Using Snapdragon LLVM ARM Compiler 6.0.9 with Android NDK:

The Snapdragon LLVM ARM Compiler 6.0.9 can be used as a drop-in replacement for LLVM shipped as part of the Android NDK.

1. Snapdragon LLVM ARM Compiler 6.0.9 has been verified to work with Android NDK versions r17 for Windows (64-bit) and Linux (64-bit)

2. On Windows, it is assumed that the user has Cygwin setup. The examples below need to be adjusted to follow the Windows path specifications if Cygwin is not used.

3. On Windows and Linux it is recommended that you extract the Android NDK under a directory which *does not* contain spaces, like:

C:\android-ndk-r17
/local/mnt/workspace/android-ndk-r17

This directory is referred to as <NDK_ROOT> in this README.

On Windows, extract Snapdragon-llvm-6.0.9-windows64.zip under <NDK_ROOT>.

On Linux, extract Snapdragon-llvm-6.0.9-linux64.tar.gz under <NDK_ROOT>.

4. The following toolchains would be used for linking, by default:
For ARMv8 32-bit --> arm-linux-androideabi-4.9
For AArch64 --> aarch64-linux-android-4.9

5. On Windows, in order to avoid errors due to missing MSVC Redistributable DLLs, make sure you set PATH to the Snapdragon LLVM bin directory as follows:
export PATH=<NDK_ROOT>/toolchains/llvm-Snapdragon_LLVM_for_Android_6.0/prebuilt/windows-x86_64/bin:$PATH

6. The Snapdragon LLVM ARM 6.0.9 toolchain is designed to work with both 32-bit and 64-bit versions of the NDK as described below:

For generating ARMv8 32-bit code, invoke your compilation line as follows:

ndk-build NDK_TOOLCHAIN_VERSION=snapdragonclang APP_ABI="armeabi-v7a" -C <some_project>

For generating AArch64 (64-bit) code, invoke your compilation line as follows:

ndk-build NDK_TOOLCHAIN_VERSION=snapdragonclang APP_ABI="arm64-v8a" -C <some_project>

7. If you want to specify your custom flags to the compiler in order to override the default flags you can use the flag APP_CFLAGS as follows:

ndk-build NDK_TOOLCHAIN_VERSION=snapdragonclang APP_ABI="armeabi-v7a" \
APP_CFLAGS="-O3" -C <some_project>

Similarly, to specify custom flags for the linker you can use the
flag APP_LDFLAGS.

8. We STRONGLY RECOMMEND the following command line flags, to be set through APP_CFLAGS, for best performance. These options ensure that all high performance optimization features in the Snapdragon LLVM compiler are enabled to deliver maximum performance in 32-bit and 64-bit modes. For this release, if you continue to use the default Android NDK compatible flags, you may see performance regression.

If your project does not require precise math, please set
APP_CFLAGS="-Ofast -mcpu=cortex-a57"

If your project requires IEEE 754 floating point compliance, please set
APP_CFLAGS="-O3 -mcpu=cortex-a57"

9. A standalone toolchain for the Android NDK environment using the Snapdragon LLVM ARM compiler can be created using the make_standalone_toolchain utility. Note: For NDK r17, the default make_standalone_toolchain.py cannot be used with Snapdragon LLVM toolchain to create a standalone toolchain. So we provide a custom script called make_standalone_toolchain_snapdragon_llvm.py.

For example, to create a standalone toolchain for Linux 64-bit environment, the following commands can be used:

For ARMv8 32-bit:

<NDK_ROOT>/build/tools/make_standalone_toolchain_snapdragon_llvm.py \
--arch arm --api 28 --install-dir <some_dir>

The above command line specifies that you are targeting Android API level 28. If you do not specify the API level, the default will be set to the minimum supported level for the given architecture (currently 14 for 32-bit architectures).

For AArch64:

<NDK_ROOT>/build/tools/make_standalone_toolchain_snapdragon_llvm.py \
--arch arm64 --api 28 --install-dir <some_dir>

The above command line specifies that you are targeting Android API level 28. If you do not specify the API level, the default will be set to the minimum supported level for the given architecture (currently 21 for 64-bit architectures).

10. By default, the ndk-build tool will build an application for all the valid targets (viz. ARMv8 32-bit, AARCH64, X86). This will result in compilation errors for X86 targets since the Snapdragon LLVM ARM Compiler 6.0.9 cannot generate code for these targets. The errors can be avoided if the user explicitly passes APP_ABI="armeabi-v7a" or APP_ABI="arm64-v8a" when using the Snapdragon LLVM ARM Compiler 6.0.9. However, in situations where the build system cannot be changed, and hence these flags cannot be set, we provide the following wrappers for x86 in order to avoid compilation errors:

x86-snapdragonclang
x86_64-snapdragonclang

These wrappers are exact copies of their Clang versions which are distributed as part of the Android NDK. These wrappers would simply invoke the LLVM compiler that comes with NDK (for X86 targets only), thus avoiding compilation errors when using the Snapdragon LLVM ARM Compiler 6.0.9.

Note: The Snapdragon LLVM ARM Compiler 6.0.9 would be invoked for ARMv8 32-bit and AARCH64 targets.

Contacts & Bug Reporting
http://developer.qualcomm.com/llvm-forum