6f946ecb0c410afc8bb08
newlib	from August 2022	git://sourceware.org/git/newlib-cygwin.git revision: bfee9c6ab0c3c9a5742e84509d01ec6472aa62c4
newlib-nano		
binutils	based on 2.38	git://sourceware.org/git/binutils-gdb.git branch: binutils-2_38-branch revision: 5c0b4ee406035917d0e50aa138194fab57ae6bf8
GDB	based on 12	git://sourceware.org/git/binutils-gdb.git branch: gdb-12-branch revision: 7f70cce769c1eced62012b0529907ea957cb9c55
libexpat	based on 2.2.5	Sources are provided in release source tar ball
Linux Kernel		git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
libgmp	based on 6.2	Sources are provided in release source tar ball
libisl	based on 0.15	Sources are provided in release source tar ball
libmpfr	based on 3.1.6	Sources are provided in release source tar ball
libmpc	based on 1.0.3	Sources are provided in release source tar ball
libiconv	based on 1.15	Sources are provided in release source tar ball

Installation Instructions

Verifying the downloaded packages

You may check using MD5 checksum as follows:

```
$ md5sum --check arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-linux-gnu.tar.xz.asc
arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-linux-gnu.tar.xz: OK
```

Similarly for using SHA256 checksum, use the following instructions:

```
$ sha256sum --check arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-linux-  
gnu.tar.xz.sha256asc  
arm-gnu-toolchain-11.3.rel1-x86_64-aarch64-none-linux-gnu.tar.xz: OK
```

Installing on Linux

To install a toolchain on Linux, unpack the tarball to the preferred installation directory using the following instruction:

On x86_64:

```
$ tar xJf arm-gnu-toolchain-11.3.rel1-x86_64-<TRIPLE>.tar.xz -C /path/to/install/dir
```

On aarch64:

```
$ tar xJf arm-gnu-toolchain-11.3.rel1-aarch64-<TRIPLE>.tar.xz -C /path/to/install/  
dir
```

If you want to use gdb, then Python3.8 is required to be installed.

Installing on macOS

To install a toolchain on macOS, unpack the tarball to the preferred installation directory using the following instruction:

On darwin-x86_64:

```
$ tar xJf arm-gnu-toolchain-11.3.rel1-darwin-x86_64-<TRIPLE>.tar.xz -C /path/to/  
install/dir
```

Installing on Windows

To install the toolchain on Windows, you may choose to run the installer:

```
arm-gnu-toolchain-11.3.rel1-mingw-w64-i686-<TRIPLE>.exe
```

and follow the instructions. The installer can also be run on the command line. When run on the command-line, the following options can be set:

```
- /S Run in silent mode  
- /P Adds the installation bin directory to the system PATH  
- /R Adds an InstallFolder registry entry for the install.
```

For example, to install the tools silently, amend users PATH and add registry entry:

```
> arm-gnu-toolchain-11.3.rel1-mingw-w64-i686-<TRIPLE>.exe /S /P /R
```

Alternatively, you may use the zip package if you cannot run the installer. In order to do so, you must extract the content of the zip file at a preferred folder.

Arm recommends that you always install into the default installation location. Installing into a different location could expose your system to risks associated with weaker access permissions. For example, the access permissions inherited from the installation directory might allow non-admin users to modify the installed content. Arm recommends restricting write access, to any custom installation location, to only the users who are allowed to run the installer.

Known Dependencies

Known dependencies:

- GDB's Python support on Linux hosts requires installation of Python3.8, Python3.8-dev or libpython3.8.
- arm-none-linux-gnueabi-hf-gdb on Linux hosts requires liblzma.so.5.
- Toolchains dedicated for Windows host require mingw-w64 library, a complete runtime environment for GCC.
- The following executables in the Windows hosted toolchains:
 - aarch64-none-linux-gnu-dwp.exe
 - aarch64-none-linux-gnu-ld.gold.exe
 - arm-none-linux-gnueabi-hf-dwp
 - arm-none-linux-gnueabi-hf-ld.gold.exe

have additional dependencies on the following dlls:

- libwinpthread-1.dll
- libgcc_s_sjlj-1.dll
- libstdc++-6.dll
- libgcc_s_dw2-1.dll

You can obtain the required dlls from the MinGW-W64 GCC-8.1.0 packages from SourceForge:

- i686-posix-sjlj
- i686-posix-dwarf

Invoking GCC

On Linux and macOS, either invoke with the complete path like this:

```
$ <install-dir>/arm-gnu-toolchain-11.3.rel1-<HOST_ARCH>-aarch64-none-elf/bin/  
aarch64-none-elf-gcc
```

where, depending on the host, <HOST_ARCH> is one of:

```
x86_64  
aarch64  
darwin-x86_64
```

Or set the path and then invoke the toolchain like this:

```
$ export PATH=$PATH:<install-dir>/arm-gnu-toolchain-11.3.rel1-<HOST_ARCH>-aarch64-  
none-elf/bin  
$ aarch64-none-elf-gcc --version
```

On Windows, although the above approaches also work, it can be more convenient to either have the installer register environment variables, or run `<install-dir>\bin\gccvar.bat` to set environment variables for the current cmd.

For Windows zip package, after extracting the files, we can invoke the toolchain either using the complete path as follows:

```
<install-dir>\bin\aarch64-none-elf-gcc
```

or run `<install-dir>\bin\gccvar.bat` to set environment variables for the current cmd.

Architecture Options

This toolchain is built and optimized for Arm processors.

This section describes how to invoke GCC/G++ with the correct command-line options for variants of Cortex-A, Cortex-R and Cortex-M processors.

```
$ aarch64-none-elf-gcc [-mthumb] -mcpu=CPU[+extension...] -mfloat-abi=ABI
```

-mcpu:

For the permissible CPU names and extensions, see the GCC online manual: <https://gcc.gnu.org/onlinedocs/gcc-11.3.0/gcc/ARM-Options.html#index-mcpu-2>

Use the optional extension name with `-mcpu` to disable the extensions that are not present in the CPU of your choice.

By default, `-mfpv=auto` and this enables the compiler to automatically select the floating-pointing and Advanced SIMD instructions based on the `-mcpu` option and extension.

-mfloat-abi:

If floating-point or Advanced SIMD instructions are present, then use the `-mfloat-abi` option to control the floating-point ABI, or use `-mfloat-abi=soft` to disable floating-point and Advanced SIMD instructions.

For the permissible values of `-mfloat-abi`, see the GCC online manual: <https://gcc.gnu.org/onlinedocs/gcc-11.3.0/gcc/ARM-Options.html#index-mfloat-abi>

-mthumb:

When using processors that can execute in Arm state and Thumb state, use `-mthumb` to generate code for Thumb state.

Examples:

Examples with no floating-point and Advanced SIMD instructions:

```
$ arm-none-eabi-gcc -mcpu=cortex-m7+nofp
$ arm-none-eabi-gcc -mcpu=cortex-r5+nofp -mthumb
$ arm-none-eabi-gcc -mcpu=cortex-a53+nofp -mthumb
$ arm-none-eabi-gcc -mcpu=cortex-a57 -mfloat-abi=soft -mthumb
```

Examples with single-precision floating-point with soft-float ABI:

```
$ arm-none-eabi-gcc -mcpu=cortex-m7+nofp.dp -mfloat-abi=softfp
$ arm-none-eabi-gcc -mcpu=cortex-r5+nofp.dp -mfloat-abi=softfp -mthumb
```

Examples with single-precision floating-point with hard-float ABI:

```
$ arm-none-eabi-gcc -mcpu=cortex-m7+nofp.dp -mfloat-abi=hard
$ arm-none-eabi-gcc -mcpu=cortex-r5+nofp.dp -mfloat-abi=hard -mthumb
```

Examples with double-precision floating-point with soft-float ABI:

```
$ arm-none-eabi-gcc -mcpu=cortex-m7 -mfloat-abi=softfp
$ arm-none-eabi-gcc -mcpu=cortex-r5 -mfloat-abi=softfp -mthumb
```

Examples with double-precision floating-point with hard-float ABI:

```
$ arm-none-eabi-gcc -mcpu=cortex-m7 -mfloat-abi=hard
$ arm-none-eabi-gcc -mcpu=cortex-r5 -mfloat-abi=hard -mthumb
```

Example with floating-point and Advanced SIMD instructions with soft-float ABI:

```
$ arm-none-eabi-gcc -mcpu=cortex-a53 -mfloat-abi=softfp -mthumb
```

Example with floating-point and Advanced SIMD instructions with hard-float ABI:

```
$ arm-none-eabi-gcc -mcpu=cortex-a53 -mfloat-abi=hard -mthumb
```

Example with MVE and floating-point with soft-float ABI:

```
$ arm-none-eabi-gcc -mcpu=cortex-m55 -mfloat-abi=softfp
```

Example with MVE and floating-point with hard-float ABI:

```
$ arm-none-eabi-gcc -mcpu=cortex-m55 -mfloat-abi=hard
```

Available multilibs

Arm GNU Toolchain 11.3.Rel1 supports a set of multilibs in each toolchain.

To list all multilibs supported by any of the toolchain, use `--print-multi-lib` option. For example,

```
$ aarch64-none-elf-gcc --print-multi-lib
```

To check which multilib is selected by the arm-none-eabi toolchain based on `-mthumb`, `-mcpu`, `-mfp` and `-mfloat-abi` command line options:

```
$ aarch64-none-elf-gcc [-mthumb] -mcpu=CPU -mfp=FPU -mfloat-abi=ABI --print-multi-dir
```

For example:

```
$ arm-none-eabi-gcc -mcpu=cortex-a55 -mfp=auto -mfloat-abi=hard --print-multi-dir  
thumb/v8-a+simd/hard  
$ arm-none-eabi-gcc -mcpu=cortex-r5 -mfp=auto -mfloat-abi=softfp --print-multi-dir  
thumb/v7+fp/softfp  
$ arm-none-eabi-gcc -mcpu=cortex-m0 -mfp=auto -mfloat-abi=soft --print-multi-dir  
thumb/v6-m/nofp
```

C Libraries

This section only applies for arm-none-eabi targets.

Arm GNU Toolchain 11.3.Rel1 is released with two prebuilt C libraries based on newlib, for arm-none-eabi target.

One is the standard newlib and the other is newlib-nano for reduced code size. To distinguish them, the nano versions are renamed with `_nano` suffix:

```
libc.a --> libc_nano.a  
libg.a --> libg_nano.a
```

To use newlib-nano, users should provide additional gcc compile and link time option:

```
--specs=nano.specs
```

At compile time, a 'newlib.h' header file especially configured for newlib-nano will be used if `--specs=nano.specs` is passed to the compiler.

`nano.specs` also handles two additional gcc libraries: `libstdc++_nano.a` and `libsupc++_nano.a`, which are optimized for code size.

For example:

```
$ arm-none-eabi-gcc src.c --specs=nano.specs ${OTHER_OPTIONS}
```

This option can also work together with other specs options such as:

`-specs=rdimon.specs`

Please note that `-specs=nano.specs` is both a compiler and linker option. Be sure to include in both compiler and linker options if compiling and linking are separated.

Additional newlib-nano libraries usage

Formatted input/output of floating-point number are implemented as weak symbol. If you want to use `%f`, you have to pull in the symbol by explicitly specifying “-u” command option.

```
-u _scanf_float  
-u _printf_float
```

e.g. to output a float, the command line is like:

```
$ arm-none-eabi-gcc --specs=nano.specs -u _printf_float ${OTHER_LINK_OPTIONS}
```

Semi-hosting

Users can choose to use or not use semi-hosting by using the following instructions.

If you need semi-hosting, link as follows:

```
$ arm-none-eabi-gcc --specs=rdimon.specs ${OTHER_LINK_OPTIONS}
```

If you don't need semi-hosting or if you use retarget, link as follows:

```
$ arm-none-eabi-gcc --specs=nosys.specs ${OTHER_LINK_OPTIONS}
```

Linker scripts & start-up code

This section only applies for arm-none-eabi targets.

Latest update of linker scripts template and start-up code is available on <https://developer.arm.com/tools-and-software/embedded/cmsis>

Samples

This section only applies for arm-none-eabi targets.

Examples are available at:

```
<install-dir>/share/gcc-arm-none-eabi/samples
```

Read readme.txt under it for further information.

GDB Server for CMSIS-DAP based hardware debugger

This section only applies for arm-none-eabi targets.

CMSIS-DAP is the interface firmware for a Debug Unit that connects the Debug Port to USB. More detailed information can be found at <http://www.keil.com/support/man/docs/dapdebug/>.

A software GDB server is required for GDB to communicate with CMSIS-DAP based hardware debugger. The pyOCD is an implementation of such GDB server that is written in Python and under Apache License.

For those who are using this toolchain and have a board with CMSIS-DAP based debugger, the pyOCD is our recommended gdb server. More information can be found at <https://github.com/pyocd/pyOCD>.

Building Linux hosted toolchain from sources using Linaro's ABE

If you would like to build a toolchain yourself using the source revisions used for this release, you can do so using Linaro ABE (Advanced Build Environment) and provided ABE manifest files.

All the toolchains hosted on linux can be built using the steps provided below except for arm-gnu-toolchain-arm-none-eabi toolchain.

The example below shows how to build arm-gnu-toolchain-aarch64-none-elf toolchain using Linaro ABE build system.

Instructions

1. Install the dependencies ABE has a dependency on git-new-workdir and needs this tool to be installed in /usr/local/bin directory:

```
$ wget https://raw.githubusercontent.com/git/git/master/contrib/workdir/git-new-workdir
$ sudo mv git-new-workdir /usr/local/bin
$ sudo chmod +x /usr/local/bin/git-new-workdir
```

2. Clone ABE from the URL below and checkout the stable branch (see Getting ABE):

```
$ git clone https://git.linaro.org/toolchain/abe.git
```

3. Create the build directory and change to it. Any name for the directory will work:

```
$ mkdir build && cd build
```

4. Configure ABE (from the build directory):

```
$ ../abe/configure
```

5. Download the toolchain manifest file:

Download the toolchain manifest file from [arm Developer download page](https://developer.arm.com/-/media/Files/downloads/gnu/11.3.rel1/manifest/arm-gnu-toolchain-aarch64-none-elf-abe-manifest.txt), into the build folder, for the required toolchain, for example, arm-gnu-toolchain-aarch64-none-elf-abe-manifest.txt:

```
$ wget https://developer.arm.com/-/media/Files/downloads/gnu/11.3.rel1/manifest/arm-gnu-toolchain-aarch64-none-elf-abe-manifest.txt
```

6. Build toolchain (from the build directory):

```
$ ../abe/abe.sh --manifest arm-gnu-toolchain-aarch64-none-elf-abe-manifest.txt --build all
```

The built toolchain will be installed and available for use in the builds/destdir/x86_64-pc-linux-gnu/bin/ directory.

The example below shows how to build arm-gnu-toolchain-arm-none-eabi from sources using Linaro ABE build system.

Instructions

1. Install the dependencies ABE has a dependency on git-new-workdir and needs this tool to be installed in /usr/local/bin directory:

```
$ wget https://raw.githubusercontent.com/git/git/master/contrib/workdir/git-new-workdir
$ sudo mv git-new-workdir /usr/local/bin
$ sudo chmod +x /usr/local/bin/git-new-workdir
```

2. Clone ABE from the URL below and checkout the stable branch (see Getting ABE):

```
$ git clone https://git.linaro.org/toolchain/abe.git
```

3. Create the build directory and change to it:

```
$ mkdir build && cd build
```

4. Configure ABE (from the build directory):

```
$ ../abe/configure
```

5. Download the toolchain manifest file:

Download the toolchain manifest file `arm-gnu-toolchain-arm-none-eabi-abe-manifest.txt` from <https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/downloads>, into the build folder:

```
$ wget https://developer.arm.com/-/media/Files/downloads/gnu/11.3.rel1/manifest/arm-gnu-toolchain-arm-none-eabi-abe-manifest.txt
```

6. Build toolchain (from the build directory):

```
$ ../abe/abe.sh --manifest arm-gnu-toolchain-arm-none-eabi-abe-manifest.txt --build all >& log &
```

7. To build toolchain with newlib-nano configuration move out of build directory and create the `build_newlib` directory and change to it:

```
$ cd .. && mkdir build_newlib && cd build_newlib
```

8. Clone ABE from the URL below and checkout the stable branch (see Getting ABE):

```
$ git clone https://git.linaro.org/toolchain/abe.git
```

9. Configure ABE (from the `build_newlib` directory):

```
$ abe/configure
```

10. Download the toolchain manifest file:

Download the toolchain manifest file `arm-gnu-toolchain-arm-none-eabi-nano-abe-manifest.txt` from <https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/downloads>, into the `build_newlib` folder:

```
$ wget https://developer.arm.com/-/media/Files/downloads/gnu/11.3.rel1/manifest/arm-gnu-toolchain-arm-none-eabi-nano-abe-manifest.txt
```

11. Build toolchain (from the `build_newlib` directory):

```
$ abe/abe.sh --manifest arm-gnu-toolchain-arm-none-eabi-nano-abe-manifest.txt --build all >& log_nano
```

12. Move out of `newlib_nano` directory and download the `copy_nano_libraries.sh` script:

Download the `copy_nano_libraries.sh` script from <https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/downloads>, to the folder above `build_newlib` directory:

```
$ cd .. && wget https://developer.arm.com/-/media/Files/downloads/gnu/11.3.rel1/manifest/copy_nano_libraries.sh
```

13. Copy the newlib nano header and newlib nano libraries build in build_newlib folder to build folder and change to build folder:

```
$ ./copy_nano_libraries.sh && cd build
```

The built arm-none-eabi toolchain will be installed and available for use in the builds/destdir/x86_64-pc-linux-gnu/bin/ directory.