LLVM/Clang and ARM 32-bit

Stefan Agner
Cross compiling with LLVM/Clang

- LLVM/Clang can cross compile by default (multiple backends)
  
  ```
  llc --version
  ```

- Currently, only compiling is done by LLVM/Clang
  - GNU cross-compiler toolchain with assembler/linker required

- Environment setup as usual (cross compiler in PATH!)
  
  ```
  export ARCH=arm
  export CROSS_COMPILE=arm-linux-gnueabihf-
  ```
Cross compiling with LLVM/Clang

● Compile using
  make CC=clang HOSTCC=clang multi_v7_defconfig
  make CC=clang HOSTCC=clang nconfig
  make CC=clang HOSTCC=clang -j4

● v4.18 adds compiler flag checks in Kbuild
  ○ hence CC/HOSTCC is required at config time!

● If using distro LLVM/Clang, add symlink in cross compiler bin dir [why?]
  cd ~/gcc-linaro-7.3.1-2018.05-x86_64_arm-linux-gnueabihf/bin
  ln -s /usr/bin/clang clang
Brief history LLVM/Clang Linux

- 2012-2014: LLVM Linux (Linux Foundation project)
  - Behan Webster, Jan-Simon Möller, Mark Charlebois
- 2015-: Compiled kernels/rebased the patches
  - [https://blog.printk.io/2015/03/cross-compile-linux-for-arm-using-llvm-clang-on-arch-linux/](https://blog.printk.io/2015/03/cross-compile-linux-for-arm-using-llvm-clang-on-arch-linux/)
- 2017: Google Android team pushing upstream 2017
  - Matthias Kaehlcke, Nick Desaulniers, Miguel Ojeda, Sedat Dilek
- 2017: Pushed fixes for build errors/warnings
  - E.g. build error for ARM in MPI
  - Lots of warnings: e.g. implicit conversion from enumeration
- 2018: Initial complete support for ARM 32-bit
Why?

- Competition etc....
- Prints really useful warning:

```c
drivers/gpu/drm/tegra/dc.c:408:18: warning: variable 'tegra124_primary_formats' is not needed and will not be emitted [-Wunused-internal-declaration]
static const u32 tegra124_primary_formats[] = {
  ^

drivers/gpu/drm/tegra/dc.c:835:18: warning: variable 'tegra124_overlay_formats' is not needed and will not be emitted [-Wunused-internal-declaration]
static const u32 tegra124_overlay_formats[] = {
  ^
```

2 warnings generated.
Upstream ARM 32-bit state

- v4.18 multi_v7_defconfig-CONFIG_EFI

- v4.19-rc3 currently broken (missing __naked preprocessor define)
  - Patch underway: https://lkml.org/lkml/2018/9/10/101
Known issues/Future work

● Function tracing fails to link
  ○ https://www.spinics.net/lists/arm-kernel/msg671262.html
  ○ Work ongoing: https://github.com/ClangBuiltLinux/linux/issues/35

● ARMv6 fails to build
  /tmp/empty-96a4d6.s: Assembler messages:
  /tmp/empty-96a4d6.s:4: Error: unknown cpu `arm1176j-s'
  ○ Assembler file contains
    .cpu arm1176j-s
  ○ LLVM/Clang issue? https://reviews.llvm.org/D18086
  ○ https://github.com/ClangBuiltLinux/linux/issues/55
Known issues/Future work

- Disable features/CPUs (currently) not supported with LLVM/Clang
  - ARMv5/ARMv6/Big Endian ([https://github.com/ClangBuiltLinux/linux/issues/57](https://github.com/ClangBuiltLinux/linux/issues/57))
  - Kconfig symbols for compiler/compiler version are very helpful!

- Use integrated assembler
  - Requires ARM unified syntax...

- Making use of static analysis tools/instrumentation
Debugging Techniques

- Compile single threaded & verbose
  ```
  make CC=clang HOSTCC=clang -j1 V=1
  ```
- Invoke the compiler manually verbose/or through CC
  ```
  clang ... -v
  make CC="clang -v" HOSTCC=clang -j1 V=1
  ```
- To retain intermediate files use `-save-temps`
- Debug compiler flag detection
  - Edit scripts/Kbuild.include to echo command before execute (cc-option)
  - Better alternative?
Thank you!

Stefan Agner
stefan@agner.ch