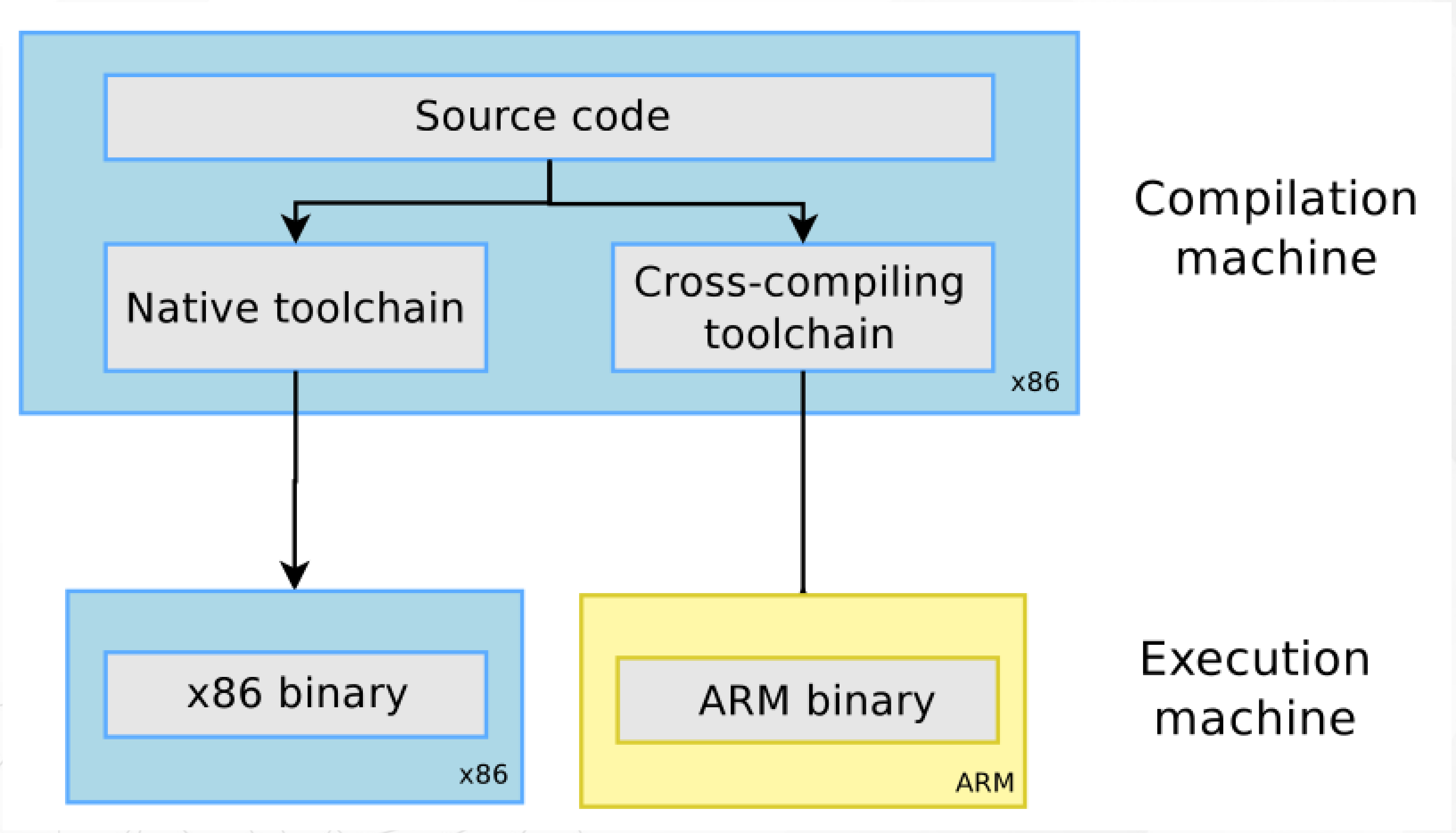
CH4 Cross Compilation Toolchain

中華行動數位科技
Chinese Action Digital Technology

Cross Compilation Tool-chain



GCC Components

- The GNU C Compiler
- The GNU Compiler Collection

Binutils

Kernel head

C/C++ libraries

GCC compiler

GDB debuger

Binutils

- Binutils
  - **as** : the assembler, that generates binary code from assembler source code
  - **ld** : the linker
  - **ar, ranlib** : to generate .a archives, used for libraries
  - **objdump, readelf, size, nm, strings** :   to inspect binaries
  - **strip** : to strip useless parts of binaries in order to reduce their size

Kernel head

- The C library and compiled programs needs to interact with the kernel
- Compiling the C library requires kernel headers, and many applications also require them
- The kernel to user space ABI is backward compatible

GCC

- GCC originally stood for the "GNU C Compiler."
- GNU Compiler Collection
  - C, C++, Ada, Objective-C, Fortran, JAVA ...
- http://gcc.gnu.org/

GCC flag

- arm-linux-gnueabihf-gcc –help
  - -c : Compile and assemble, but do not link
  - -o <file> : Place the output into <file>
  - -shared : Create a shared library
  - -g : add debug information
  - -O : sets the compiler's optimization level
  - -Wall : enables all compiler's warning messages
  - -D : defines a macro to be used by the preprocessor
  - -I : adds include directory of header files
  - -L,-l :
    - -L looks in directory for library files

C library

- The C library is an essential component of a Linux system
- Several C libraries are available:
  - **glibc, uClibc, eglibc, dietlibc, newlib**
- The choice of the C library must be made at the time of the cross-compiling toolchain generation, as the GCC compiler is compiled against a specific C library.

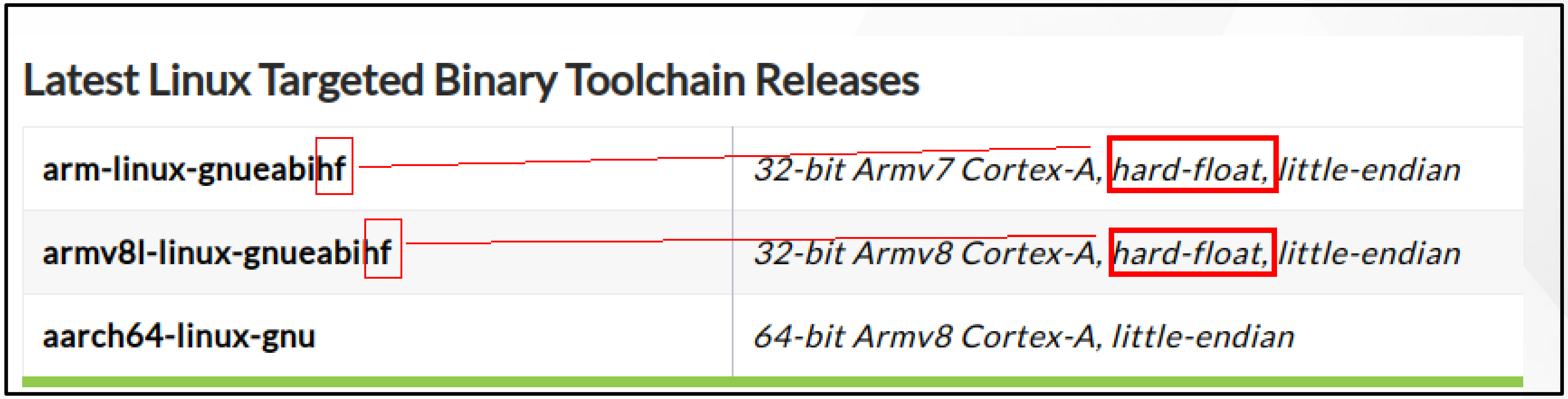
Floating point support

- For processors having a **floating point unit**, the toolchain should generate hard float code, in order to use the floating point instructions directly

- For processors without a floating point unit
  - Generate hard float code and rely on the kernel to emulate the floating point instructions
  - Generate soft float code, so that instead of generating floating point instructions, calls to a user space library are generated

Floating point support

https://www.linaro.org/downloads/



Latest Linux Targeted Binary Toolchain Releases

arm-linux-gnueabihf	*32-bit Armv7 Cortex-A, hard-float, little-endian*
armv8l-linux-gnueabihf	*32-bit Armv8 Cortex-A, hard-float, little-endian*
aarch64-linux-gnu	*64-bit Armv8 Cortex-A, little-endian*

RockPi4 Toolchain
https://releases.linaro.org/components/toolchain/binaries/7.3-2018.05/aarch64-linux-gnu/

Obtain a Toolchain

- Building a cross-compiling toolchain by ourself
  - Crosstool-NG
  - http://crosstool-ng.org/#introduction

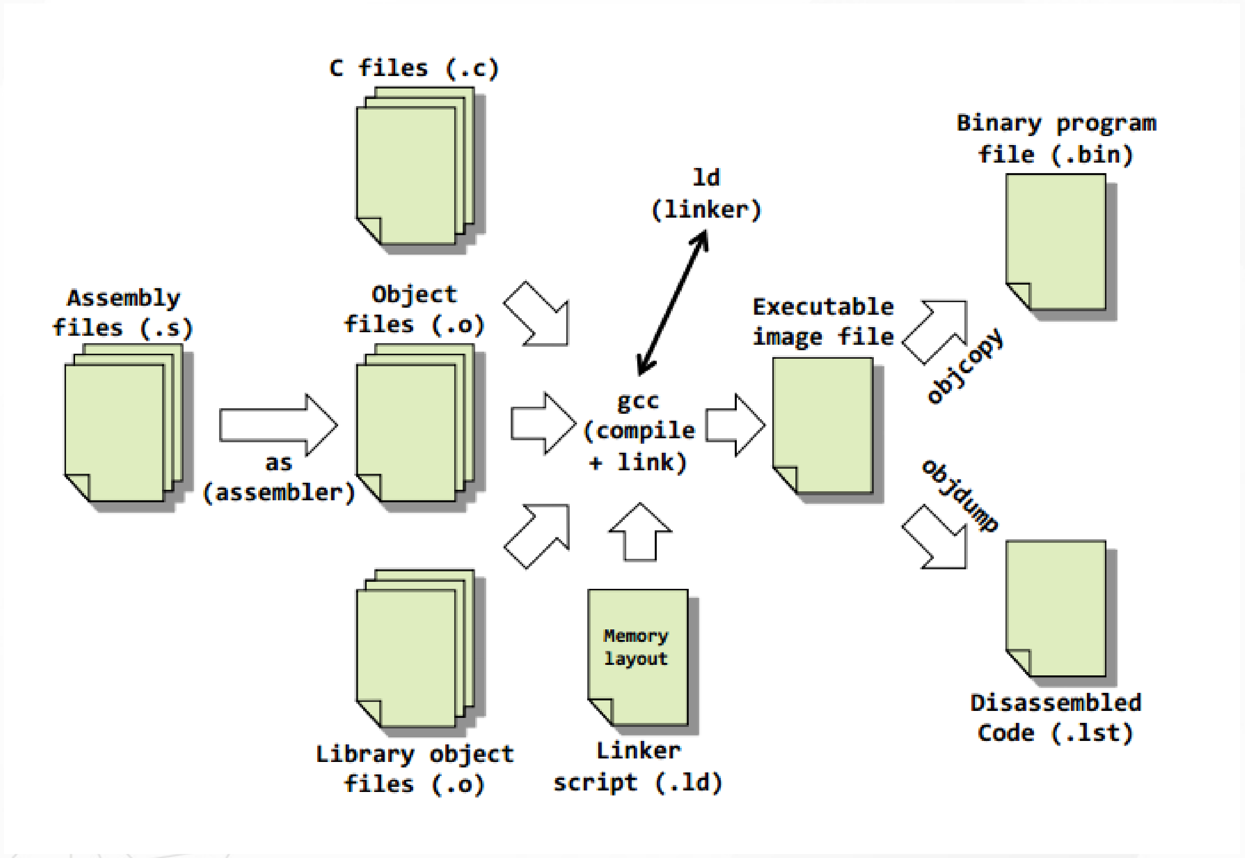
- Pre-build toolchain
  - Linaro - https://www.linaro.org/downloads/
  - By Linux distribution -
    - sudo apt-get install gcc-arm-linux-gnueabi
  - BSP
  - CodeSourcery

Installing and using Toolchain

- Add the path to toolchain binaries in your PATH: export
  - [CMD] PATH=${TOOLCHAIN_PATH}/bin/:$PATH
- Compile your applications
  - [CMD] **${PREFIX}**-gcc -o testme testme.c
- **PREFIX**
  - depends on the toolchain configuration

Compile, Assembler, Linker

Software Development Tools Overview



Tools Descriptions

- Assembler
  - Translates Assembly Language Source Files Into Machine Language Object modules
- C/C++ compiler
  - produces ARM machine code object modules
- Linker
  - Combines object files into a single executable object module

Create Linux Library

Linux Library

- Static Libraries
    - statically aware
- Dynamically Linked "Shared Object" Libraries
    - Dynamically linked at run time

Static Libraries

- static_lib_name.a
- Create static library with **ar**
  - **ar --help**
  - **ar** -cvq libctest.a test1.o test2.o
- Compile
  - gcc -o test main.c libctest.a
  - gcc -o test main.c -L/path/to/library-directory -lctest

Dynamically Linked "Shared Object" Libraries

- Dynamic_lib_name.so
- Create share library
  - gcc -share -Wl,-soname,*soname -o libname filelist liblist*
  - gcc -**shared** -Wl,-soname,libctest.so.1 -o libctest.so.1.0 test1.o test2.o
  - ln -s libctest.so.1.0 libctest.so.1
  - ln -s libctest.so.1 libctest.so
- gcc -o test main.c -L/library_PATH/ -lctest
- export LD_LIBRARY_PATH=LIB_PATH:$LD_LIBRARY_PATH
- ./test

Dynamically Linked "Shared Object" Libraries

- ldconfig
- configure dynamic linker run-time bindings
- /etc/ld.so.conf
  - 1. $ vim /etc/ld.so.conf
    - and add LIB in path /usr/local
  - 2. #ldconfig /usr/local/
    - /etc/ld.so.cache

What and Need soname ?

Real-name	libctest.so.1.0		
Soname	libctest.so.1	→	libctest.so.1.0
Linkname	libctest.so	→	libctest.so.1

Modify

Real-name	libctest.so.1.1		
Soname	libctest.so.1	→	libctest.so.1.1
Linkname	libctest.so	→	libctest.so.1

Real-name	libctest.so.1.5		
Soname	libctest.so.1	→	libctest.so.1.5
Linkname	libctest.so	→	libctest.so.1

main.c no need to re-compile

```
gcc -o test main.c -L/library_PATH/ -lctest
```