LLVM COMPILER INFRASTRUCTURE

# How To Build On ARM

## Introduction

This document contains information about building/testing LLVM and Clang on an ARM machine.

This document is *NOT* tailored to help you cross-compile LLVM/Clang to ARM on another architecture, for example an x86_64 machine. To find out more about cross-compiling, please check How To Cross-Compile Clang/LLVM using Clang/LLVM.

## Notes On Building LLVM/Clang on ARM

Here are some notes on building/testing LLVM/Clang on ARM. Note that ARM encompasses a wide variety of CPUs; this advice is primarily based on the ARMv6 and ARMv7 architectures and may be inapplicable to older chips.

1. The most popular Linaro/Ubuntu OS's for ARM boards, e.g., the Pandaboard, have become hard-float platforms. There are a number of choices when using CMake. Autoconf usage is deprecated as of 3.8.

   Building LLVM/Clang in `Relese` mode is preferred since it consumes a lot less memory. Otherwise, the building process will very likely fail due to insufficient memory. It's also a lot quicker to only build the relevant back-ends (ARM and AArch64), since it's very unlikely that you'll use an ARM board to cross-compile to other arches. If you're running Compiler-RT tests, also include the x86 back-end, or some tests will fail.

   ```
   cmake $LLVM_SRC_DIR -DCMAKE_BUILD_TYPE=Release \
                       -DLLVM_TARGETS_TO_BUILD="ARM;X86;AArch64"
   ```

   Other options you can use are:

   ```
   Use Ninja instead of Make: "-G Ninja"
   Build with assertions on: "-DLLVM_ENABLE_ASSERTIONS=True"
   Force Python2: "-DPYTHON_EXECUTABLE=/usr/bin/python2"
   Local (non-sudo) install path: "-DCMAKE_INSTALL_PREFIX=$HOME/llvm/instal"
   CPU flags: "DCMAKE_C_FLAGS=-mcpu=cortex-a15" (same for CXX_FLAGS)
   ```

   After that, just typing `make -jN` or `ninja` will build everything. `make -jN check-all` or `ninja check-all` will run all compiler tests. For running the test suite, please refer to LLVM Testing Infrastructure Guide.

2. If you are building LLVM/Clang on an ARM board with 1G of memory or less, please use `gold` rather then GNU `ld`. In any case it is probably a good idea to set up a swap partition, too.

   ```
   $ sudo ln -sf /usr/bin/ld /usr/bin/ld.gold
   ```

3. ARM development boards can be unstable and you may experience that cores are disappearing, caches being flushed on every big.LITTLE switch, and other similar issues. To help ease the effect of this, set the Linux scheduler to "performance" on **all** cores using this

little script:

```
# The code below requires the package 'cpufrequtils' to be installed.
for ((cpu=0; cpu<`grep -c proc /proc/cpuinfo`; cpu++)); do
    sudo cpufreq-set -c $cpu -g performance
done
```

Remember to turn that off after the build, or you may risk burning your CPU. Most modern kernels don’t need that, so only use it if you have problems.

4. Running the build on SD cards is ok, but they are more prone to failures than good quality USB sticks, and those are more prone to failures than external hard-drives (those are also a lot faster). So, at least, you should consider to buy a fast USB stick. On systems with a fast eMMC, that’s a good option too.
5. Make sure you have a decent power supply (dozens of dollars worth) that can provide *at least* 4 amperes, this is especially important if you use USB devices with your board. Externally powered USB/SATA harddrives are even better than having a good power supply.