Cross Compilation

EC EN 427

BYU Electrical & Computer Engineering
IRA A. FULTON COLLEGE OF ENGINEERING

Executables

What happens if we build the `hello_world` application and try to run it on our computer instead of the PYNQ board?

The file utility

- https://man7.org/linux/man-pages/man1/file.1.html

Compiled for my computer: (instructor/lec/cross-compiling/a.out)

- ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=a6d414c44dcc5175caca410ea6fb71426b48a6ea, for GNU/Linux 3.2.0, not stripped

Compiled for the PYNQ board (apps/helloworld/helloworld)

- ELF 32-bit LSB pie executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 3.2.0, with debug_info, not stripped

A binary makes three promises

1. ISA (Instruction set architecture)
   - Which instructions the CPU understands
   - E.g. x86, x86_64, ARM, ARM64, RISC-V

2. Operating System
   - Syscalls, file format, loader
   - E.g. Linux / ELF, Windows / PE (.exe)

3. Application Binary Interface
   - How code interacts at the binary level
   - How functions pass arguments and return values
   - Whether floating-point arguments go in FPU registers or integer registers

Let's compare binaries: objdump

- We can use "objdump -d" to disassemble the executable
- Compare hello_world executables

Cross Compiling

- **Cross-compiling:** is the process of building executable software on one system (the host) that is designed to run on a completely different system (the target) with different hardware or an operating system

- This is commonly done in embedded systems. Why?
  - **Necessity.** We may not be able to run a compiler on the target device.
    - E.g. Arduino lacks processing capability, enough RAM
    - Bare metal systems have no operating system, impractical to run compiler
  - **Convenience.** Host computer may be faster, more memory, parallel builds, have access to better toolchains, etc.
    - You can compile your software on your PYNQ board, but it's much slower

The toolchain name & compiler triple

- What compiler are we using?
  - `gcc-arm-11.2-2022.02-x86_64-arm-none-linux-gnueabihf`
  - (When you run the make target, it downloads this from the ARM website)

gcc-arm-11.2-2022.02-x86_64-arm-none-linux-gnueabihf

Product & Version

Host: Runs on your x86_64 computer

ISA (Instruction Set Architecture) & Vendor, e.g. x86_64-apple

Operating System

ABI:
- gnu = GNU C library
- eabi = Arm EABI calling conventions
- hf = Hard-float (There are hardware floating point units versus doing floating point in software libraries)

Sysroot & glibc

- A sysroot is a directory that stands in for the root filesystem of a target system. Inside it you find the usual layout — usr/include, usr/lib, lib — populated with the headers and libraries belonging to *that* system rather than the machine you're working on.

```
tools/gcc-arm-11.2-2022.02-x86_64-arm-none-linux-gnueabihf/bin/arm-none-linux-gnueabihf-g++ -print-sysroot
```

- Important part of sysroot: Link-time libraries (libc: C library, libgcc: GCC library)

```
tools/gcc-arm-11.2-2022.02-x86_64-arm-none-linux-gnueabihf/bin/arm-none-linux-gnueabihf-readelf -V userspace/build/apps/helloworld/helloworld
```

  - GLIBC_2.34 → The system we run on must have at least version 2.34 of GNU C Library

  - Have you ever cross-compiled an executable and run into a message like this?

```
version 'GLIBC_2.34' not found
```