Cross Toolchain for Raspberry Pi

Peng-Sheng Chen

Spring, 2018

What can You Learn?

- 由 source code開始，自己編譯支援Raspberry Pi相關的cross compiler、cross assembler、cross linker等工具

Introduction

- **A total solution for embedded system development**
  - Hardware => SOC, MCU, peripheral…
  - Software => software-development tools, applications
- **Software-development tools**
  - Compiler, assembler, linker, debugger, C standard library, IDE, …
  - Commercial => Very high cost
  - Open source => Free. Users can modify, distribute, and use the software

Software-Development Tools (Open Source)

Stable

Robust

Popular

Portable
(Embedded system)

→ GNU

GNU Binutils

- The GNU binutils is a collection of binary tools
  - ld - the GNU linker
  - as - the GNU assembler
  - Other binary tools
    - ar - A utility for creating, modifying and extracting from archives
    - gprof - Displays profiling information
    - objcopy - Copys and translates object files
    - objdump - Displays information from object files (disassembler)
    - …
- Easy to port binutils to other platforms
- Suitable for embedded-system development

GNU C/C++ Compiler

- Retargetable
- Available from the Free Software Foundation
  - GPL Licensed
  - Free
- Written by *Richard Stallman* originally (FSF)
- Front-end, back-end system with support for many of each
- Code quality is better than some known compiler, very close to DEC's compiler on Alpha
- Supplied as vendor compiler by NeXT, BeOS, Linux, BSD

Front Ends

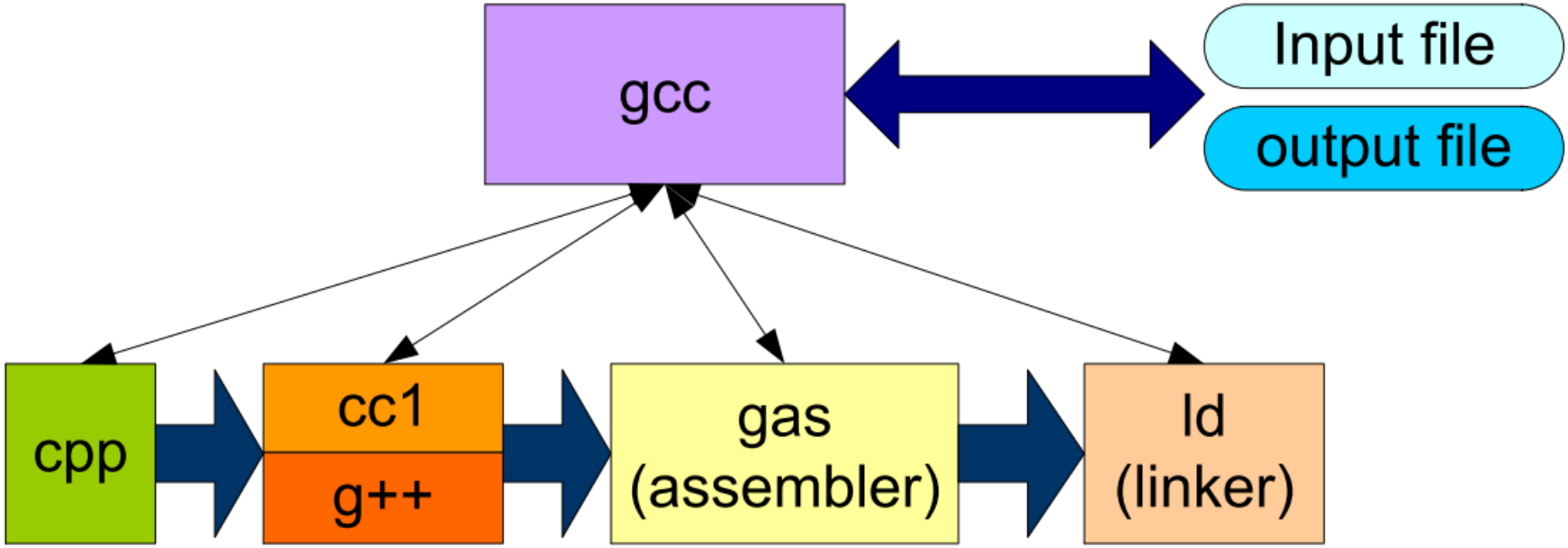
- C、C++、Objective C
- Ada 95 (GNAT)
- Fortran77、Fortran95
- Pascal
- Modula-2、Modula-3
- Java (supported from gcc 3.0)
  - Java->Bytecode, Bytecode->native code
  - Java->Native code
- Cobol
- Chill (Cygnus, a language in the Modula II family used in European telecommunications)

Machines Supported by GCC

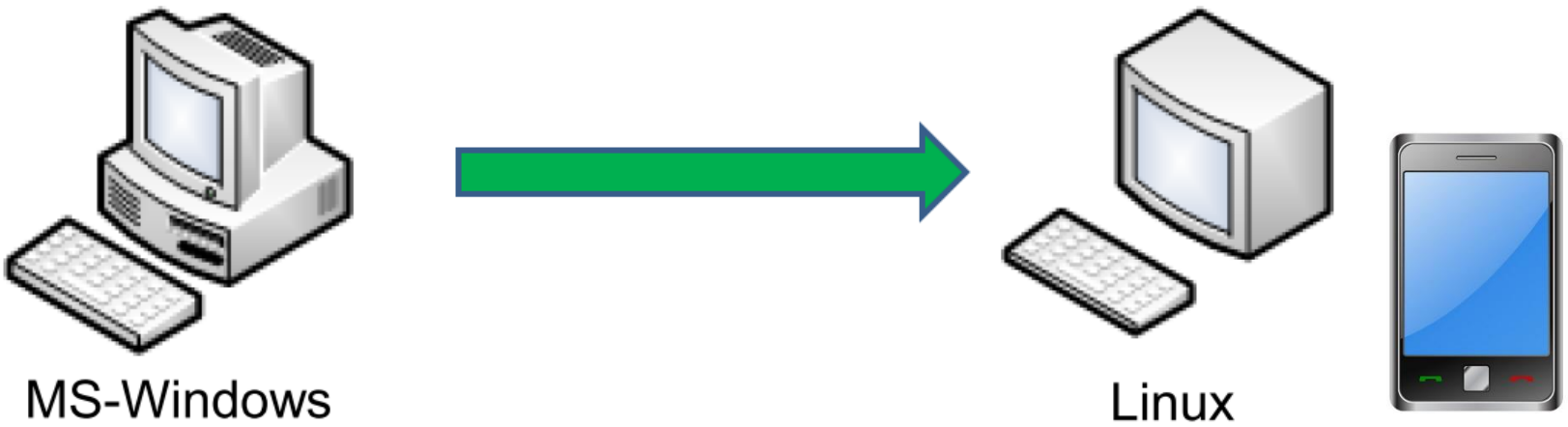
- Essentially all machines in widespread.
- ARM
- Alpha (DEC)
- Intel x86 Families、i860、i960
- Motorola 680x0、68C11、DSP56000
- Hitachi SH、H8300
- MIPS、IBM PowerPC、HP PA-RISC、SUN SPARC
- Intel IA64, AMD x86-64
- Cell processor (used by PS3)
- …

More than 30 architectures

Compilation



- Q1:
  - Build a cross compiler which executes on MS-Windows and produces Linux executable code

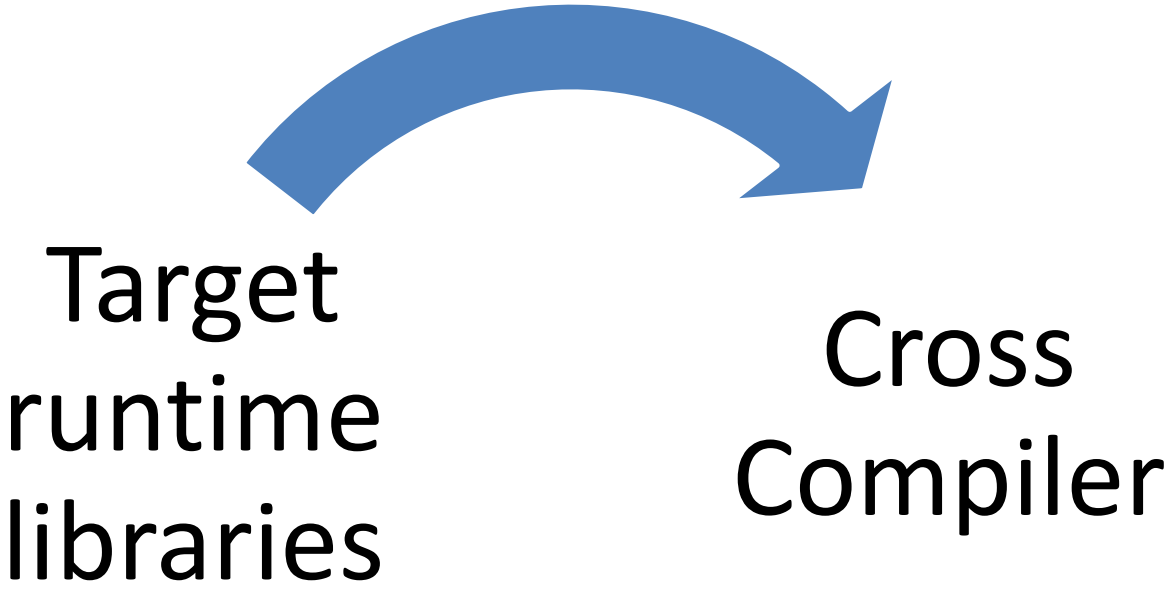


Target machine

Machine Classification

- **Build machine**
  - **The machine which builds cross-toolchains**
- **Host machine**
  - **The machine which cross-toolchains will execute on**
- **Target machine**
  - **The machine which cross-toolchains will produce output for**

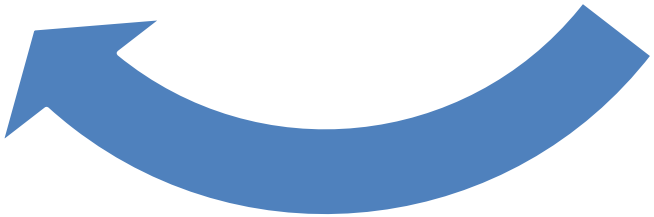
Cross Compiler



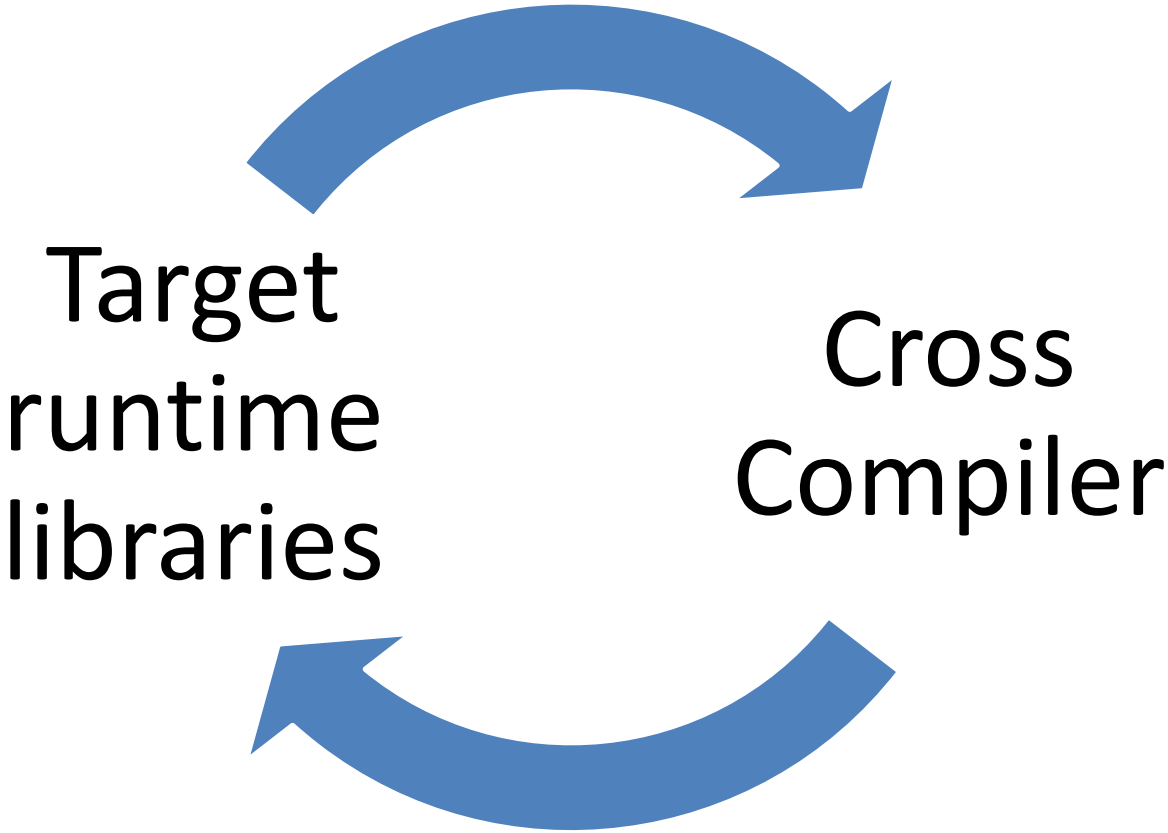
Cross Compiler

Target runtime libraries

Cross Compiler



Cross Compiler



Cross Compiler

實驗步驟

實驗步驟 (1)

- 在PC上安裝Linux作業系統
  - Ex: Ubuntu 16.04 (64bit version) based on Vmware or virtual PC.

實驗步驟 (2)

- **Assume:**
  - --target=arm-linux-gnueabihf

- **下載GCC 4.9.3、binutils-2.25.1、GLIBC 2.19、Linux kernel**
- Build cross binutils
- Build a bare-metal cross compiler for ARM
- **Build target header files and runtime libraries**
  - Build a GLIBC
  - Or copy from the target machine
- **Build a cross compiler for ARM Linux**

Download Source Code

- **Binutils 2.25.1**
- **GCC 4.9.3**
- **GLIBC 2.19**
- **Linux kernel (from Raspberry Pi website)**

Download Source Code: binutils

- To get a list of available branches, use the command:

  `http://ftp.gnu.org/gnu/binutils/`
  `or ftp ftp.gnu.org:/gnu/binutils/`

  Download binutils-2.25.1.tar.gz

Download Source Code: GCC

- To get a list of available branches, use the command:

```
$ svn ls svn://gcc.gnu.org/svn/gcc/branches
```

- To get a list of available branches, use the command:

```
$ svn ls svn://gcc.gnu.org/svn/gcc/tags
```

Download Source Code: GCC

- Download gcc 4.9.3

```
$ svn co \
svn://gcc.gnu.org/svn/gcc/tags/gcc_4_9
_3_release gcc
```

- Or ftp ftp.gnu.org:/gnu/gcc/gcc-4.9.3/

Download Source Code: Linux kernel

```
$ git clone \
https://github.com/raspberrypi/l
inux --depth 1
```

Download Source Code: EGLIBC

- **GLIBC**
  - `ftp ftp.gnu.org:/gnu/glibc/glibc-2.19.tar.gz`

Build target runtime library and header files from source codes

1. Build Cross Binutils

- $ mkdir myWORK　　(myWORK目錄的位置自己設定)
- $ cd myWORK
- $ tar -zxvf binutils-2.25.1.tar.gz
- $ mkdir build_binutils
- $ cd build_binutils
- $ ../binutils-2.25.1/configure \
  --prefix=**/home/pschen/**WORK/crossgcc1\
  --target=arm-linux-gnueabihf
- $ make
- $ make install

/home/pschen/請改為你自己的**home directory**
後面的投影片也一樣

2. Build a Bare-metal Cross Compiler (1)

- **安裝GMP、MPFR、MPC (Building GCC requires GMP 4.2+, MPFR 2.4.0+ and MPC 0.8.0+)**
  - 使用apt-get install libgmp-dev libmpfr-dev libmpc-dev

- $ cd myWORK
- $ tar -zxvf gcc-4.9.3.tar.gz
- $ mkdir build_gcc1
- $ cd build_gcc1

2. Build a Bare-metal Cross Compiler (2)

- Add "/home/pschen/WORK/crossgcc1/bin" to PATH
- $ ../gcc-4.9.3/configure \
  --prefix=/home/pschen/WORK/crossgcc1 \
  --target=arm-linux-gnueabihf \
  --enable-languages=c **--without-headers** \
  --disable-libmudflap --disable-libatomic \
  --with-arch=armv6 --disable-shared \
  --enable-static --disable-decimal-float \
  --disable-libgomp --disable-libitm \
  --disable-libquadmath --disable-libsanitizer \
  --disable-libssp **--disable-threads** \
  --with-float=hard --with-fpu=vfp
- $ make
- $ make install

3-1. Installing Kernel Headers

Change to the directory of linux kernel:

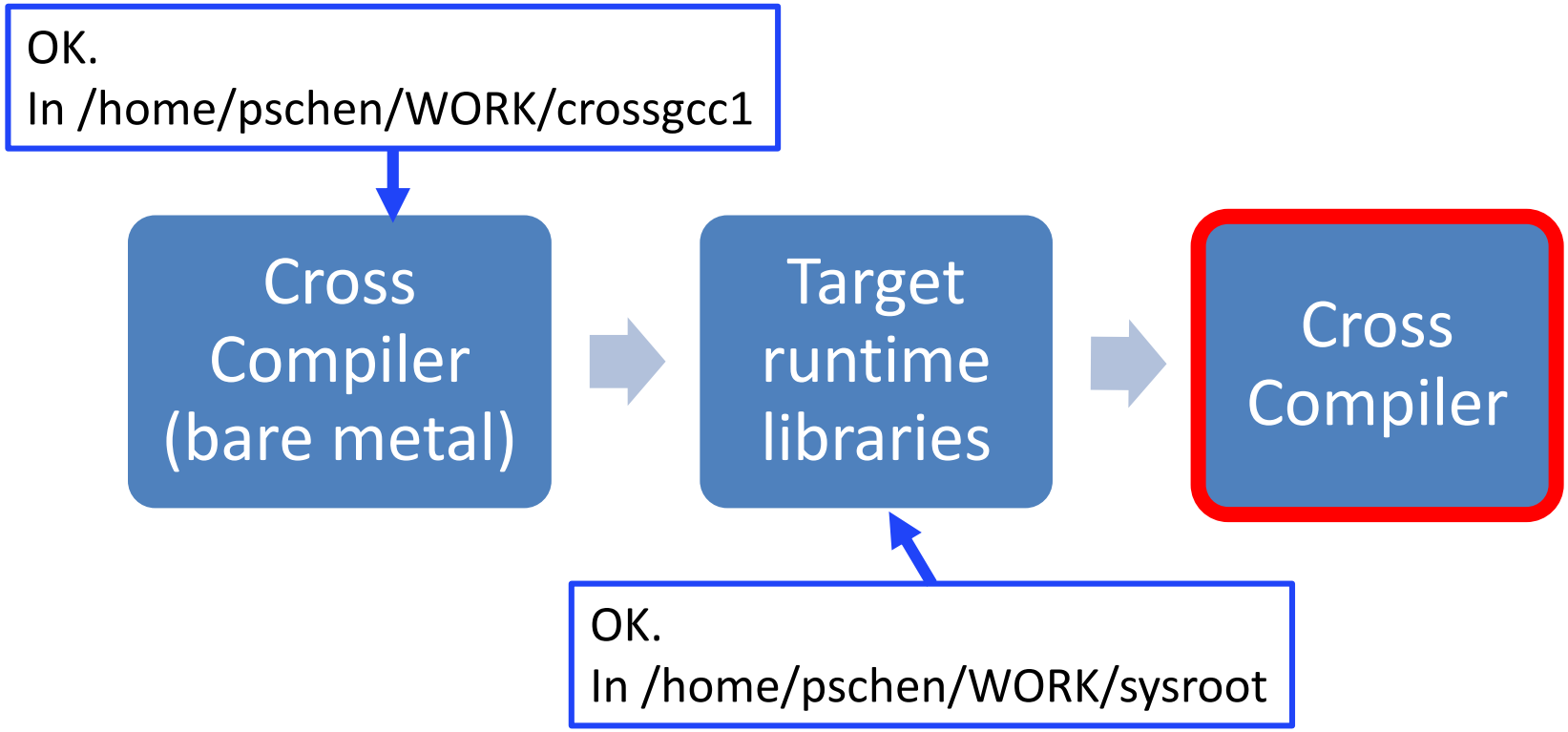
- $ cd linux
- $ make headers_install **ARCH=arm** \
  INSTALL_HDR_PATH=**/home/pschen/WORK/sysroot/usr/**

3-2. Building GLIBC

- Add /home/pschen/WORK/crossgcc1/bin to the PATH.

- `$ cd myWORK`
- `$ mkdir build_eglibc`
- `$ cd build_eglibc`
- `$ ../glibc-2.19/configure --prefix=/usr \`
  `--host=arm-linux-gnueabihf \`
  `--target=arm-linux-gnueabihf \`
  `--with-headers=`**`/home/pschen/WORK/sysroot`**`/usr/include \`
  `--includedir=/usr/include` **`--enable-add-ons`** `\`
  `-disable-multilib`
- `$ make`
- `$ make install install_root=`**`/home/pschen/WORK/sysroot`**

Begin to Build a Complete Cross Compiler



4-1. Build Cross Binutils

- Change to the directory myWORK
- $ mkdir build_binutils2
- $ cd build_binutils2
- $ ../binutils-2.25.1/configure \
  --prefix=**/home/pschen/WORK/crossgcc2** \
  **--target=arm-linux-gnueabihf **
  --with-sysroot=/home/pschen/WORK/sysroot
- $ make
- $ make install

4-2. Build a Cross Compiler (1)

- Add /home/pschen/WORK/**crossgcc2**/bin to the PATH.

- `Change to the directory myWORK`
- `$ mkdir build_gcc2`
- `$ cd build_gcc2`

4-2. Build a Cross Compiler (2)

- ```
  $ ../gcc-4.9.3/configure \
  --prefix=/home/pschen/WORK/crossgcc2 \
  --target=arm-linux-gnueabihf \
  --enable-languages=c \
  --with-sysroot=/home/pschen/WORK/sysroot
  --with-arch=armv6 --with-fpu=vfp --with-float=hard \
  --disable-libmudflap --enable-libgomp \
  --disable-libssp --enable-libquadmath \
  --enable-libquadmath-support \
  --disable-libsanitizer --enable-lto \
  --enable-threads=posix --enable-target-optspace \
  --with-linker-hash-style=gnu --disable-nls \
  --disable-multilib --enable-long-long
  ```

- %make
- %make install
  ```

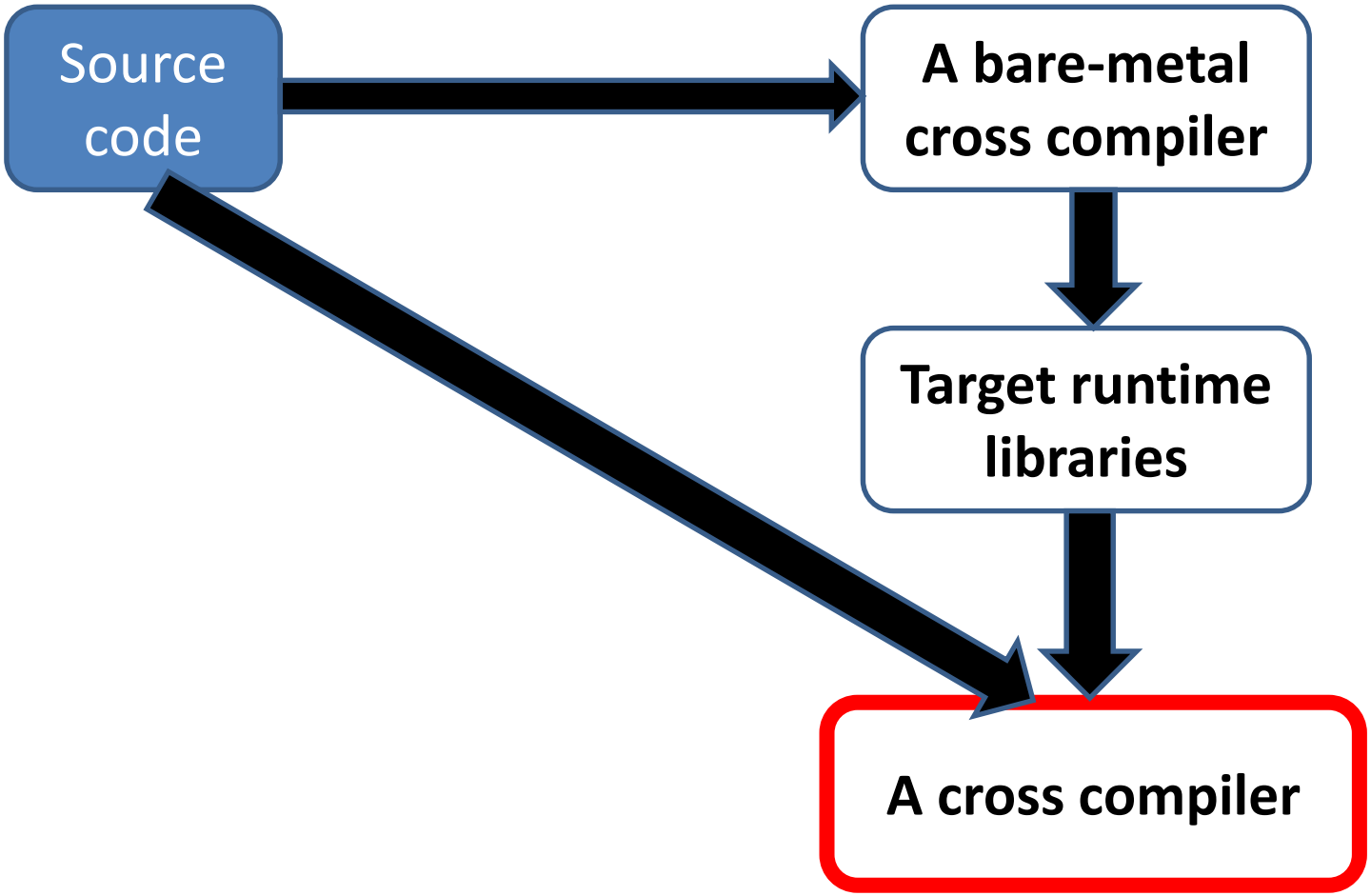
DEMO: Testing

- 驗證與測試
  - 若可編譯C program，並在ARM Linux上正確執行，則表示成功.
  - $ arm-linux-gnueabihf-gcc -S test.c

    產生相對應的ARM組合語言test.s，這個測試只表示compiler可以產生ARM組合語言，但並不表示程式可以在ARM Linux上執行.

(測試的C程式請包含C的標準函式呼叫，以便驗證glibc有安裝成功)

Conclusion


Source code
A bare-metal cross compiler
Target runtime libraries
A cross compiler