Embedded Linux System

Out Line

- CH01 Introduction to Embedded System
- CH02 Embedded Linux (1)
- CH03 Embedded Linux (2)
- CH04 Basic Software And Tool
- CH05 Cross-compile Toolchain
- CH06 Introduction to Bootloader (u-boot)
- CH07 Embedded Linux Kernel
- CH08 RootFS
- CH09 Linux Device Driver
- CH10 Control Hardware Driver

Introduction to Embedded System

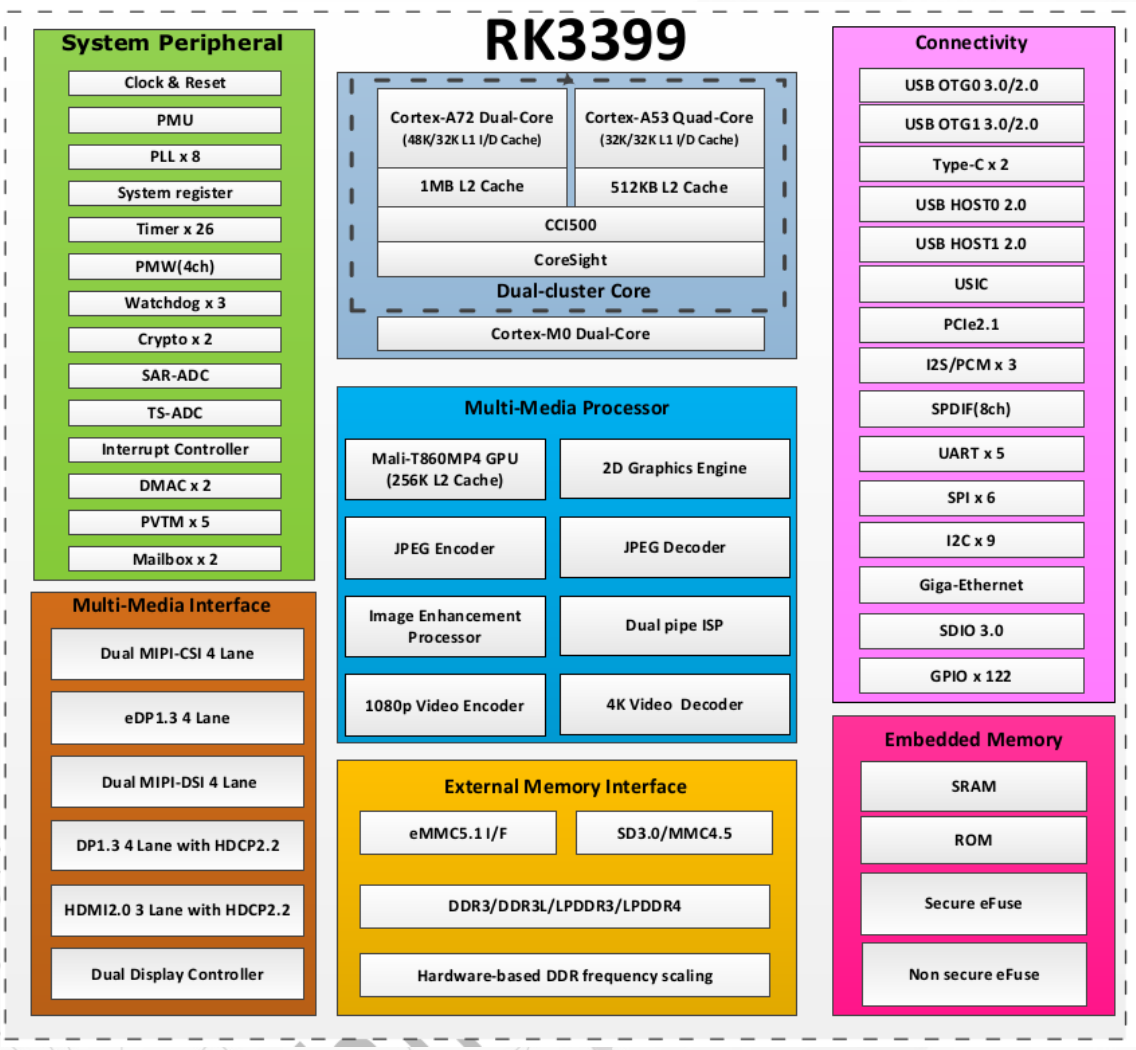
Embedded System

- An embedded system
  - combination of computer hardware and software
  - specifically designed for a particular function
- Applications
  - Mobile phone
  - Digital camera
  - Smart TV
  - Navigation system

Feature

- Designed to do some specific task
  - Low power
  - Small size
  - Special operating ranges
  - Low cost
- Install OS ?

SOC RK3399



SOC – System On Chip

- Processor

  → ARM, X86, MIPS ….

- RAM

  → 8MB ~ 4 GB

- Storagee

  → Nand, Nor flash

  → SD/MMC/eMMc

- System Bus

  → AMBA, AHB, APB, AXI ...

SOC – System On Chip

- Communication
  - I2C, I2S, USB, PCI/PCIe ...
- Media system
  - JPEG, H.264 ..
- System component
  - DMA, RTC ..

Embedded Linux ?

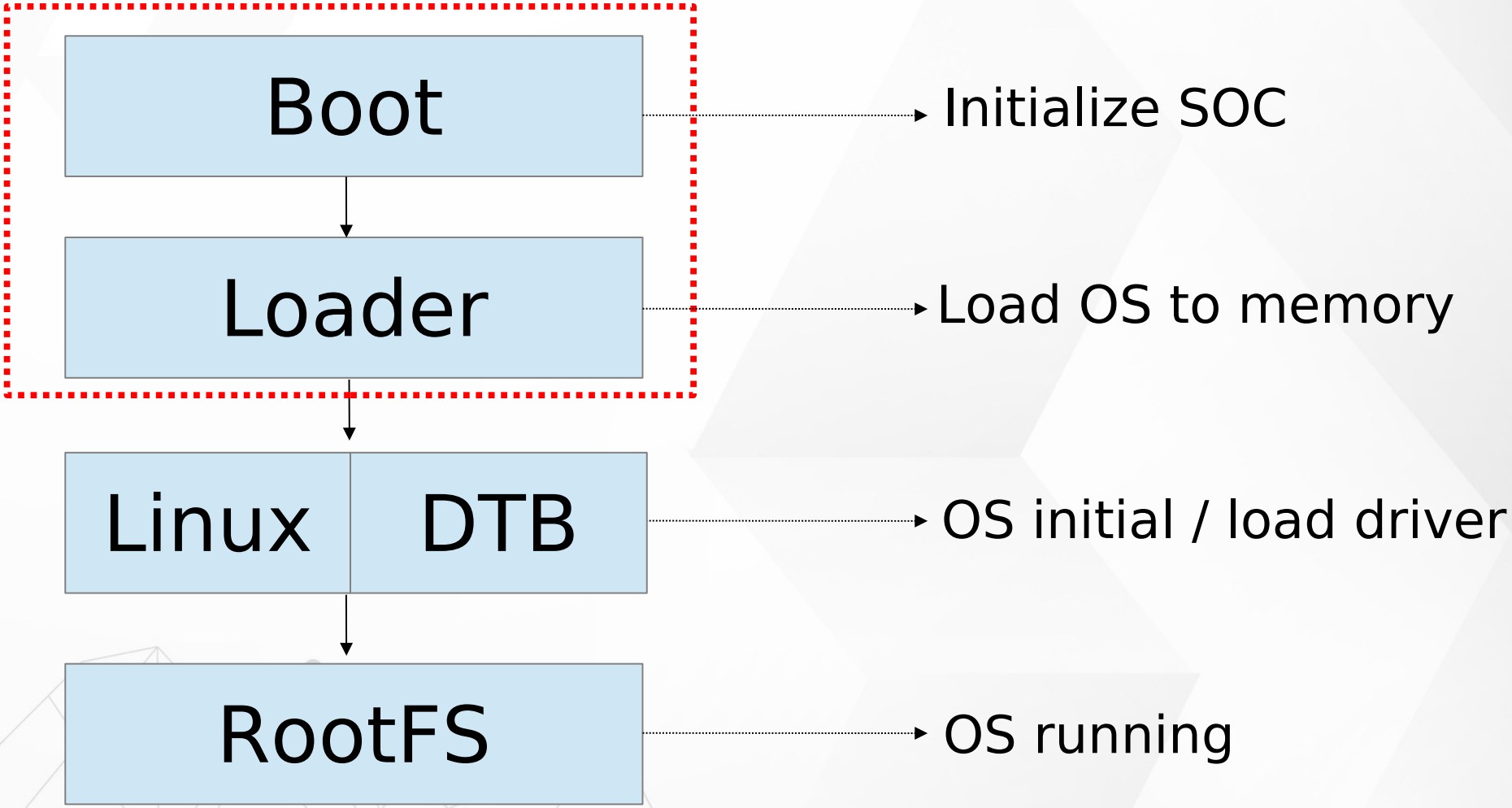
Embedded Linux is the usage of the Linux kernel and various open-source components in **embedded systems** (from Free Electrons)

Linux Advantages

- Re-use components
- Quickly design and develop complicated products
- No need to re-develop components

  → TCP/IP stack, USB stack, PCI stack ...
- Allow you modify components

Embedded Linux Booting



Software Components

- Cross-compilation Toolchain
- Boot-loader
- Linux Kernel, DeviceTree
- RootFS
- C library
- Libraries and applications
- BSP (Board Support Package)

Develop Environment

Develop Environment

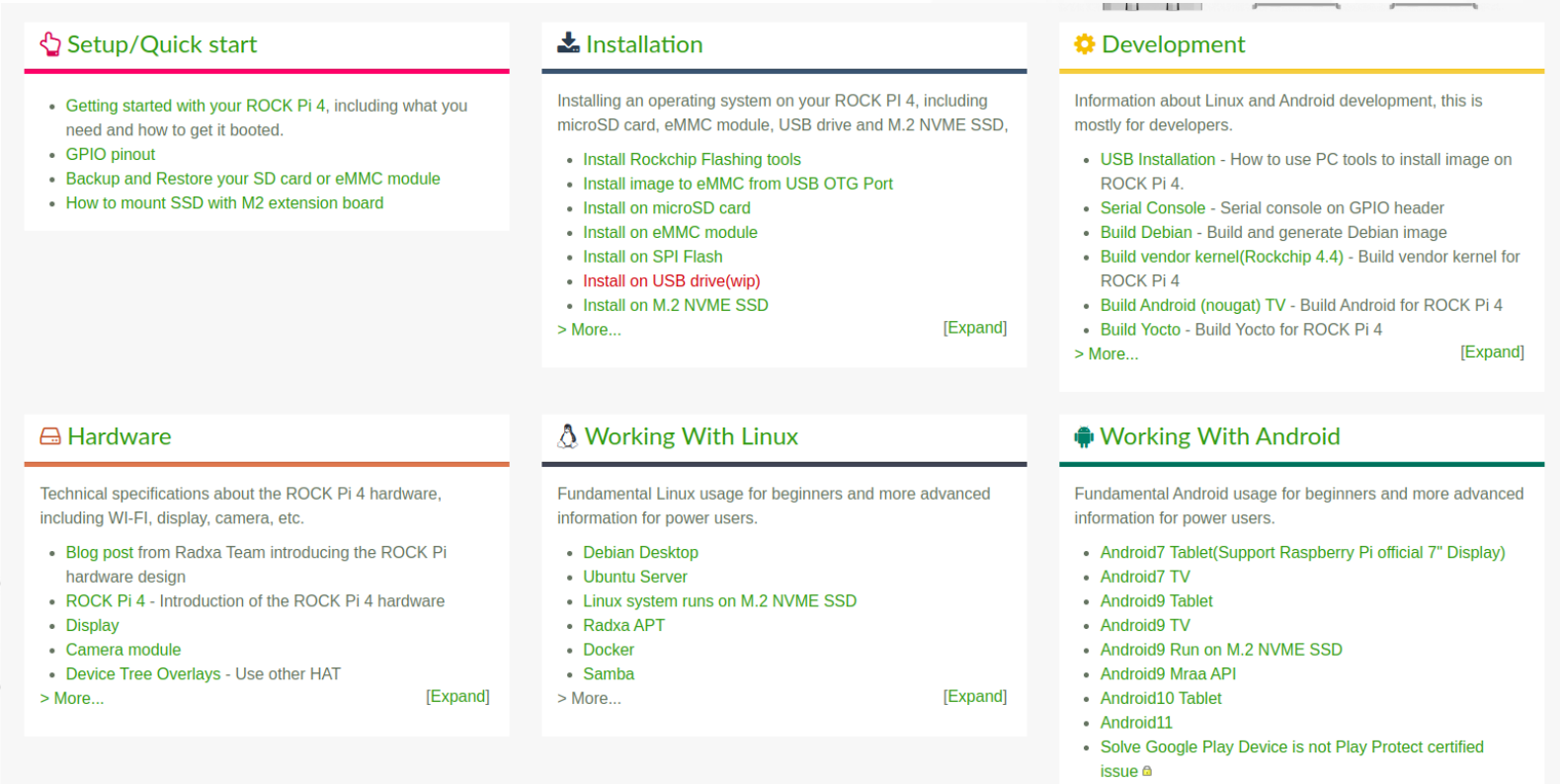
- Host PC (Linux)
- Toolchain
- BSP (Board Support Package)
- Target Board EVB (RockPi4)

BSP

- Board Support Package
- From chip vendor
  - Distribution
    - Bootloader
    - Linux Kernel
    - Device Driver
    - Rootfs

RK3399 Debian BSP (1)

- RK3399 Debian BSP
  - https://wiki.radxa.com/Rockpi4



Setup/Quick start

- Getting started with your ROCK Pi 4, including what you need and how to get it booted.
- GPIO pinout
- Backup and Restore your SD card or eMMC module
- How to mount SSD with M2 extension board

Installation

Installing an operating system on your ROCK PI 4, including microSD card, eMMC module, USB drive and M.2 NVME SSD,

- Install Rockchip Flashing tools
- Install image to eMMC from USB OTG Port
- Install on microSD card
- Install on eMMC module
- Install on SPI Flash
- Install on USB drive(wip)
- Install on M.2 NVME SSD

> More... [Expand]

Development

Information about Linux and Android development, this is mostly for developers.

- USB Installation - How to use PC tools to install image on ROCK Pi 4.
- Serial Console - Serial console on GPIO header
- Build Debian - Build and generate Debian image
- Build vendor kernel(Rockchip 4.4) - Build vendor kernel for ROCK Pi 4
- Build Android (nougat) TV - Build Android for ROCK Pi 4
- Build Yocto - Build Yocto for ROCK Pi 4

> More... [Expand]

Hardware

Technical specifications about the ROCK Pi 4 hardware, including WI-FI, display, camera, etc.

- Blog post from Radxa Team introducing the ROCK Pi hardware design
- ROCK Pi 4 - Introduction of the ROCK Pi 4 hardware
- Display
- Camera module
- Device Tree Overlays - Use other HAT

> More... [Expand]

Working With Linux

Fundamental Linux usage for beginners and more advanced information for power users.

- Debian Desktop
- Ubuntu Server
- Linux system runs on M.2 NVME SSD
- Radxa APT
- Docker
- Samba

> More... [Expand]

Working With Android

Fundamental Android usage for beginners and more advanced information for power users.

- Android7 Tablet(Support Raspberry Pi official 7" Display)
- Android7 TV
- Android9 Tablet
- Android9 TV
- Android9 Run on M.2 NVME SSD
- Android9 Mraa API
- Android10 Tablet
- Android11
- Solve Google Play Device is not Play Protect certified issue

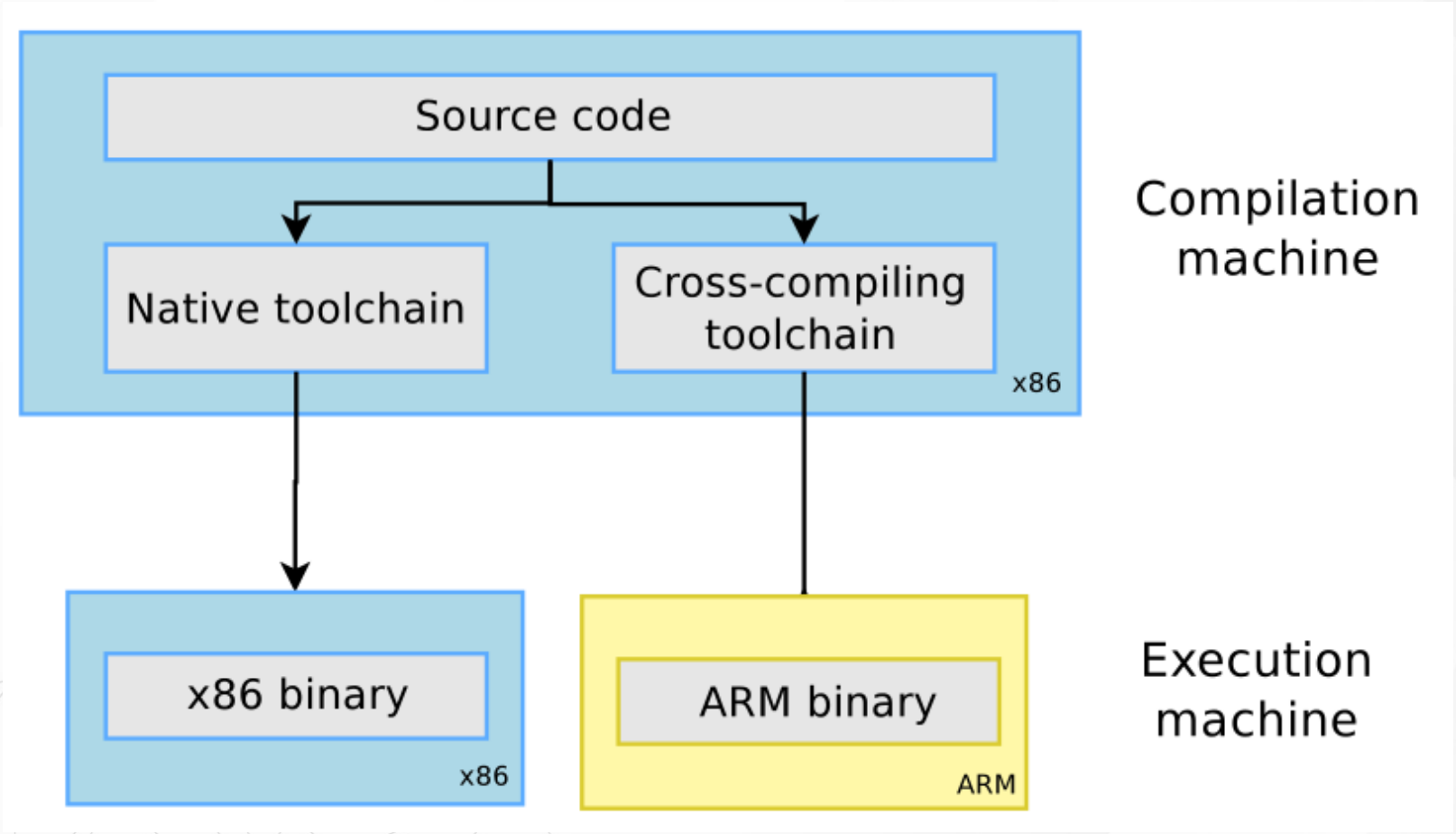
RK3399 Debian BSP (2)

- Boot-Loader
  - RKBin, U-Boot
- Kernel
  - Linux Kernel source
- Rootfs
  - Debian File System
- Tool-Chain
  - Compile tool

Cross Compilation

- Native Environment
  - Host x86PC (Linux)
  - gcc
- Cross-Compile
  - aarch64-linux-gnu-gcc

Cross Compilation



RK3399 and SOC

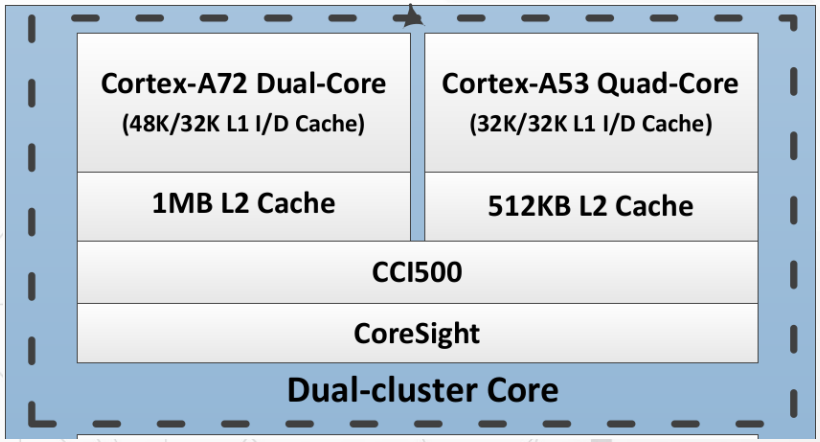
RockPi WiKi

- Rock Pi4 Wiki
  - https://wiki.radxa.com/Rockpi4
- Rock Pi 4 Feture
  - https://wiki.radxa.com/Rockpi4/getting_started
- Rock Pi 4 Debin
  - https://wiki.radxa.com/Rockpi4/Debian
  - https://wiki.radxa.com/Rock4/downloads

CPU

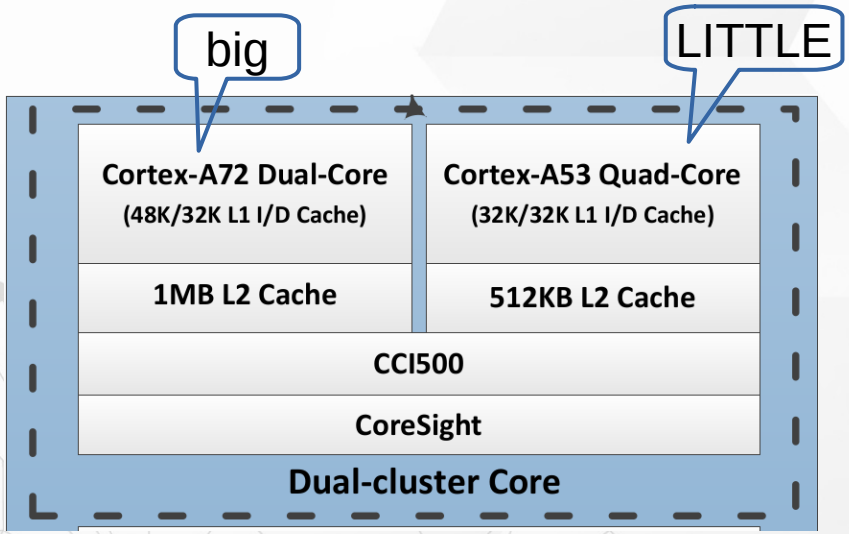
CPU (1)

- Two CPU clusters
  - Big cluster with dual-core Cortex-A72
    - high-performance
  - Little cluster with quad-core Cortex-A53
    - low power



CPU (2)

- Arm big.LITTLE technology
  - ”LITTLE” processors are designed for maximum power efficiency
  - ”big” processors are designed to provide maximum compute performance.



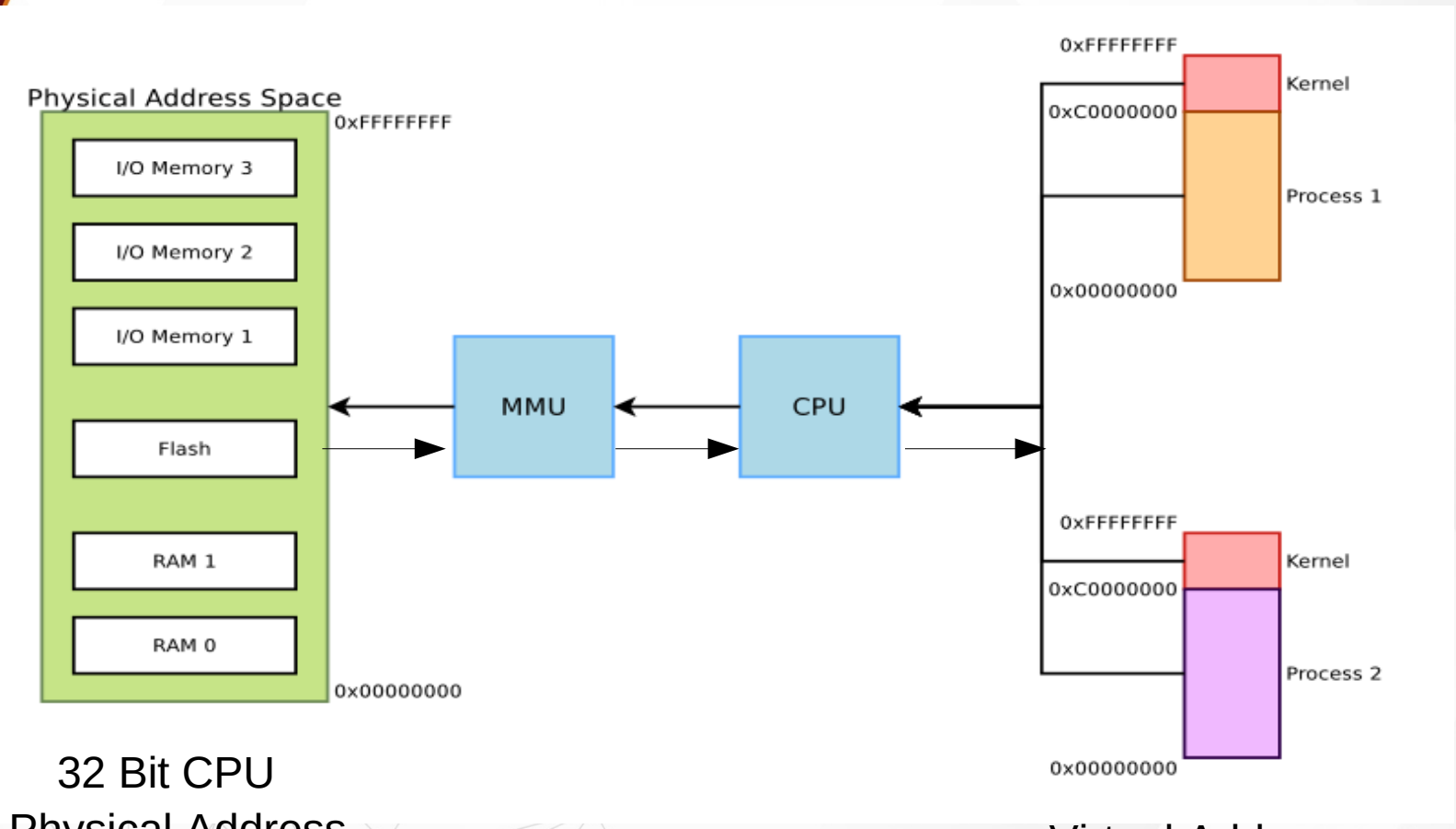
https://www.arm.com/zh-TW/technologies/big-little

Memory

Memory

- Internal ROM
  - Internal BootRom (Size : 32KB)
  - boot from
    - SPI, eMMC, SD/MMC
- Internal RAM
  - 200KB
- External
  - DDR3/DDR3L/LPDDR3/LPDDR4
  - SPI NOR/NAND Flash
  - EMMC5.1
  - SD3.0/MMC4.51

MMU (1)

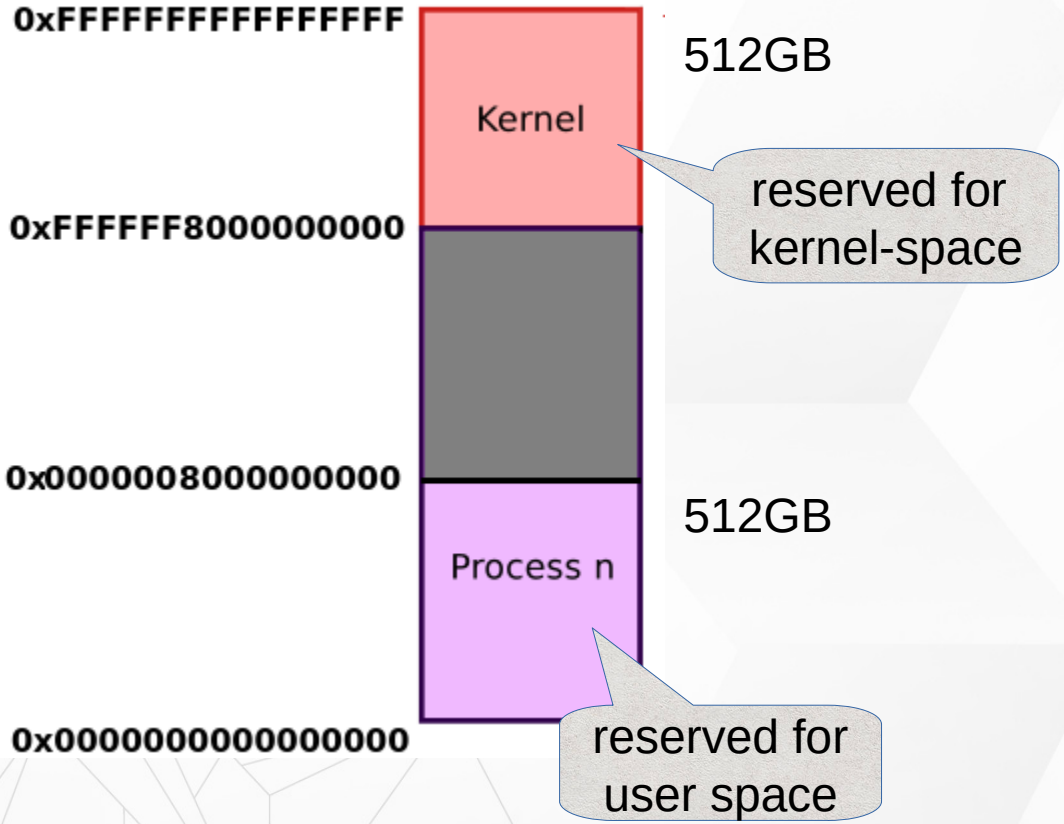


32 Bit CPU
Physical Address

Virtual Address

MMU (2)

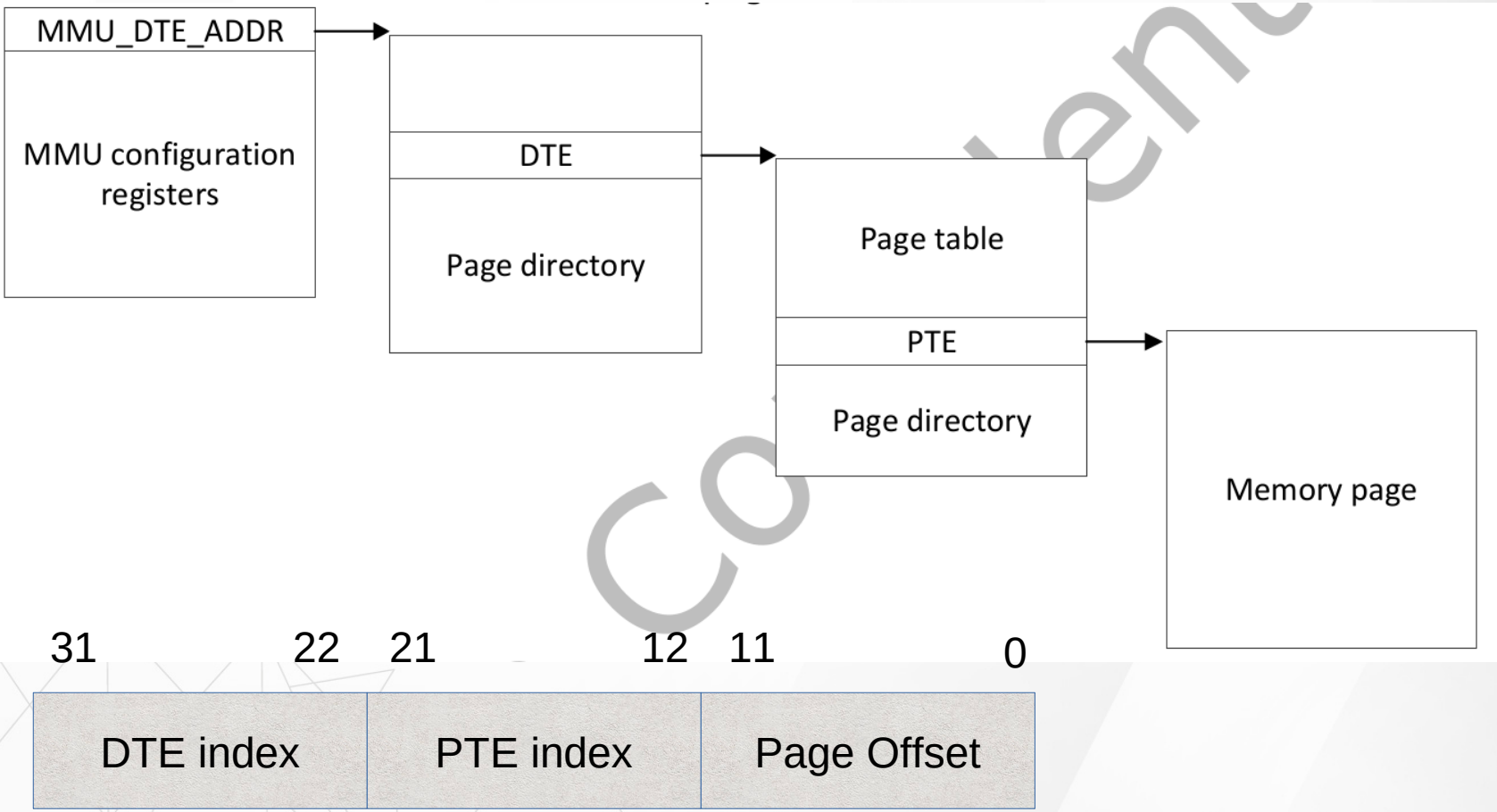
64 Bit CPU　AArch64 Linux memory layout with 4KB pages + 3 levels



MMU (3)

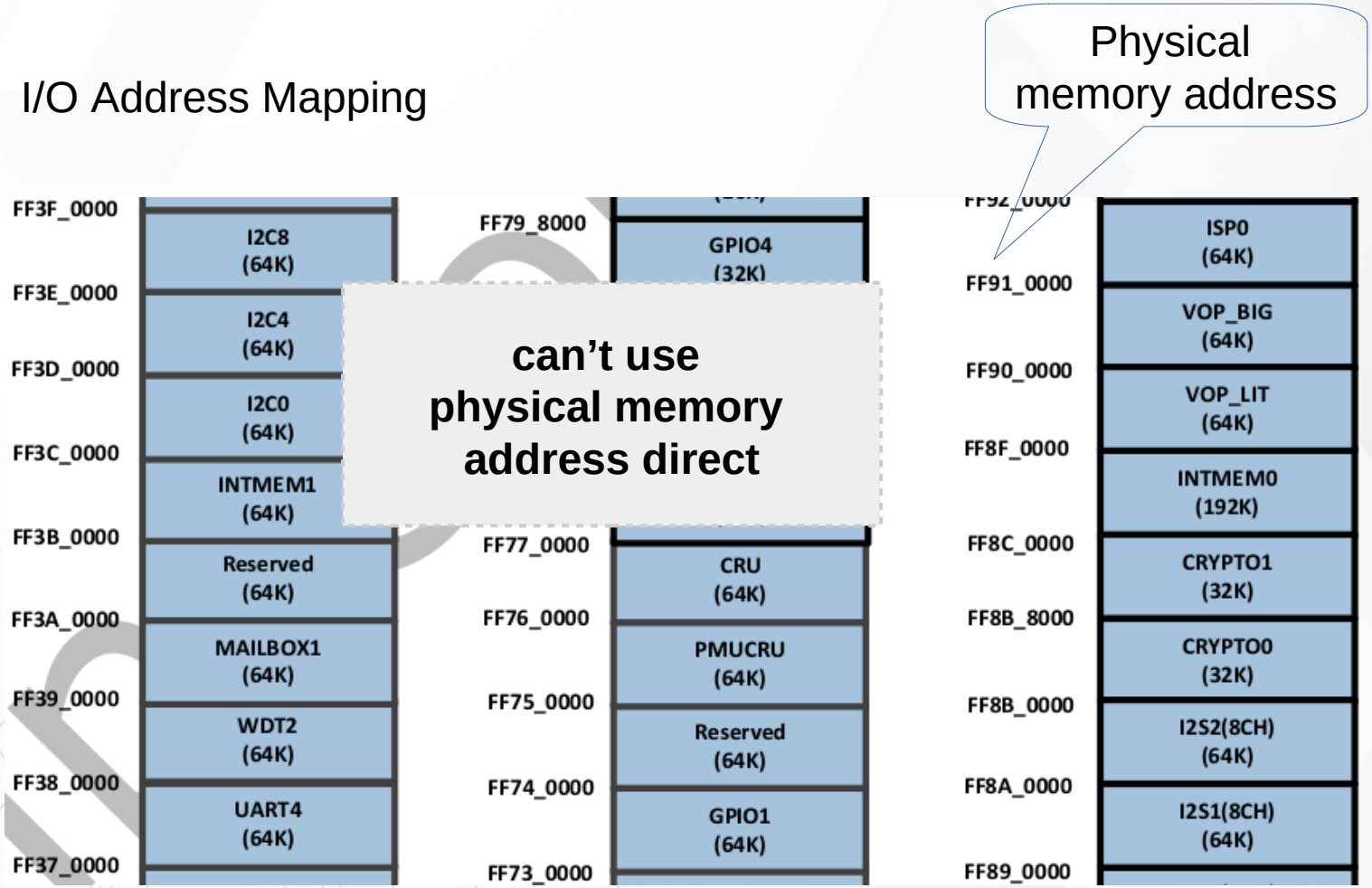
- The MMU divides memory into **4KB** pages
- 2-level page table structure
  - The First level
    - Page Directory consists of **1024** Directory Table Entries (DTEs)
    - Each pointing to a Page Table.
  - The Second level
    - The Page Table consists of **1024** Page Table Entries (PTEs)
    - Each pointing to a page in memory

MMU (4)



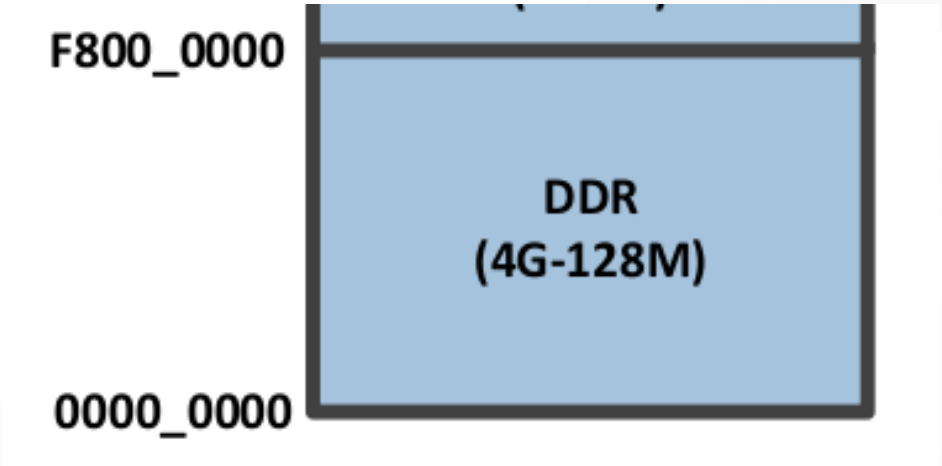
MMU (5)

I/O Address Mapping



MMU (6)

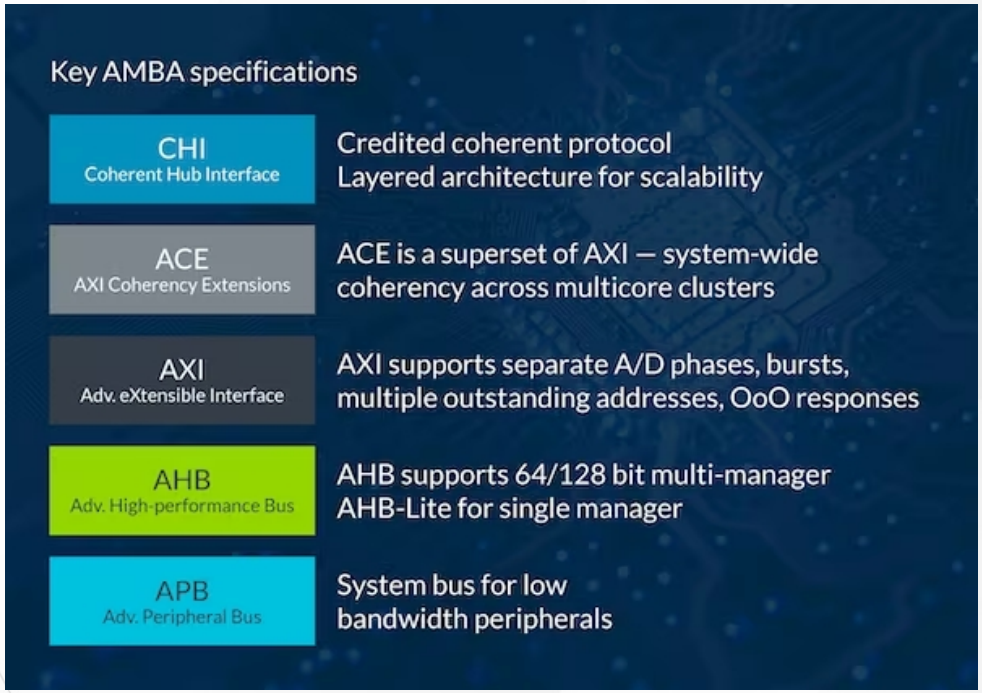
DDR SD RAM Memory



Interconnect Connect

Bus Architecture

- AMBA : Advanced Microcontroller Bus Architecture
  - AXI
  - AHB
  - APB



https://www.arm.com/zh-TW/architecture/system-architectures/amba

GPU

Graphics Engine (1)

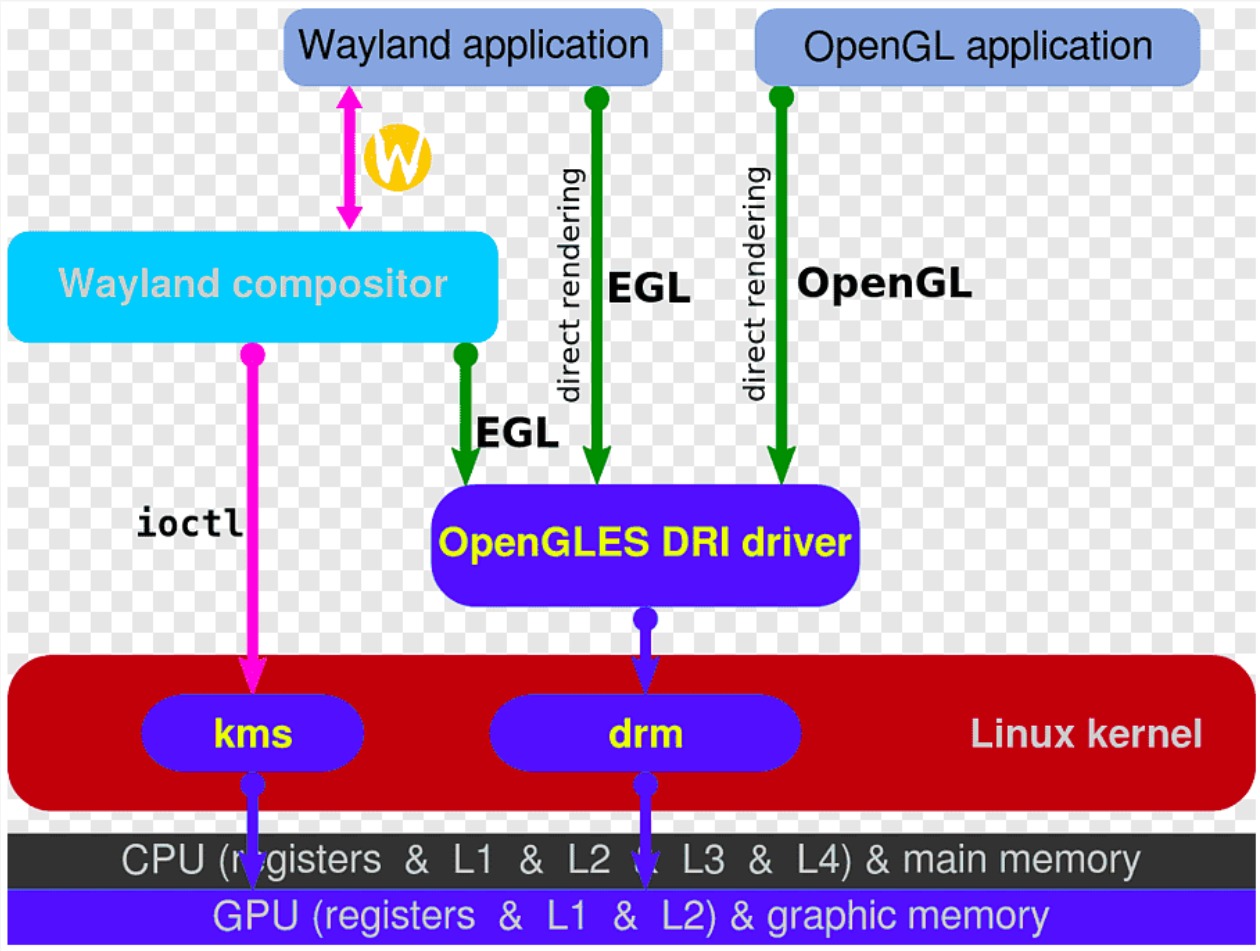
- Graphics Process Unit
- Mali-T860MP4 GPU
  - OpenGL ES1.1/2.0/3.0, OpenCL1.2,
  - 3DGraphics Engine
  - 2D Graphics Engine

Graphics Engine (2)

- OpenGL
  - Open Graphics Library
- OpenGL ES
  - OpenGL for Embedded Systems
- EGL
  - Native Platform Graphics Interface

https://www.khronos.org/egl

Graphics Engine (3)



https://en.wikipedia.org/wiki/EGL_%28API%29#/media/File:Wayland_display_server_protocol.svg

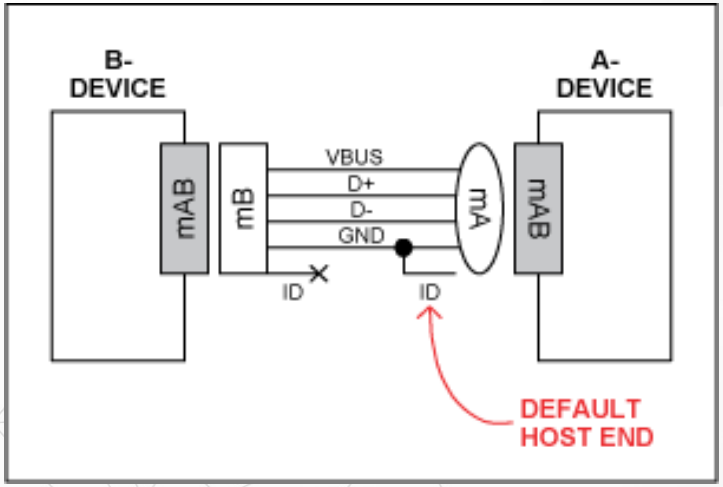
Connect

USB (1)

- USB Host
  - RK3399
  - OHCI : 1.1. Hardware Complex
  - UHCI : 1.0, 1.1 Software Complex
  - EHCI : 2.0
  - XHCI : 3.0
- USB Device
  - USB Storage

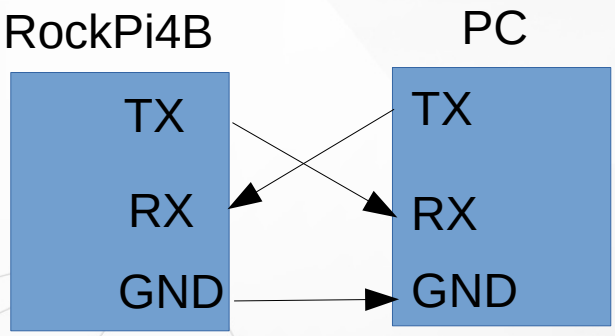
USB (2)

- USB OTG
  - USB_ID 信號爲低時，該設備應作爲 Host
  - USB_ID 信號爲高時，該設備作爲 Slave



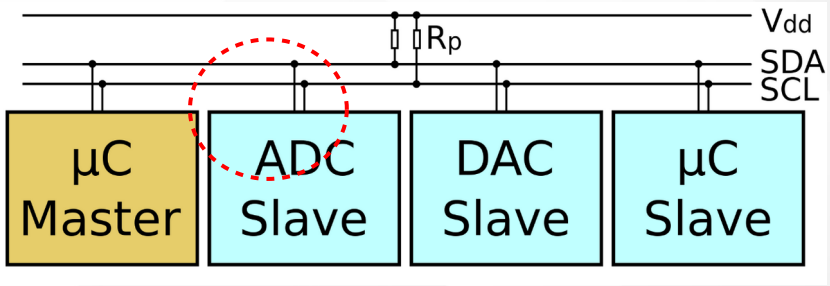
UART

- The **U**niversal **A**synchronous **R**eceiver/**T**ransmitter
  - Write Data
    - CPU → Data → APB → UART
  - Read Data
    - Data → UART → APB → CPU

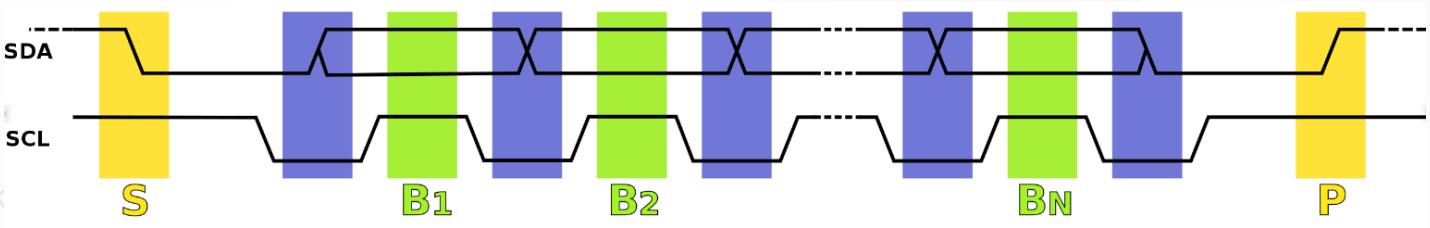


I2C (1)

- Serial bus
  - SDA data line
  - SCL clock line



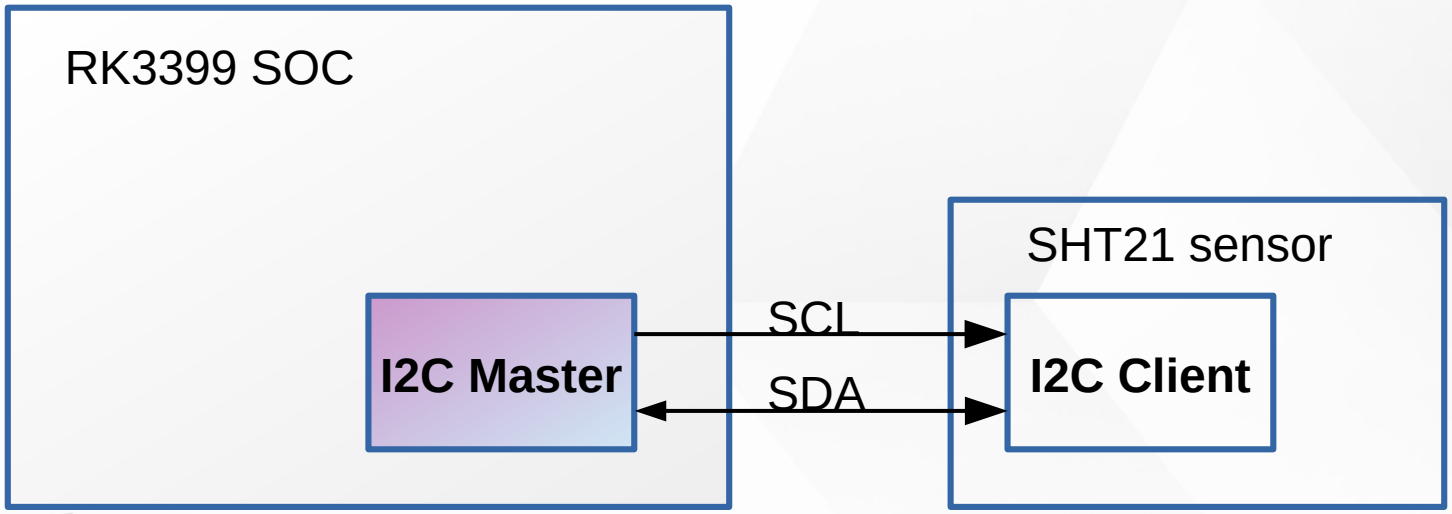
Protocol



https://en.wikipedia.org/wiki/I%C2%B2C

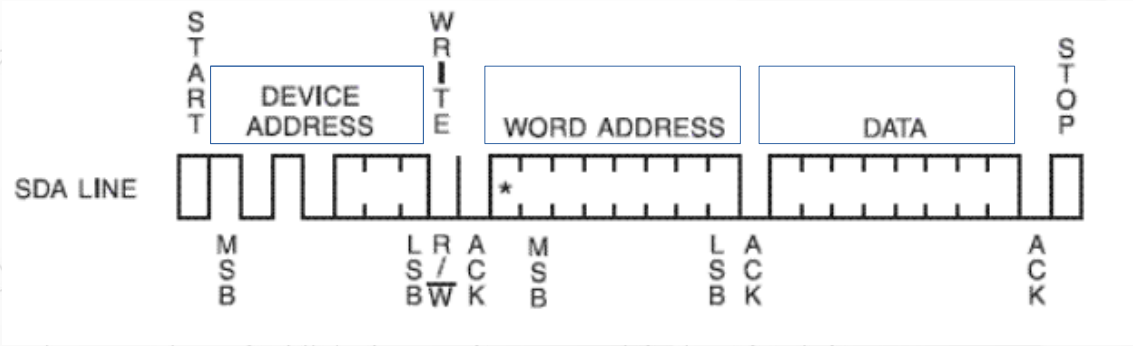
I2C (2)

Master and Client



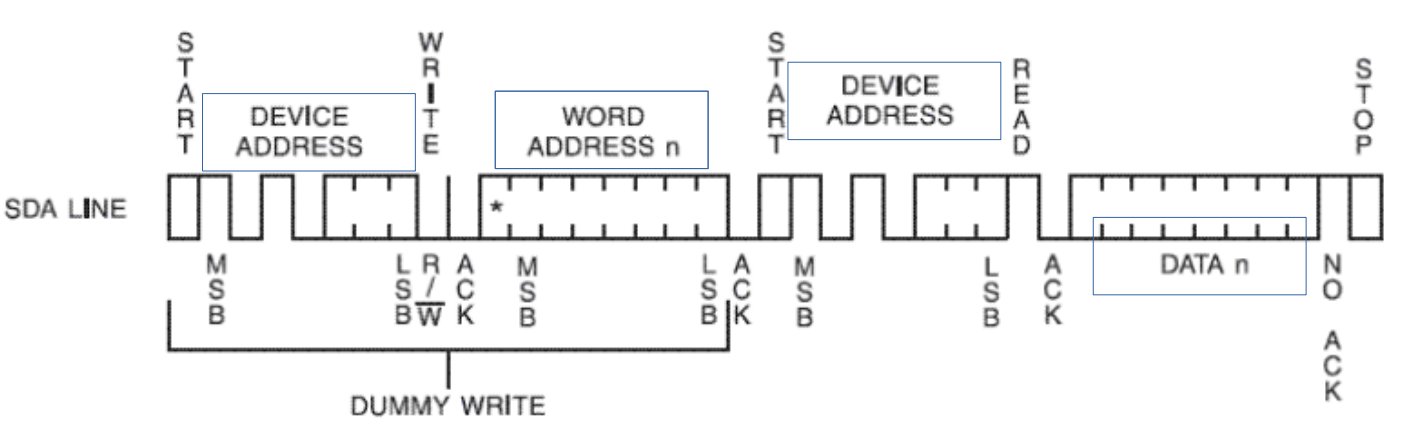
I2C protocol - Write

- Write
  - byte write
  - page write
- Device address
- Read/write bit : 0
- ACK



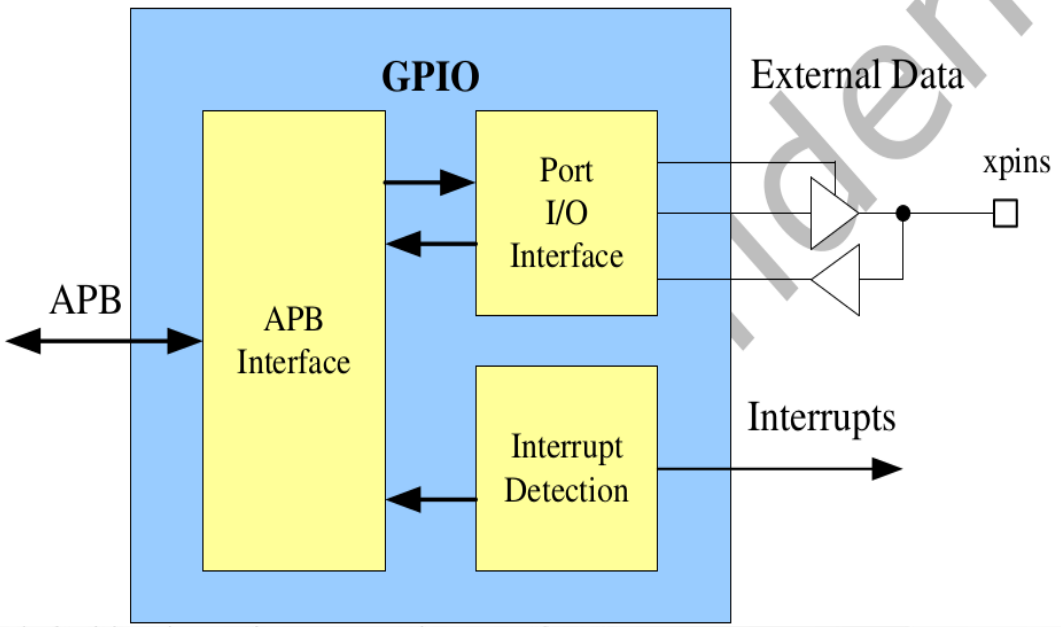
I2C protocol - Read

- Read
  - byte read
  - page read
- Device address
- **Read/write bit : 1**
- ACK



GPIO

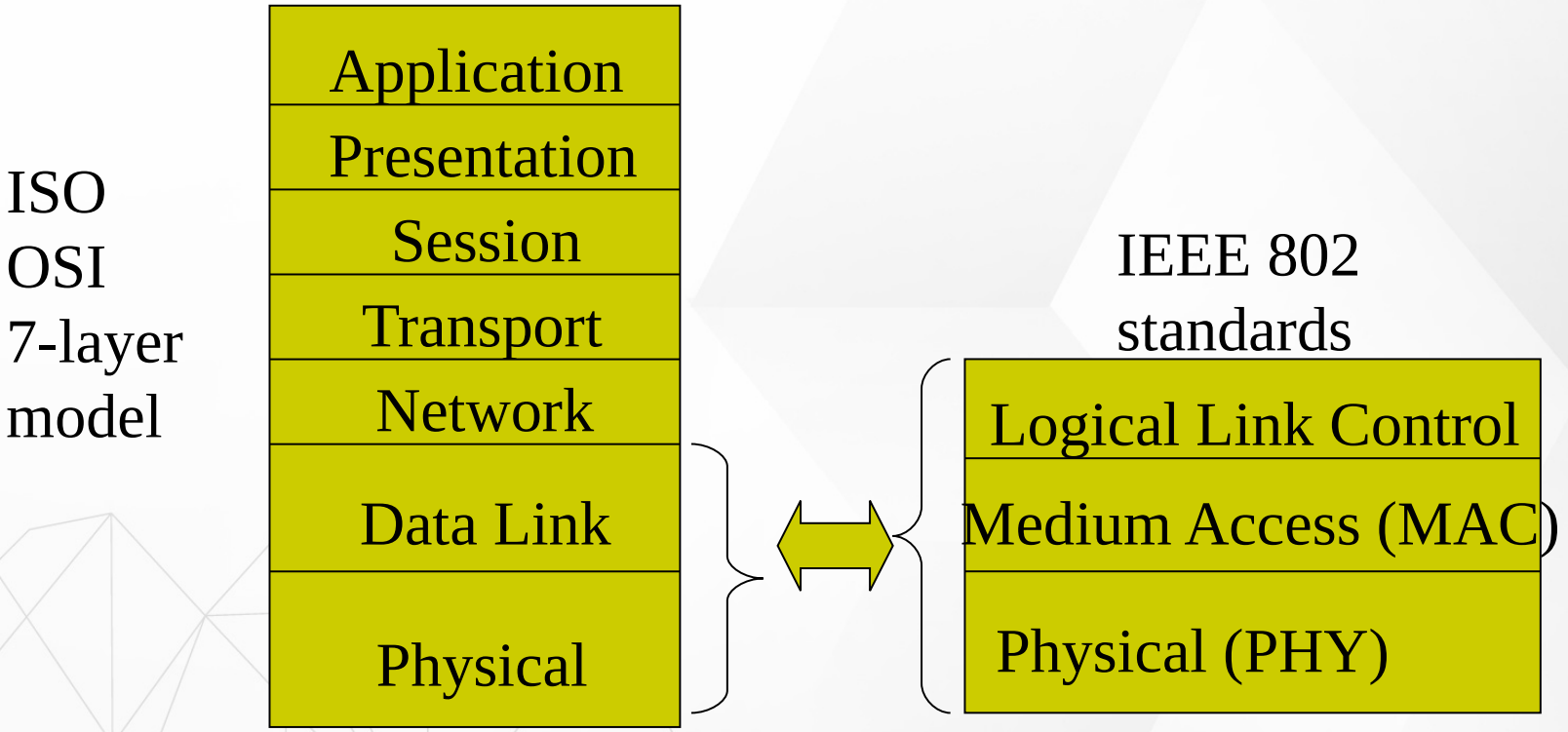
- General Purpose Programming I/O
- GPIO controls the output data and direction of external I/O pads



WiFi Basic

WIFI – 802.11 (1)

- IEEE 802.11 – 無線區域網路



https://ptolemy.berkeley.edu/projects/ofdm/ergen/docs/IEEE-802.11overview.ppt

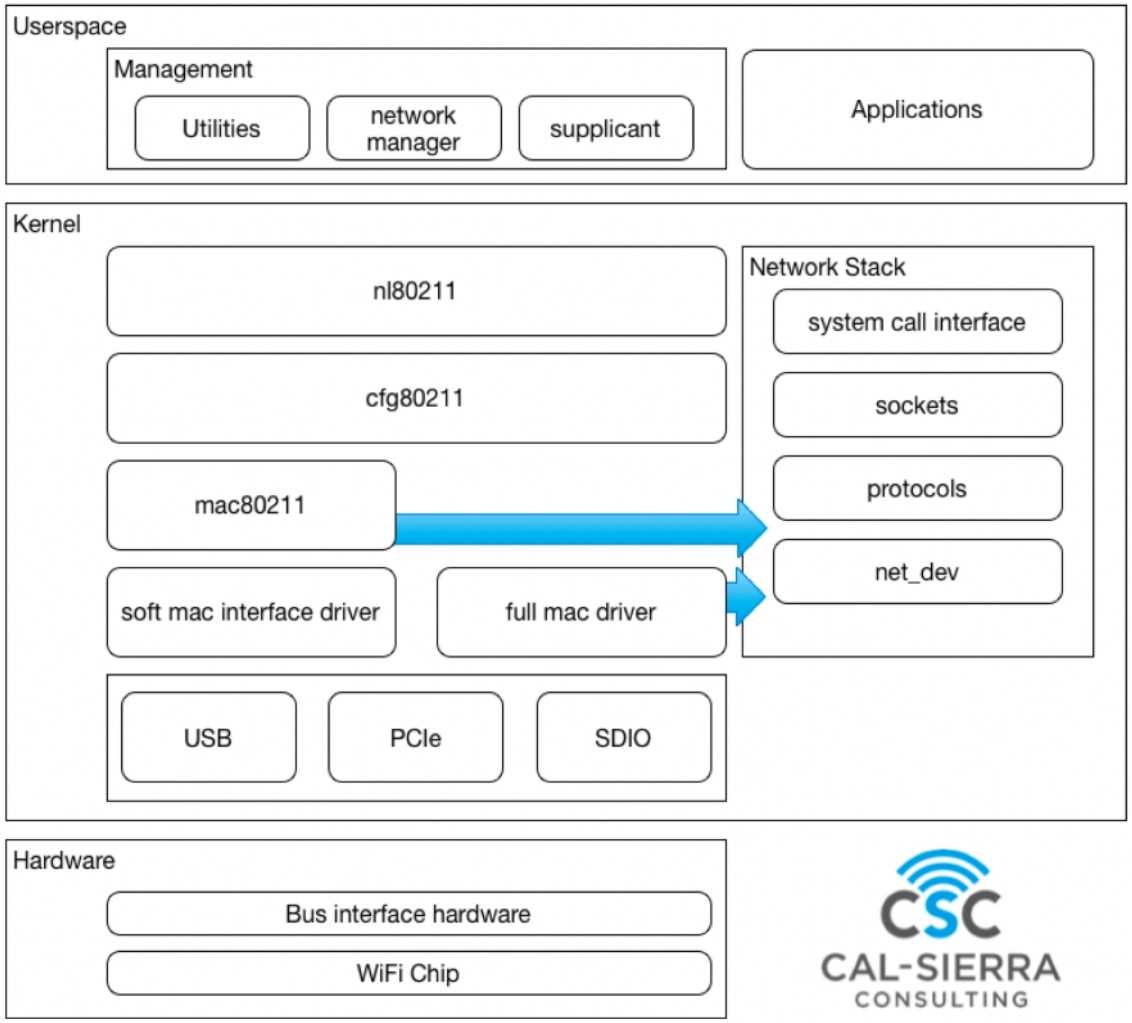
WIFI – 802.11 (2)

世代名稱[註 1]	IEEE標準		最大速率（Mbit/s）	頻率（GHz）
	名稱	發布年份		
Wi-Fi 7	802.11be	(2024) [註 2]	1376～46120	2.4/5/6
Wi-Fi 6E	802.11ax	2020	574～9608[1]	6 [2]
Wi-Fi 6			2019	2.4/5
Wi-Fi 5	802.11ac	2014	433～6933	5 [3]
Wi-Fi 4	802.11n	2008	72～600	2.4/5
Wi-Fi 3[註 3]	802.11g	2003	6～54	2.4
Wi-Fi 2[註 3]	**802.11a**	1999	6～54	5
Wi-Fi 1[註 3]	802.11b	1999	1～11	2.4
Wi-Fi 0[註 3]	802.11	1997	1～2	2.4

1. ^ Wi-Fi是Wi-Fi联盟的商標
2. ^ 預定的發布年份
3. ^ 3.0 3.1 3.2 3.3 Wi-Fi联盟未定義 Wi-Fi 0/1/2/3 的世代名稱[4][5]

https://zh.wikipedia.org/wiki/IEEE_802.11a-1999

WIFI – Linux & 802.11 (1)



CSC CAL-SIERRA CONSULTING

https://hackmd.io/@0xff07/BJkDzazVK#iwd

中華行動數位科技 Chinese Action Digital Technology

WIFI – Linux & 802.11 (2)

- mac80211
  - most associated to hardware offloading
  - the **802.11 protocol state machine lives here**
- cfg80211
  - middle-layer **Handles Everything Configurable**
- nl80211
  - The **API between user-land and kernel-land**
  - Relies on the **netlink** protocol to exchange messages between the two worlds

https://www.linuxembedded.fr/2020/05/emulating-wlan-in-linux-part-i-the-80211-stack

WIFI – Tool

- Show / manipulate wireless devices and their configuration
  - iw
- For connecting to a WPA/WPA2 network
  - wpa_supplicant

Audio

ALSA Overview

- **A**dvanced **L**inux **S**ound **A**rchitecture
  - Linux kernel
  - Software framework
- Sound Servers
  - PulseAudio, JACK ...
- ALSA stream is a data flow representing sound
  - PCM (Pulse-code modulation)

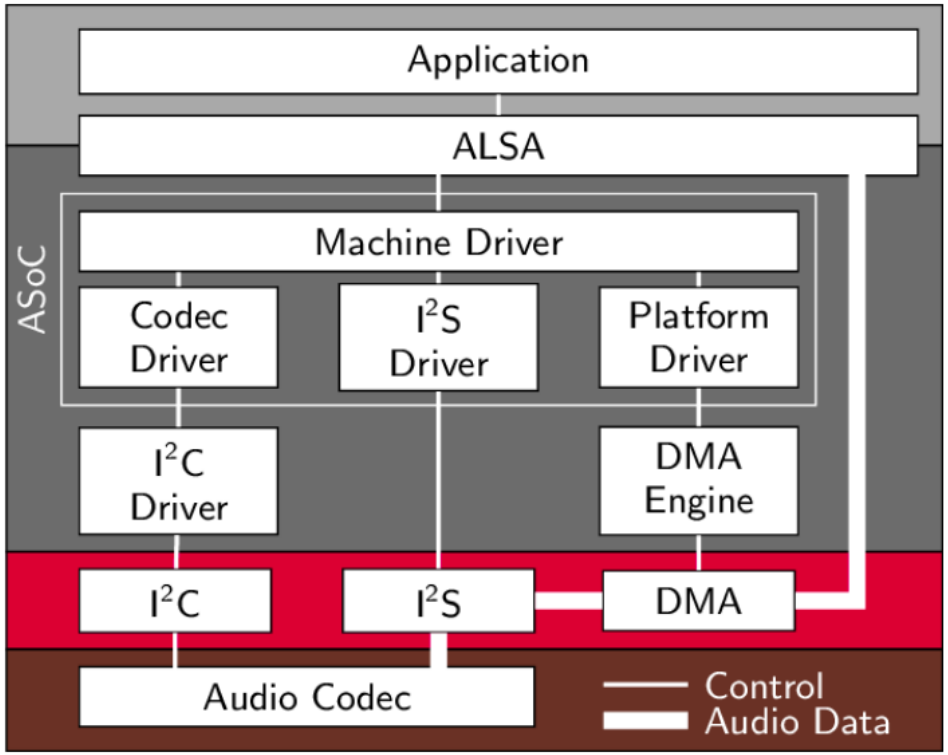
https://en.wikipedia.org/wiki/Advanced_Linux_Sound_Architecture

ALSA Overview

- Audio Codec
  - Audio Codec 就是音樂訊號（ Audio ）壓縮 / 解壓縮 (Compress/DECompress)
    的演算法或程式，前後加起來就是 Audio Codec.
- Parameters of the hardware
  - sampling rate : 44100 Hz
  - sample width : 8 bit, 16 bit, 24 bit
  - sample encoding : endianness
  - number of channels : 1 channel, 2 channel ...

https://en.wikipedia.org/wiki/Advanced_Linux_Sound_Architecture

ALSA Overview



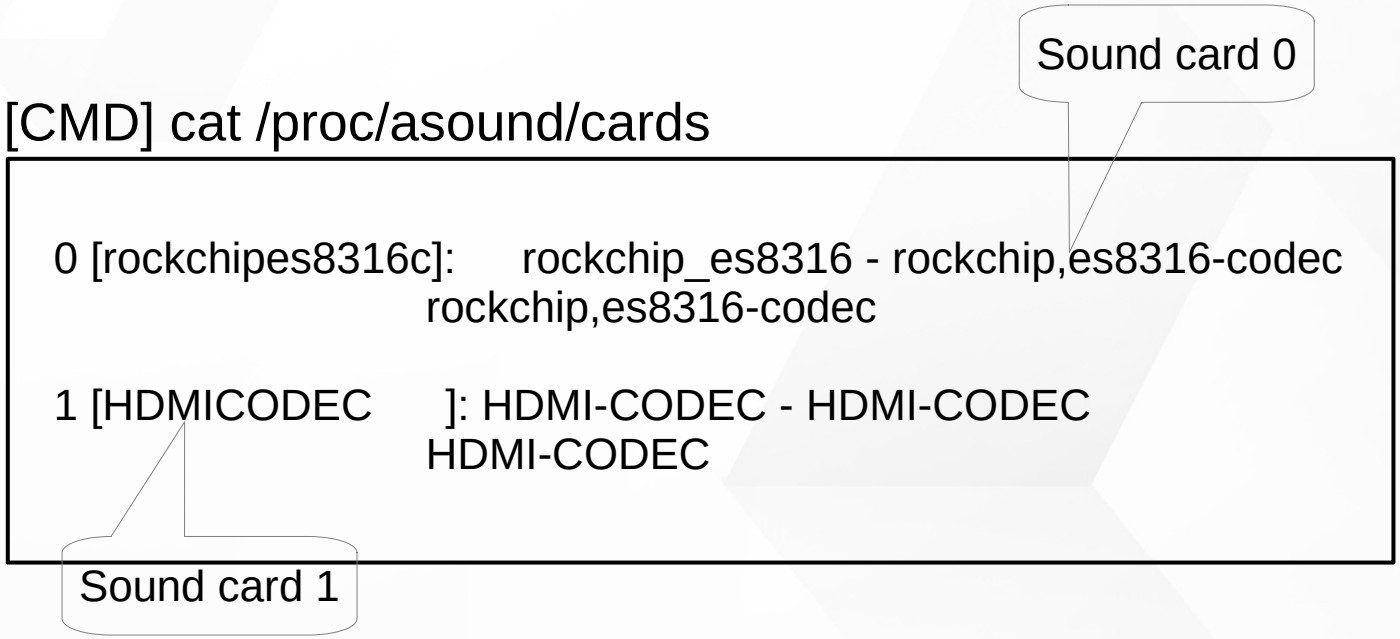
https://www.researchgate.net/figure/Structure-of-ASoC-and-the-embedment-into-the-Linux-audio-framework_fig2_262112720

Sound Card in Linux (1)

[CMD] ls /proc/asound/ -l

```
lrwxrwxrwx      1 root root 5 Apr 30 06:51 HDMICODEC -> card1
dr-xr-xr-x      4 root root 0 Apr 30 06:51 card0
dr-xr-xr-x      3 root root 0 Apr 30 06:51 card1
-r--r--r--      1 root root 0 Apr 30 06:51 cards
-r--r--r--      1 root root 0 Apr 30 06:51 devices
-r--r--r--      1 root root 0 Apr 30 06:51 hwdep
-r--r--r--      1 root root 0 Apr 30 06:51 pcm
lrwxrwxrwx      1 root root 5 Apr 30 06:51 rockchipes8316c -> card0
dr-xr-xr-x 2 root root 0 Apr 30 06:51 seq
-r--r--r--      1 root root 0 Apr 30 06:51 timers
-r--r--r--      1 root root 0 Apr 30 06:51 version
```

Sound Card in Linux (2)



Sound Card in Linux (3)

rockchipes8316c

[CMD] ls -l /proc/asound/card0/

```
-r--r--r-- 1 root root 0 Apr 30 06:59 id
dr-xr-xr-x 3 root root 0 Apr 30 06:59 pcm0c
dr-xr-xr-x 3 root root 0 Apr 30 06:59 pcm0p
```

capture

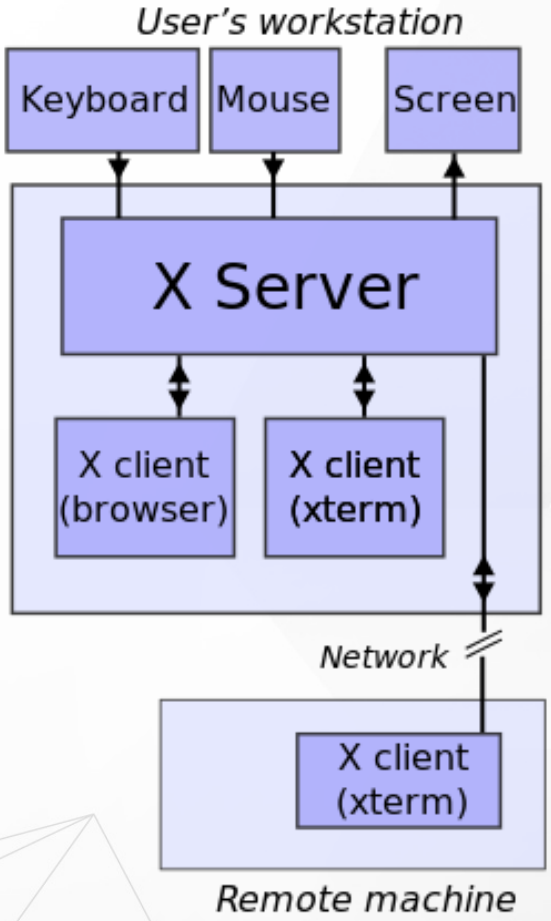
playback

[CMD] ls -l /proc/asound/card1/

```
-r--r--r-- 1 root root 0 Apr 30 07:01 id
dr-xr-xr-x 3 root root 0 Apr 30 07:01
pcm0p
```

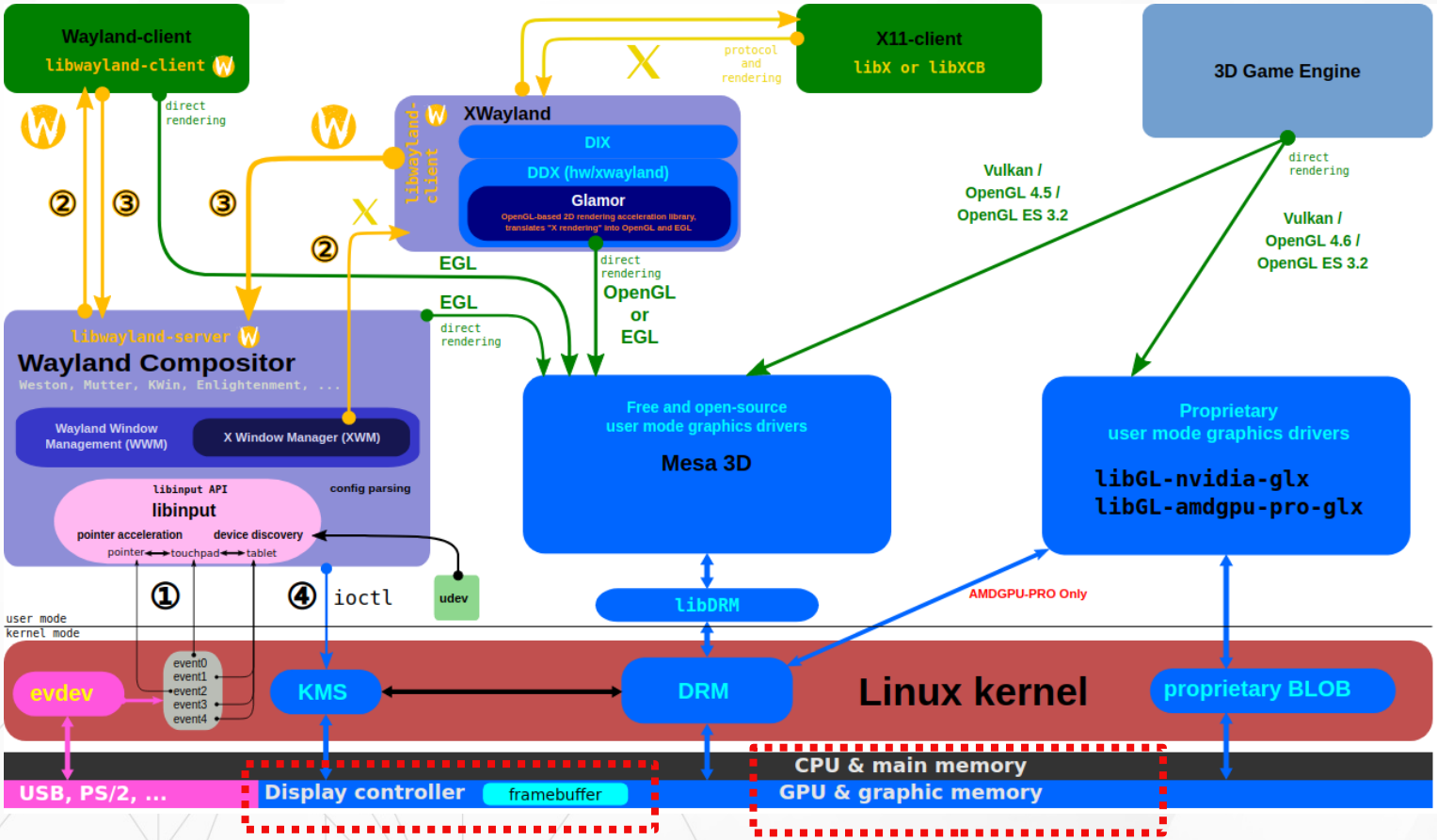
Linux Display Subsystem

Linux Windows System (1)



https://en.wikipedia.org/wiki/File:X_client_sever_example.svg

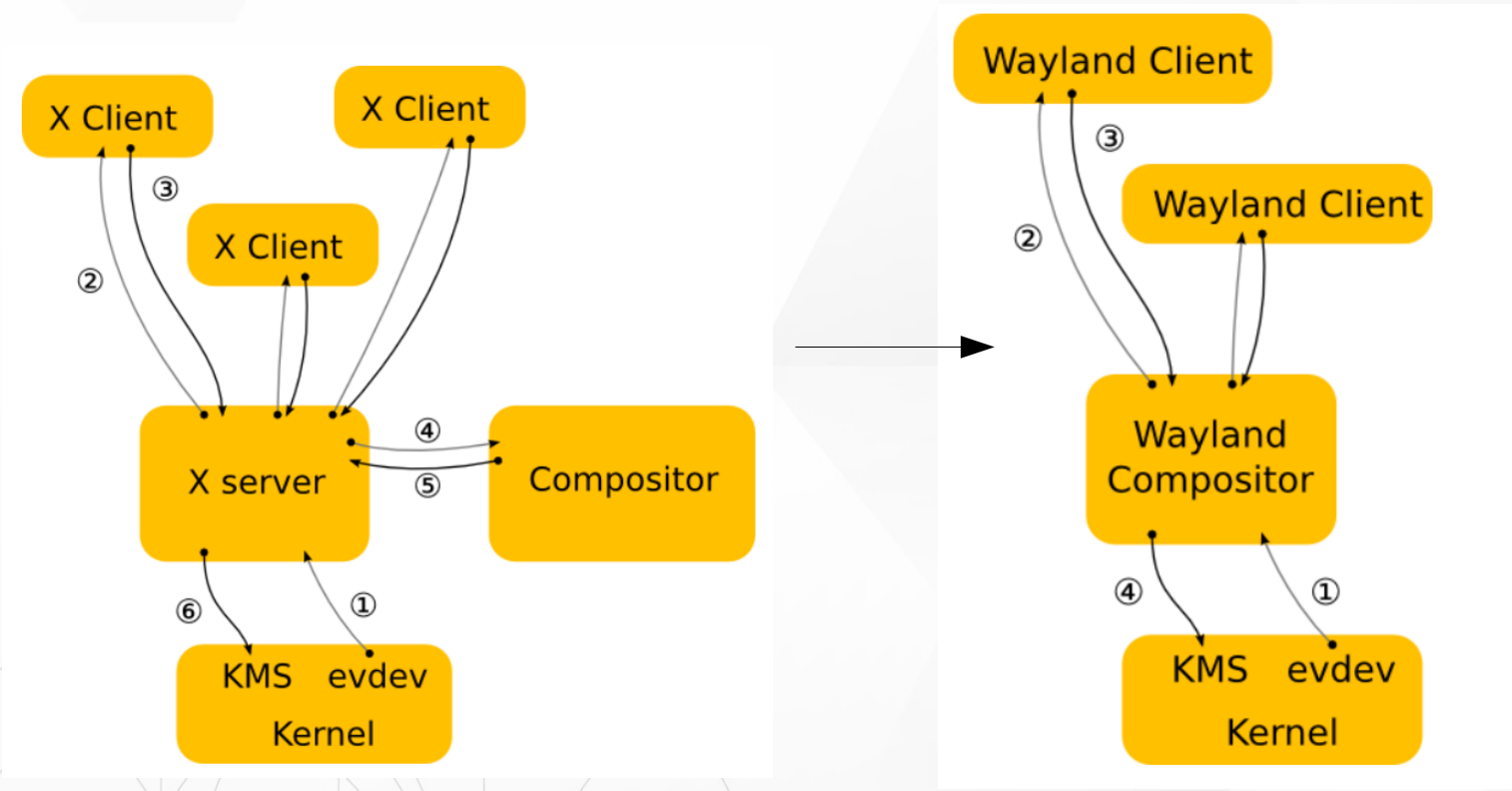
Linux Windows System (2)



https://upload.wikimedia.org/wikipedia/commons/2/2d/The_Linux_Graphics_Stack_and_glamor.svg

Linux Windows System (3)

Wayland is a replacement for the X11 window system protocol



https://wayland.freedesktop.org/architecture.html

GTK and Gnome

- GTK
  - GTK (formerly GTK+) is a free and open-source cross-platform widget toolkit for creating graphical user interfaces (GUIs).
  -
- GNOME
  - GNOME is the default desktop environment of many major Linux distributions
  - originally an acronym for GNU Network Object Model Environment
  - free and open-source desktop environment for Linux and other Unix-like[10] operating systems

https://en.wikipedia.org/wiki/GNOME

https://en.wikipedia.org/wiki/GTK

OpenGL and EGL

- EGL
  - EGL Native Platform Graphics Interface is an interface portable layer for graphics resource management.
  - works between rendering APIs such as OpenGL ES or OpenVG and the underlying native platform window system
- OpenGL
  - OpenGL (Open Graphics Library) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics
- Mesa
  - Mesa, also called Mesa3D and The Mesa 3D Graphics Library
  - it is an open source implementation of OpenGL, Vulkan, and other graphics API specifications

https://en.wikipedia.org/wiki/OpenGL

https://en.wikipedia.org/wiki/Mesa_(computer_graphics)

https://www.khronos.org/egl/

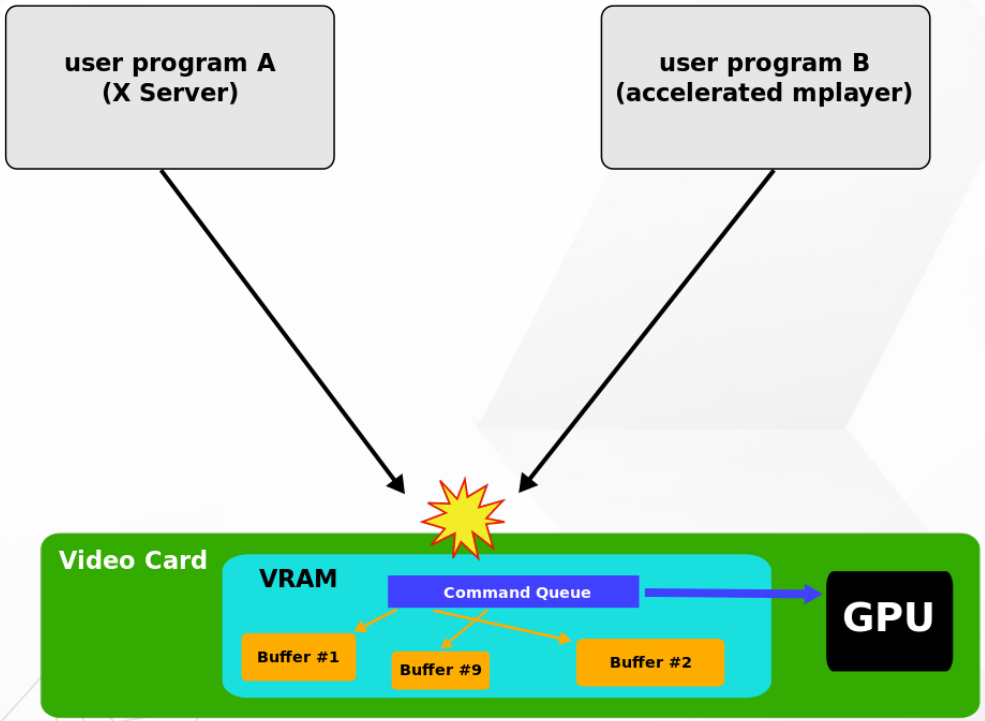
Direct Rendering Manager (DRM)

- Direct Rendering Manager
  - Management of buffers and free space within that memory.
  - Solve Frame buffer driver cannot be used GPU and multi-user process.

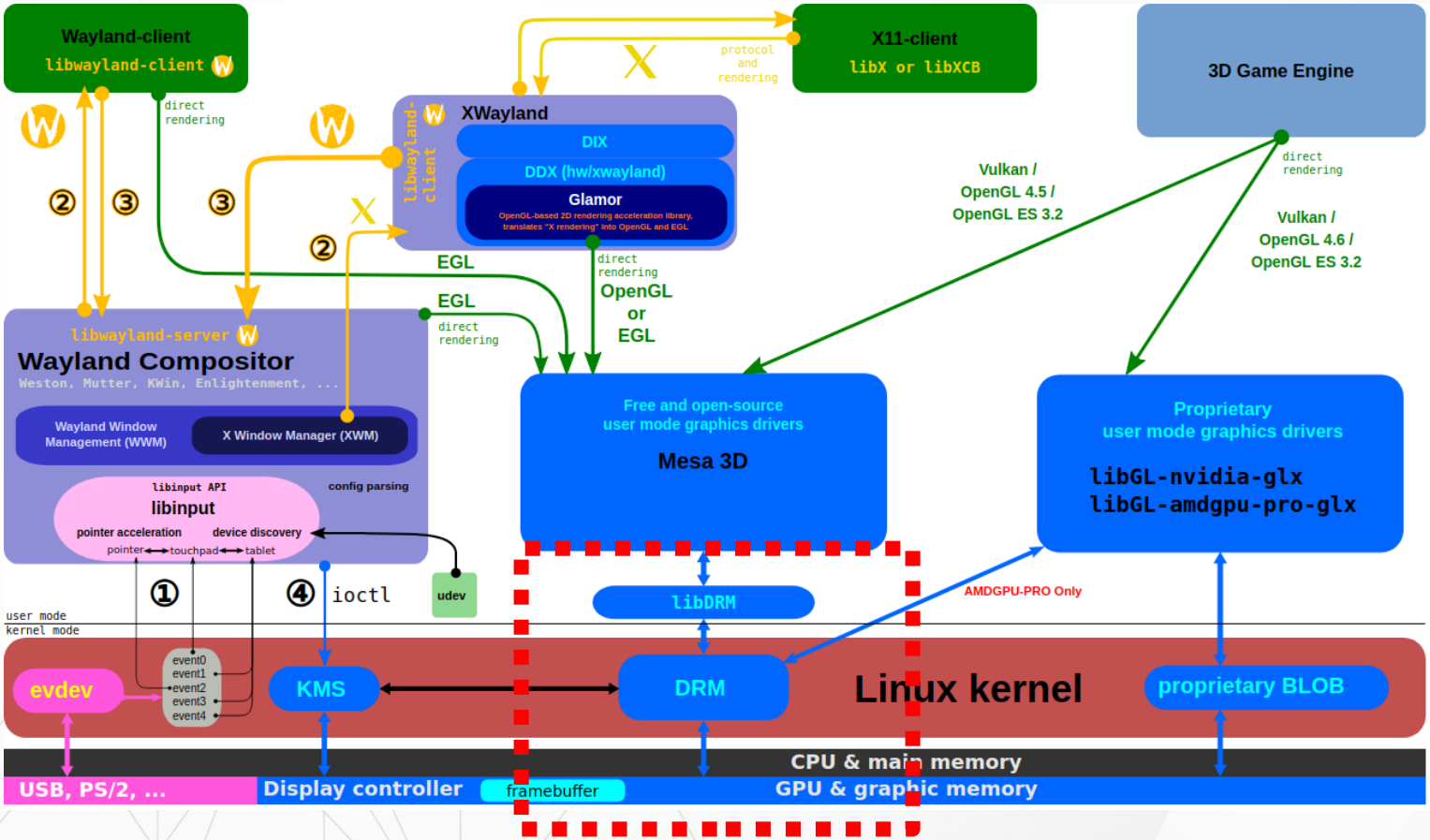
- DRM consists of
  - libdrm
    - libdrm provides a user space library for accessing the DRM
  - KMS : Kernel Mode Setting
    - Change resolution and depth
  - DRI : Direct Rendering Infrastructure
    - Interfaces to access hardware directly
  - GEM : Graphics Execution Manager
    - Buffer management
  - DRM Driver in kernel side
    - Access hardware

Direct Rendering Manager (DRM)

If no use DRM



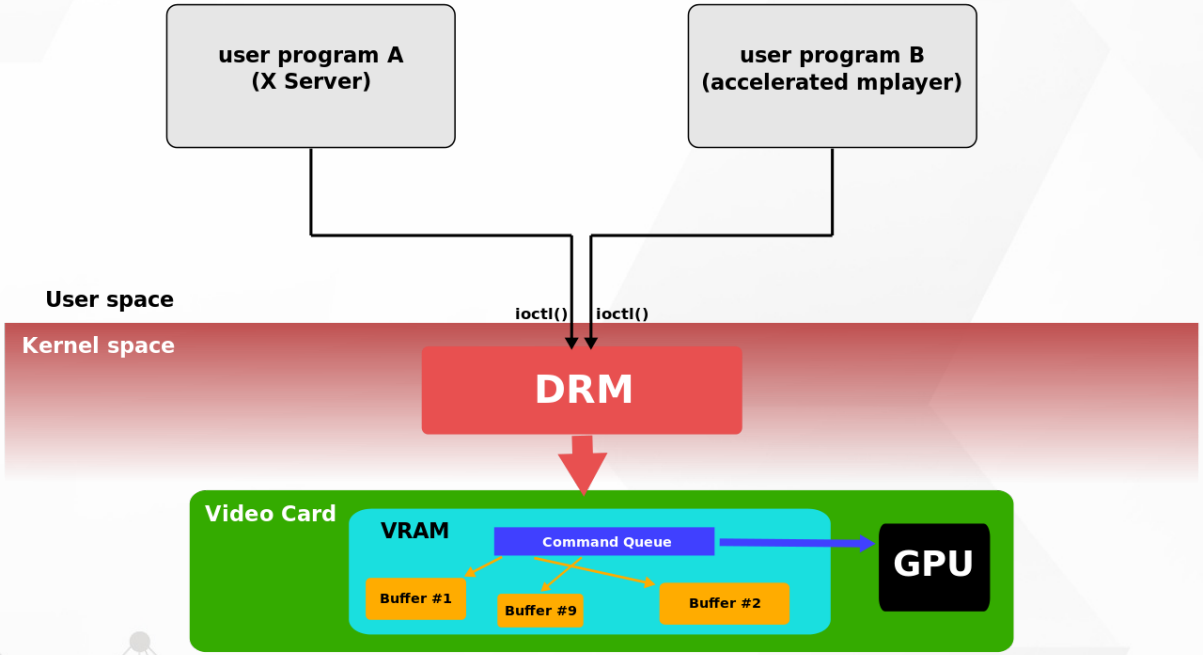
Direct Rendering Manager (DRM)



https://upload.wikimedia.org/wikipedia/commons/2/2d/The_Linux_Graphics_Stack_and_glamor.svg

Direct Rendering Manager (DRM)

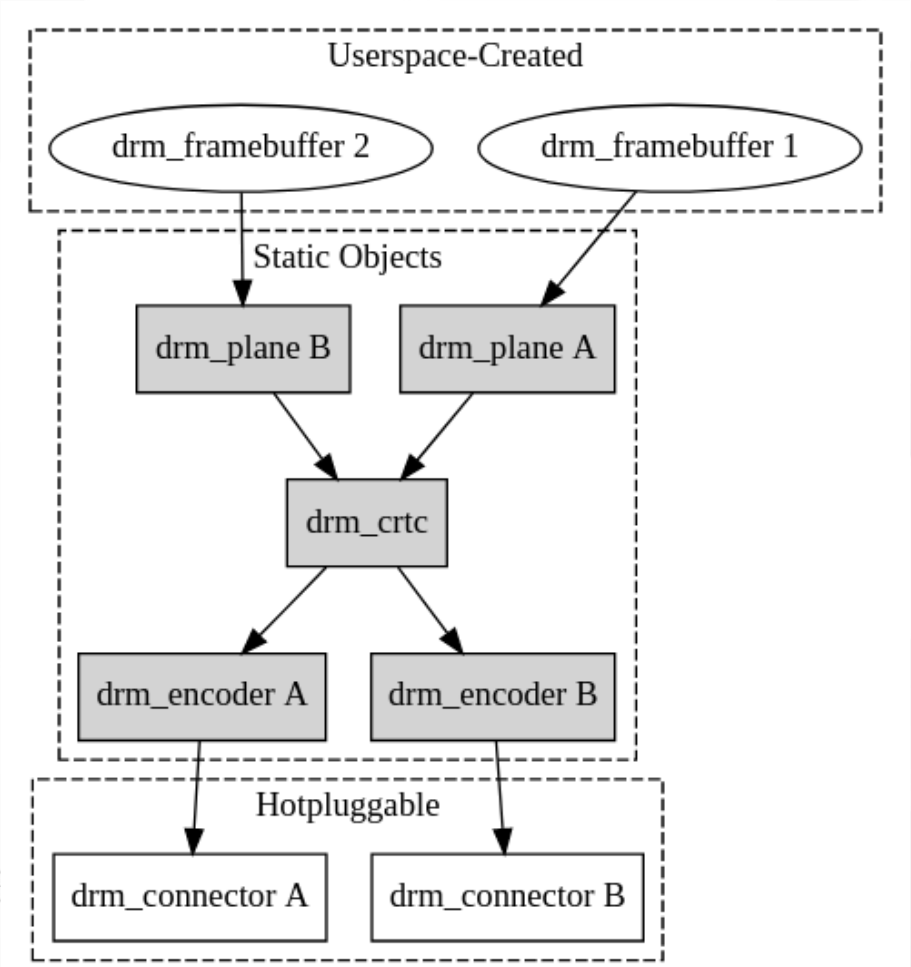
Use DRM



Kernel Mode Setting (KMS)

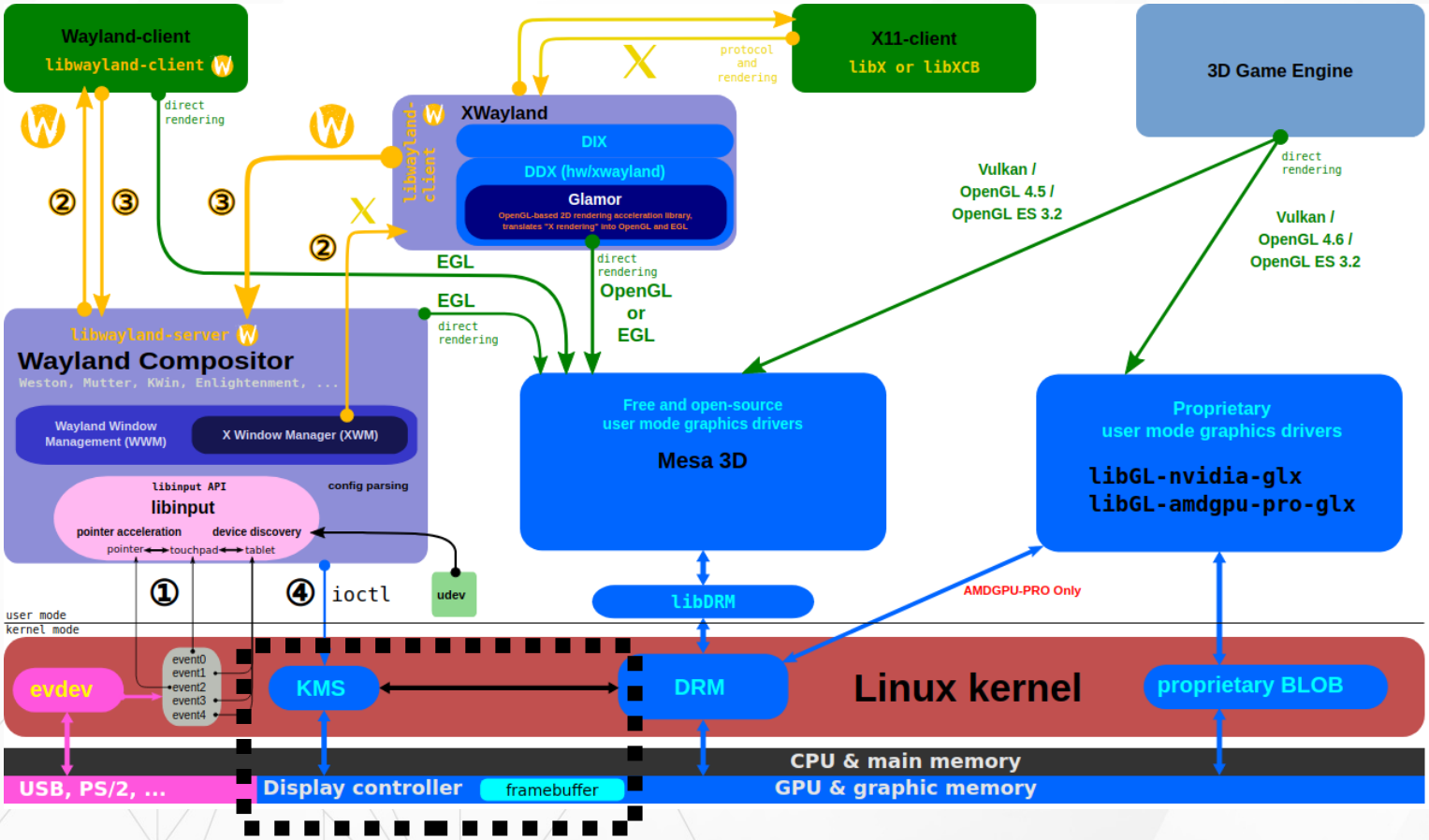
- KMS device model
  - CRTCs
  - Connectors
  - Encoders
  - Planes
- Kernel Mode Setting
  - screen resolution
  - color depth and
  - refresh rate

Kernel Mode Setting (KMS)



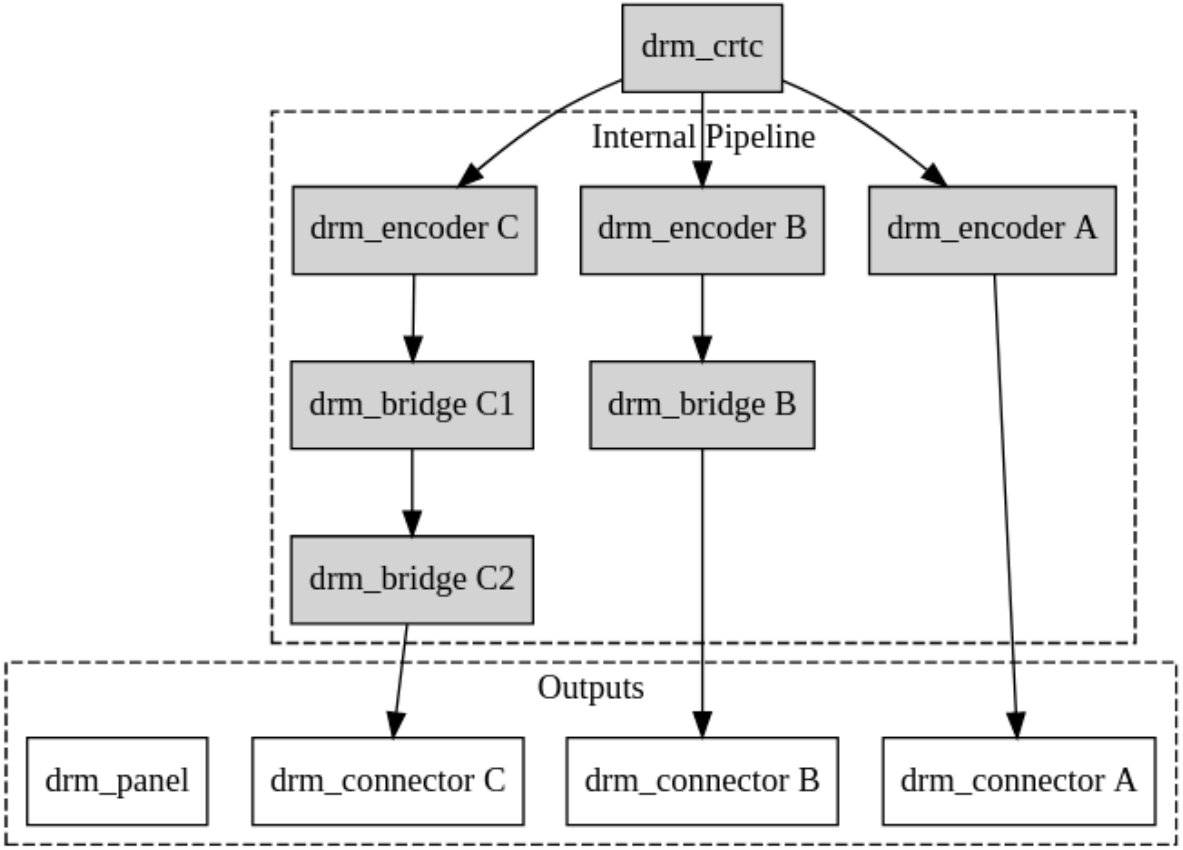
https://www.kernel.org/doc/html/v4.15/gpu/drm-kms.html

Kernel Mode Setting (KMS)



https://upload.wikimedia.org/wikipedia/commons/2/2d/The_Linux_Graphics_Stack_and_glamor.svg

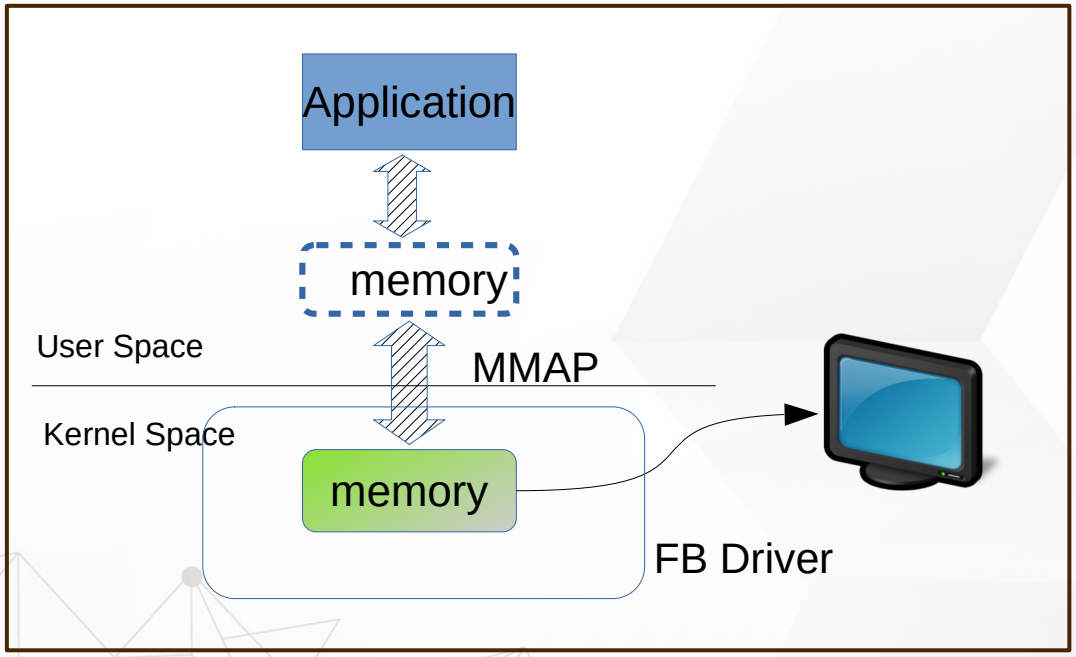
Kernel Mode Setting (KMS)



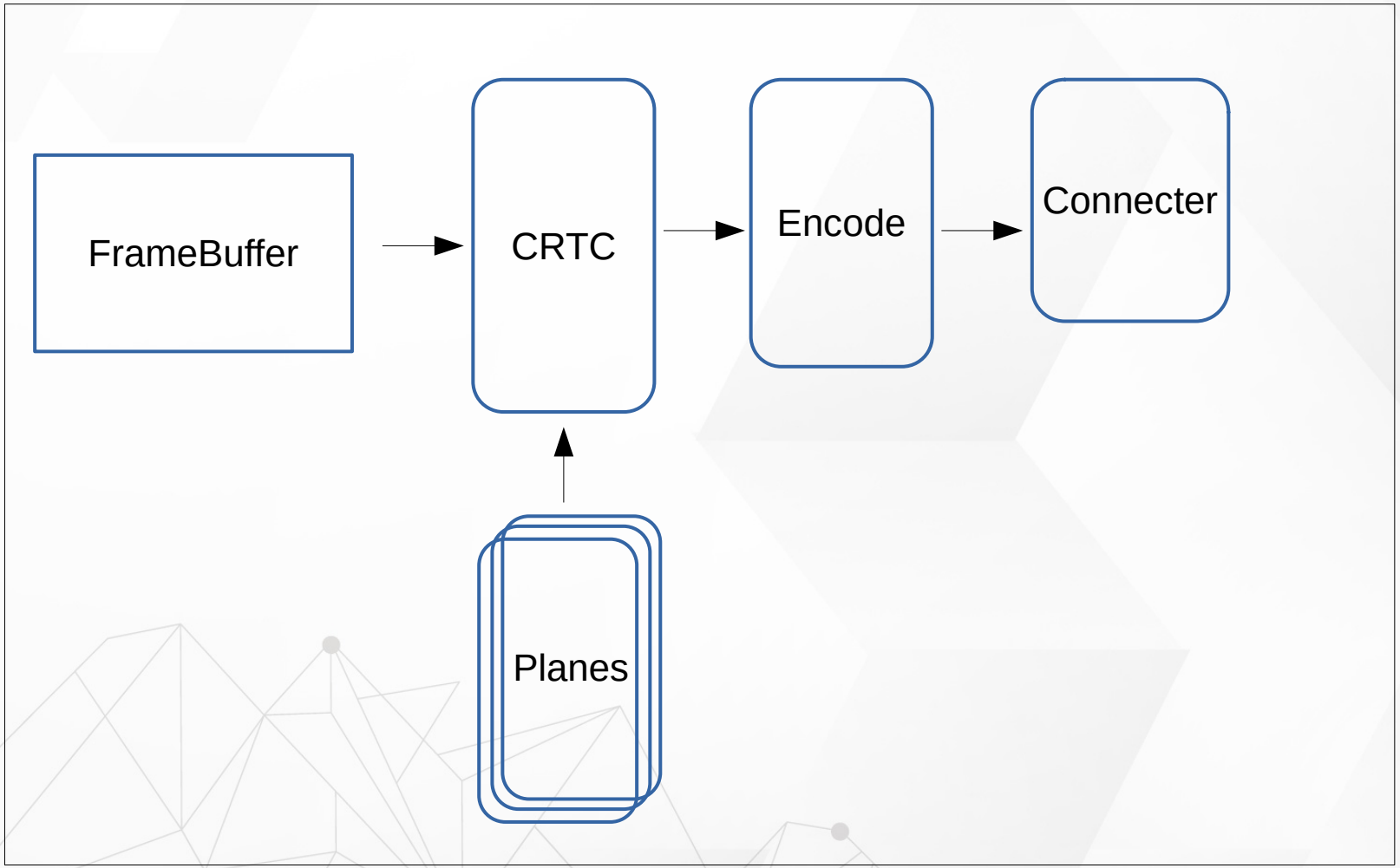
https://www.kernel.org/doc/html/v4.15/gpu/drm-kms.html

Video Frame Buffer

- The frame buffer device provides an abstraction for the graphics hardware.



Kernel Mode Setting (KMS)



Embedded Linux

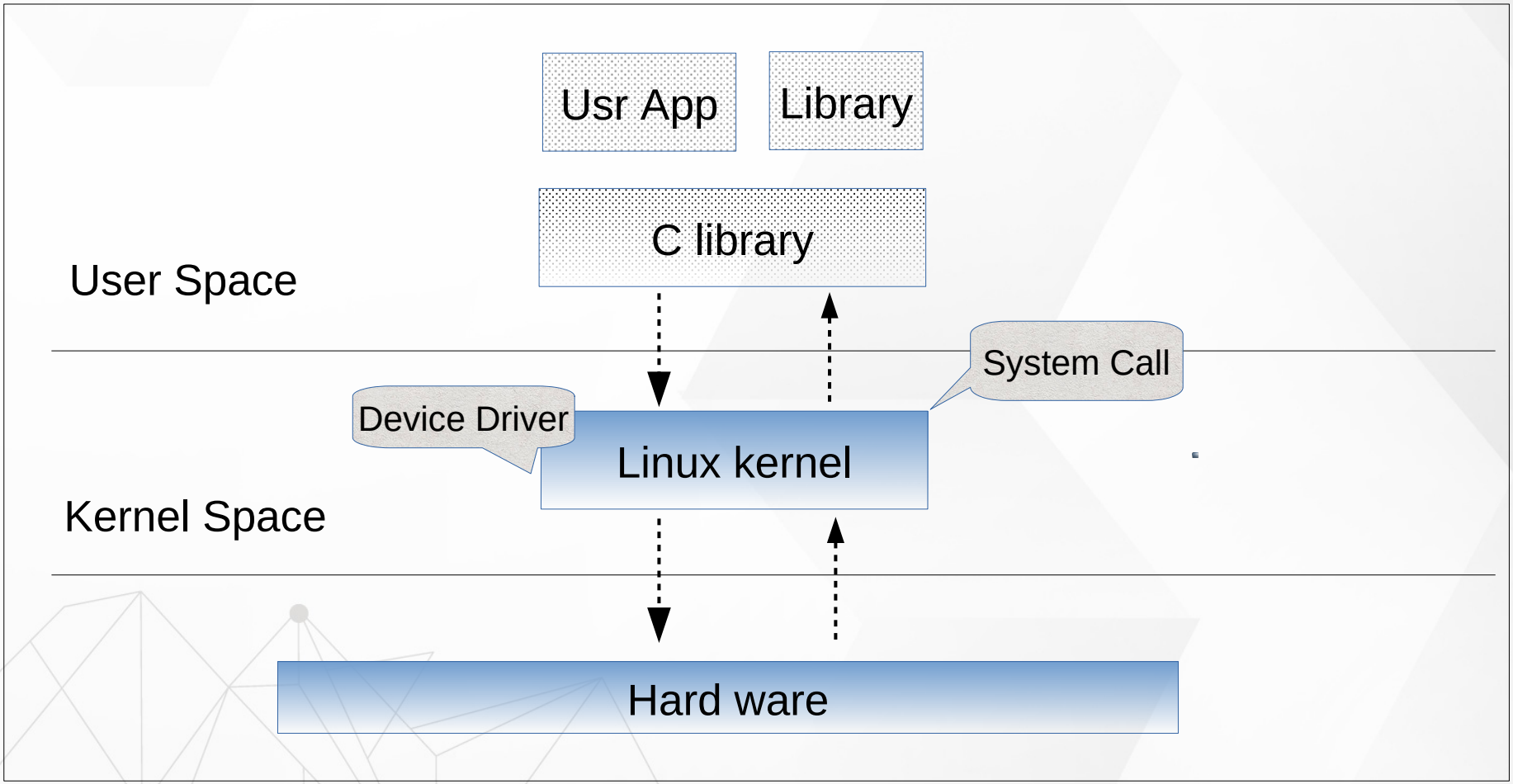
Various Layers within Linux

Various layers within Linux, also showing separation between the userland and kernel space

User mode	**User applications**	*bash*, *LibreOffice*, *GIMP*, *Blender*, *0 A.D.*, *Mozilla Firefox*, ...				
	System components	**init daemon**: *OpenRC*, *runit*, *systemd*...	**System daemons**: *polkitd*, *smbd*, *sshd*, *udevd*...	**Window manager**: *X11*, *Wayland*, *SurfaceFlinger* (*Android*)	**Graphics**: *Mesa*, *AMD Catalyst*, ...	**Other libraries:** *GTK*, *Qt*, *EFL*, *SDL*, *SFML*, *FLTK*, *GNUstep*, ...
	C standard library	`fopen`, `execv`, `malloc`, `memcpy`, `localtime`, `pthread_create` ... (up to 2000 subroutines) *glibc* aims to be fast, *musl* aims to be lightweight, *uClibc* targets embedded systems, *bionic* was written for Android, etc. All aim to be POSIX/SUS-compatible.				
Kernel mode	**Linux kernel**	`stat`, `splice`, `dup`, `read`, `open`, `ioctl`, `write`, `mmap`, `close`, `exit`, etc. (about 380 system calls) The Linux kernel System Call Interface (SCI), aims to be POSIX/SUS-compatible[2]				
		Process scheduling subsystem	IPC subsystem	Memory management subsystem	Virtual files subsystem	Network subsystem
		Other components: ALSA, DRI, evdev, klibc, LVM, device mapper, Linux Network Scheduler, Netfilter Linux Security Modules: *SELinux*, *TOMOYO*, *AppArmor*, *Smack*				
Hardware (CPU, main memory, data storage devices, etc.)						

https://en.wikipedia.org/wiki/User_space_and_kernel_space

Embedded Linux System

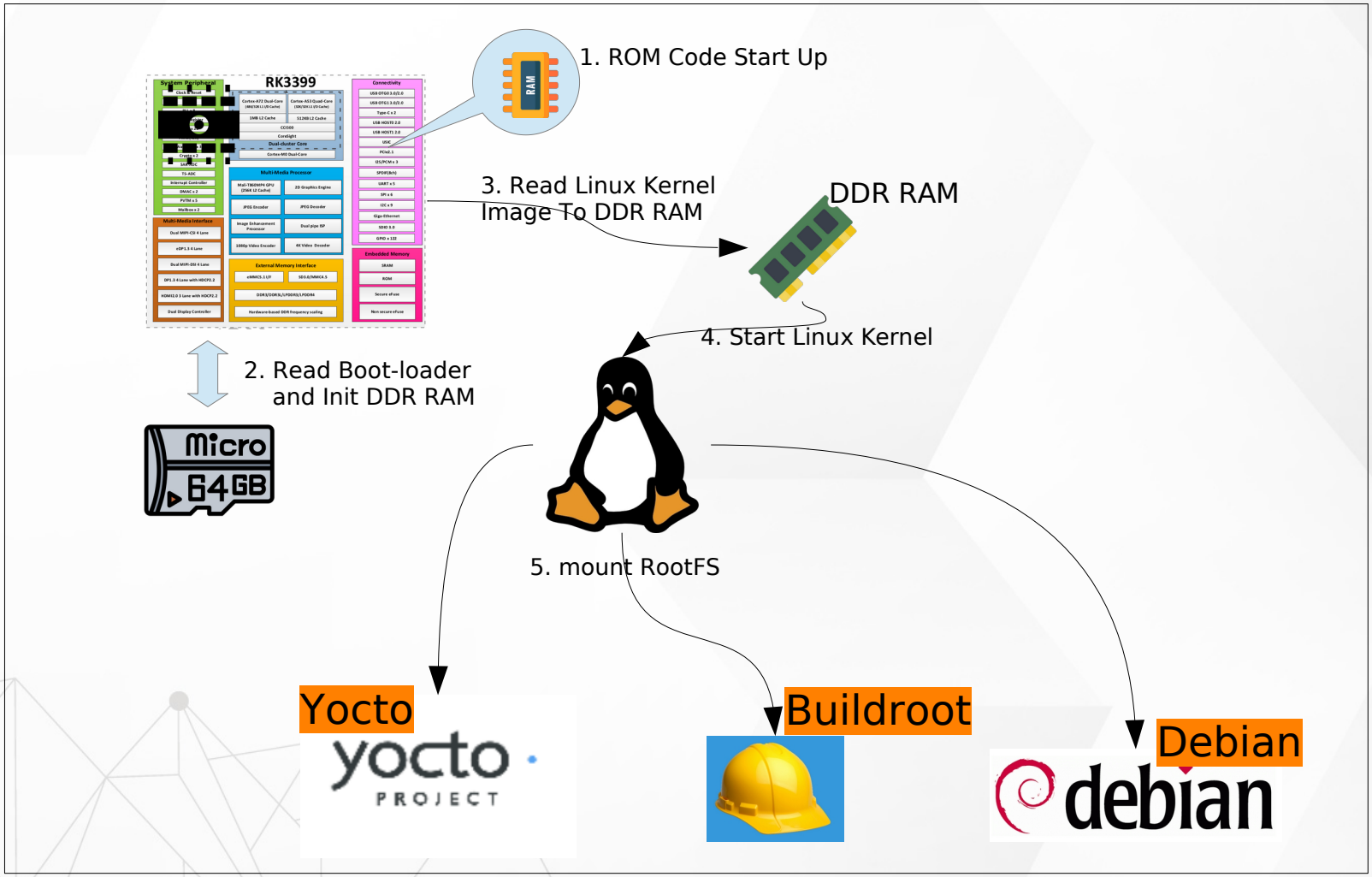


Linux kernel key features

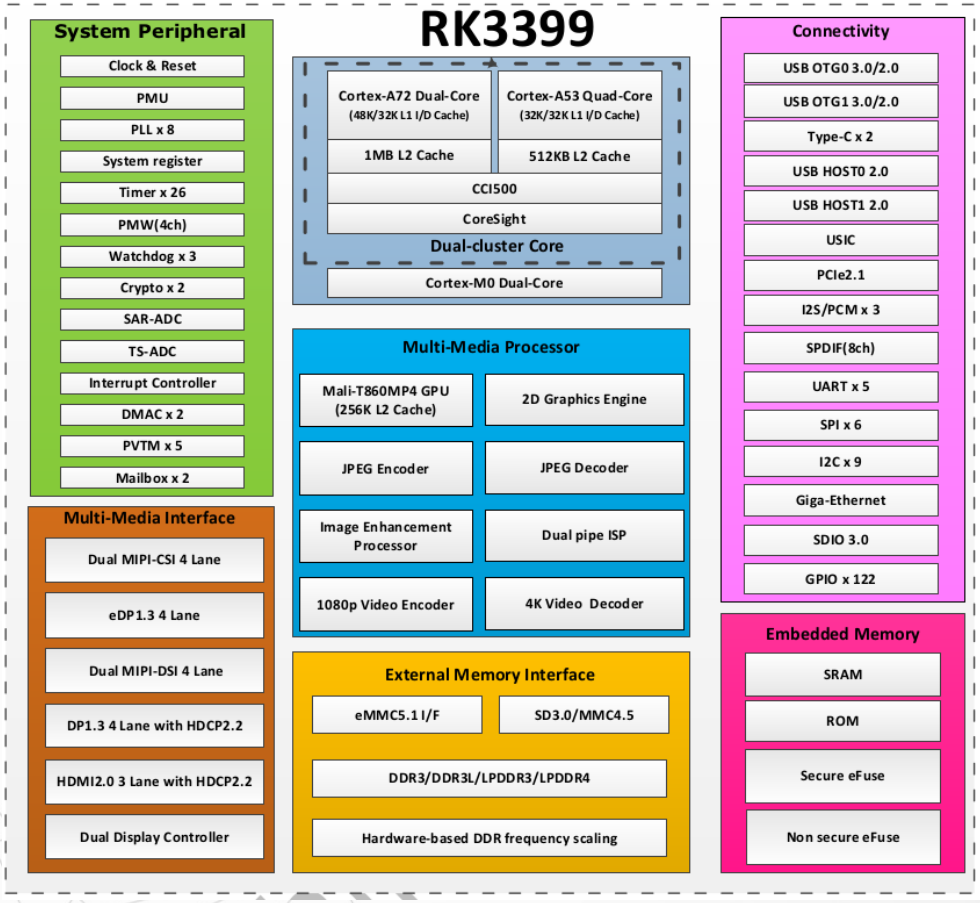
- Portability and hardware support
- Scalability
- Exhaustive networking support
- Stability and reliability
- Modularity
- Easy to program.

System Start Up

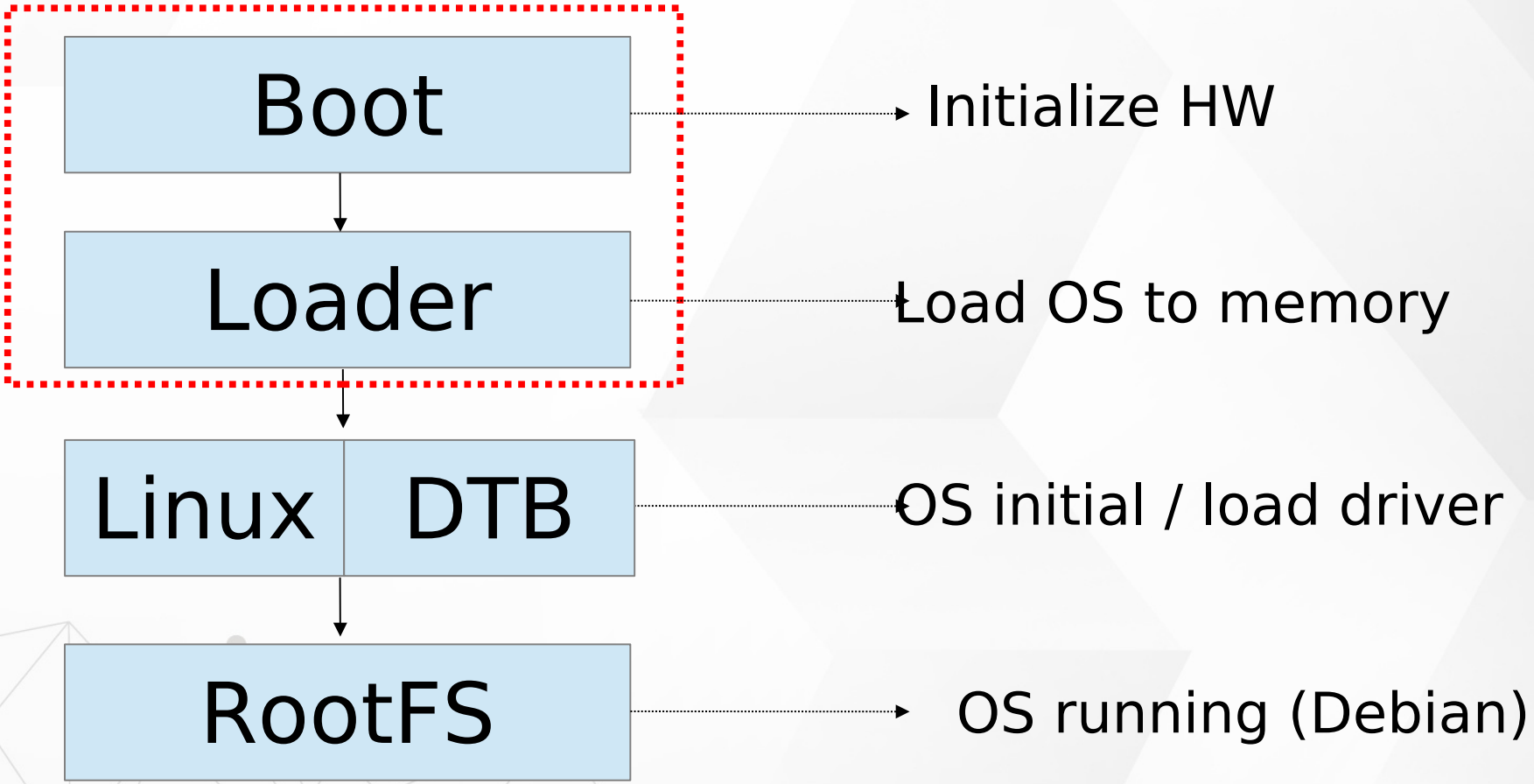
Linux Start Up



SOC RK3399



Embedded Linux System Booting

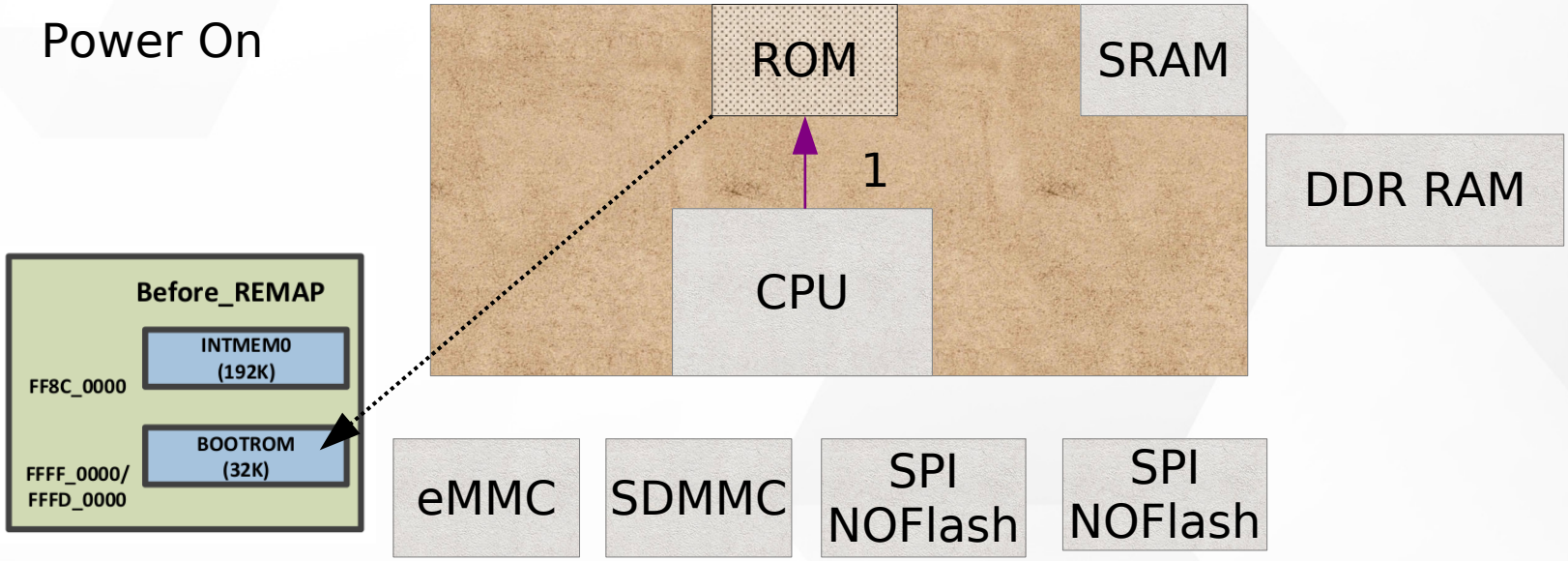


RK3399 System Boot

Reference:

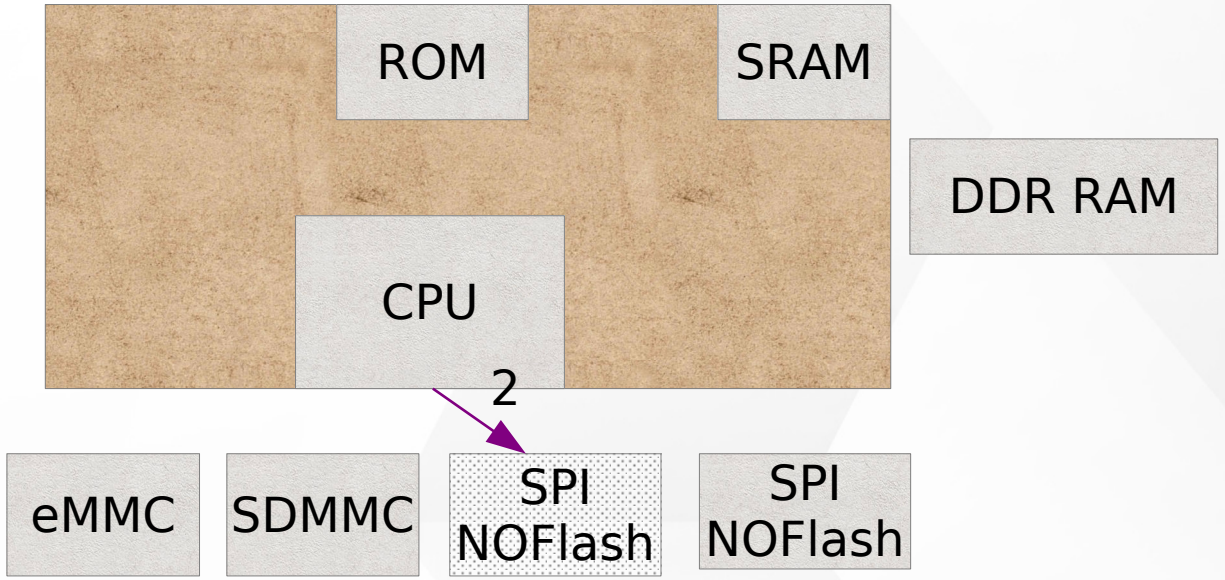
Page 30 of Rockchip_RK3399TRM_V1.3_Part1.pdf

RK3399 System Boot (1)



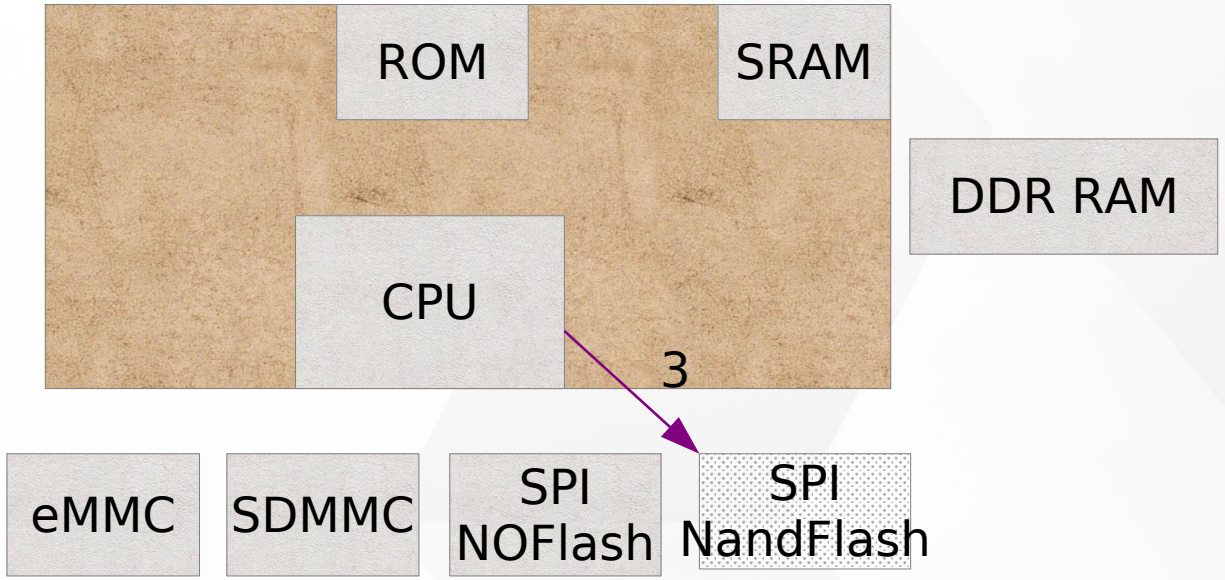
Cortex-A53 get first instruction from address **0xffff0000 romcode** start to run

RK3399 System Boot (2)



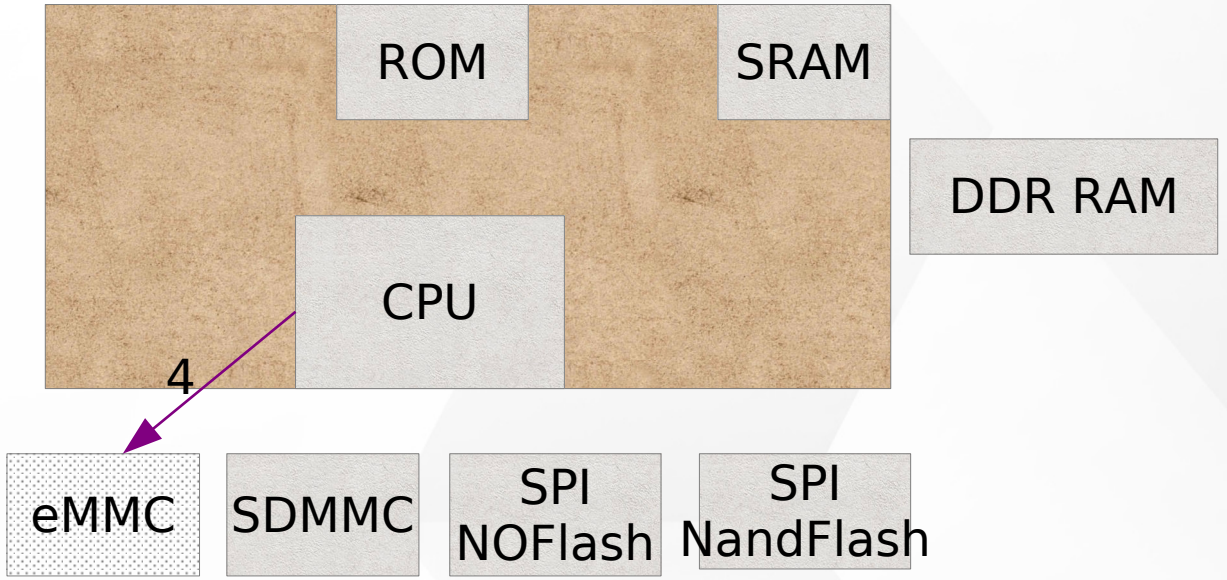
Check ID BLOCK from external **SPI Nor Flash**

RK3399 System Boot (3)



Check ID BLOCK fromexternal **SPI Nand Flash**

RK3399 System Boot (4)



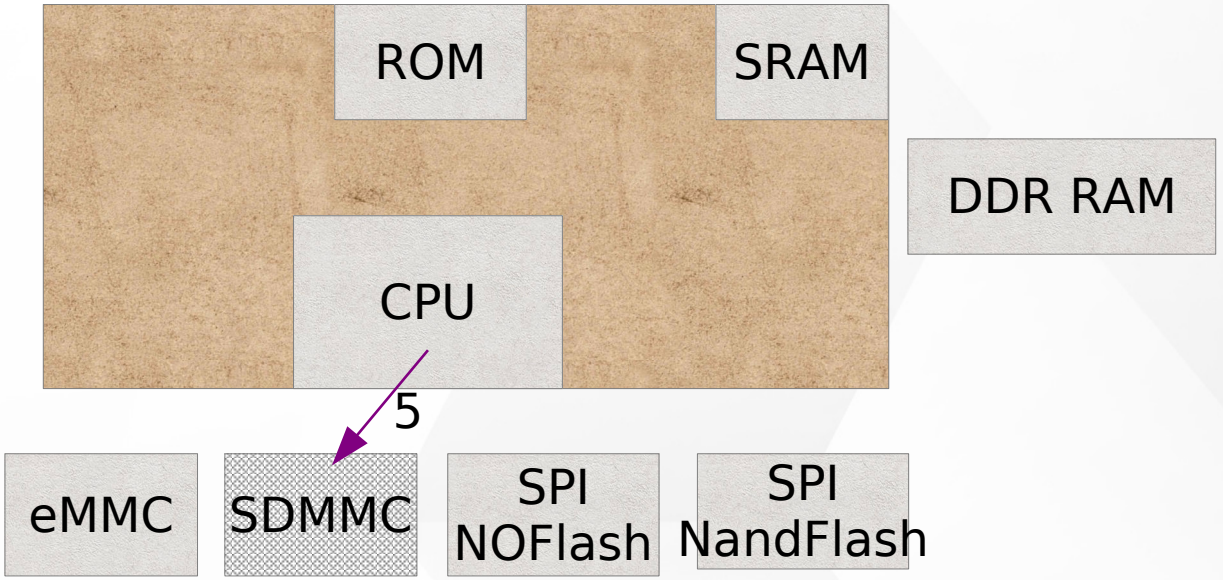
Check ID BLOCK from external **eMMC**

RK3399 System Boot (5)

BL1

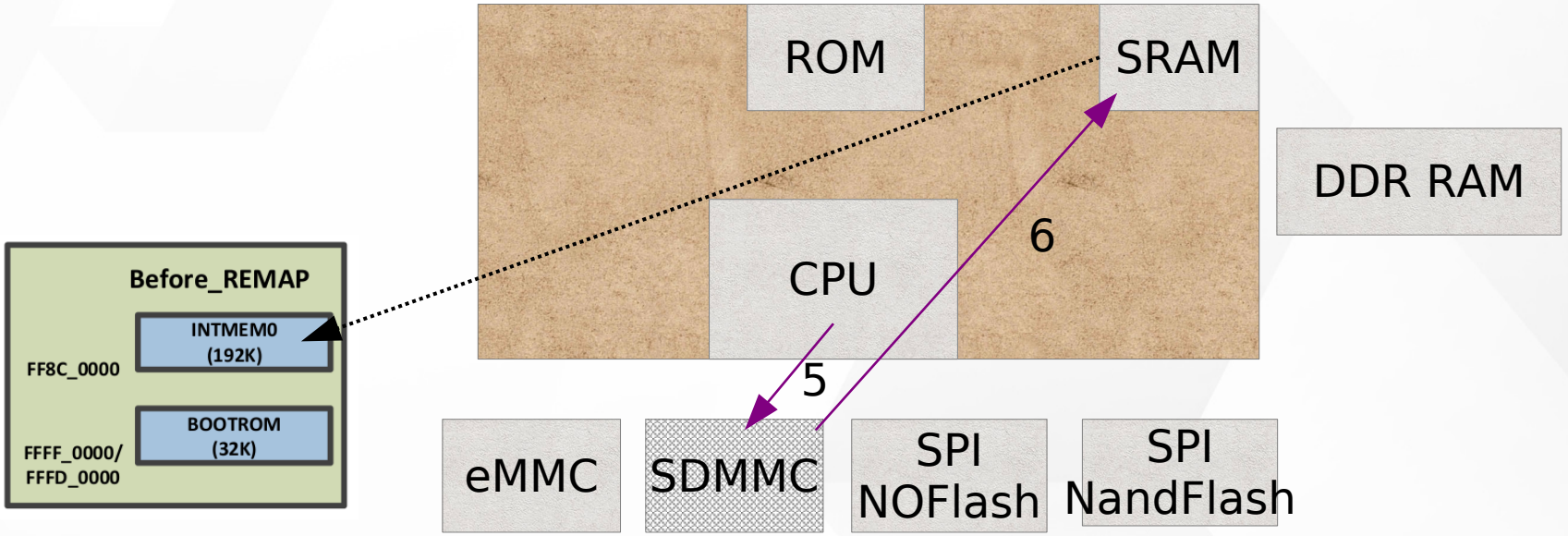
work in cache

IDB_Loader



Check ID BLOCK from external **SDMMC**

RK3399 System Boot (6)



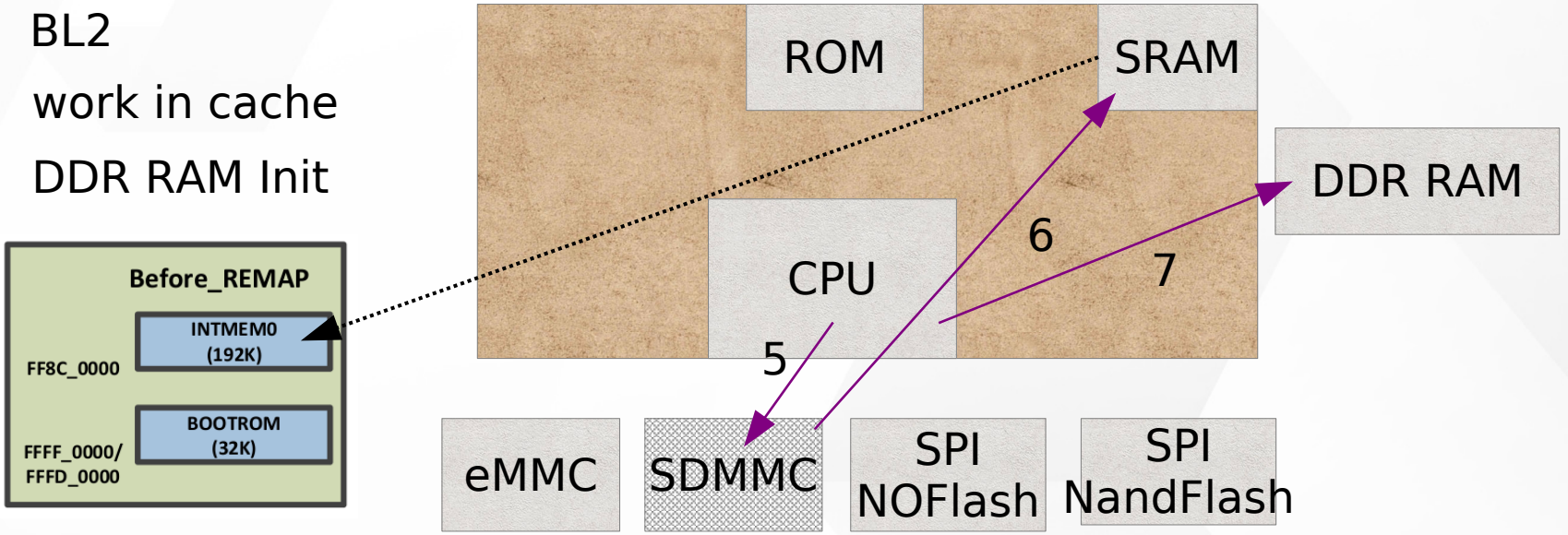
5. Check ID BLOCK from external **SDMMC**
6. Read **2nK** SDRAM initialization image code to **internal SRAM**

RK3399 System Boot (7)

BL2

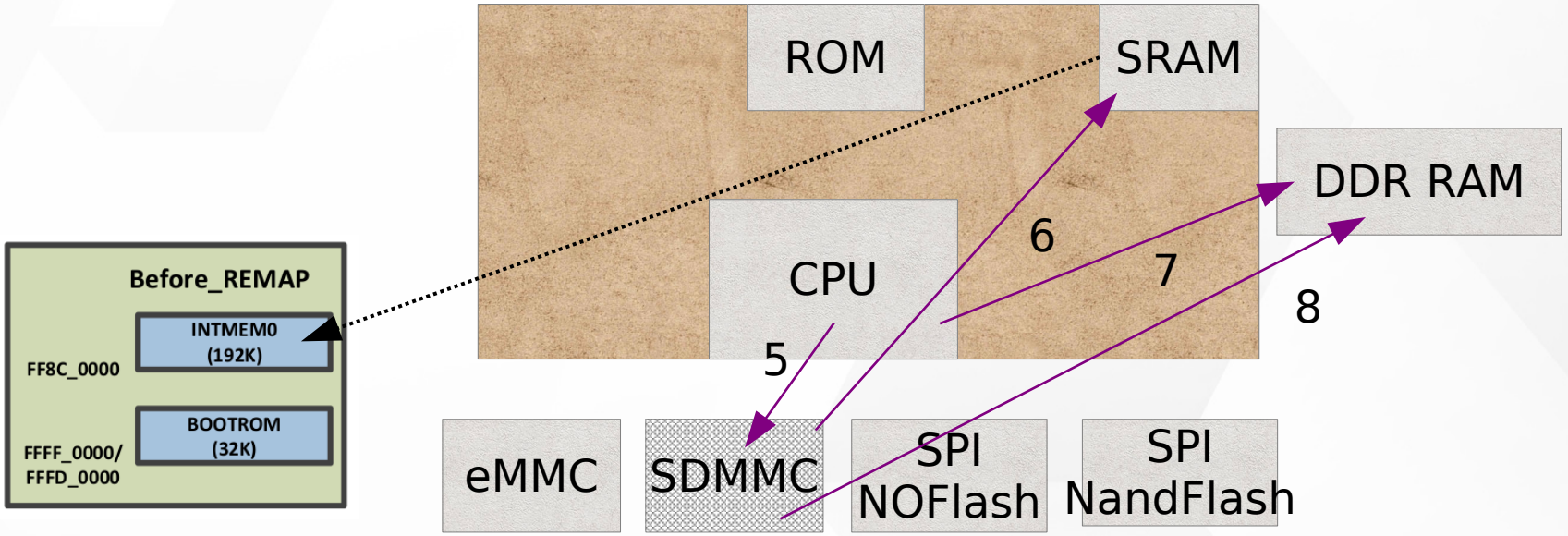
work in cache

DDR RAM Init



5. Check ID BLOCK from external **SDMMC**
6. Read **2nK** SDRAM initialization image code to **internal SRAM**
7. Run boot code to do DDR initialization

RK3399 System Boot (8)



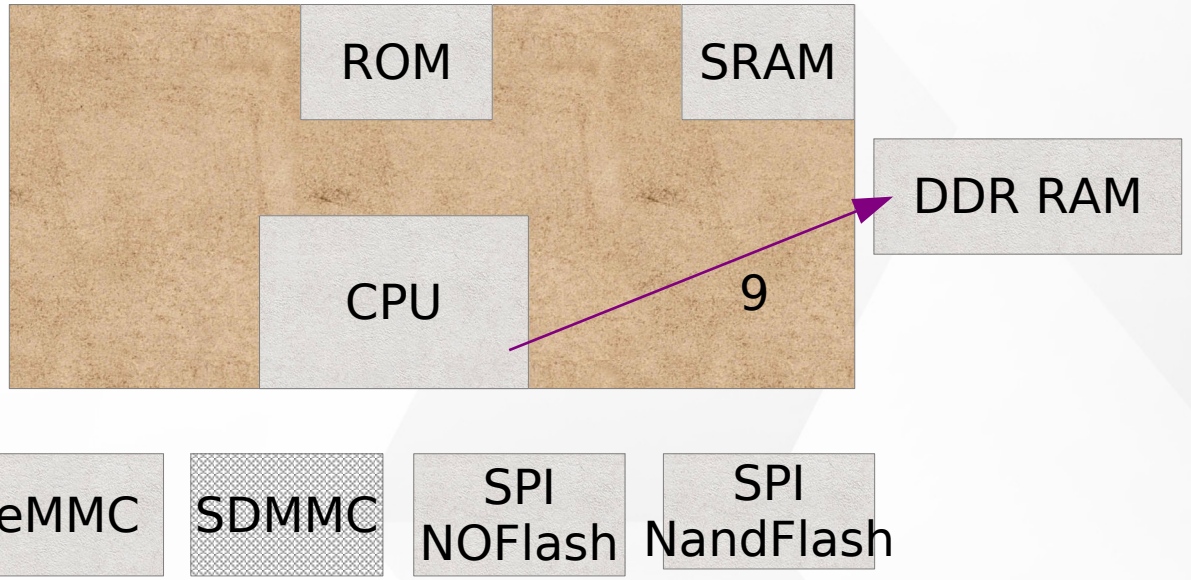
5. Check ID BLOCK from external **SDMMC**
6. Read **2nK** SDRAM initialization image code to **internal SRAM**
7. Run boot code to do DDR initialization
8. Transfer boot code to DDR then Run boot code

RK3399 System Boot (9)

BL3

work in DDR RAM

UBoot

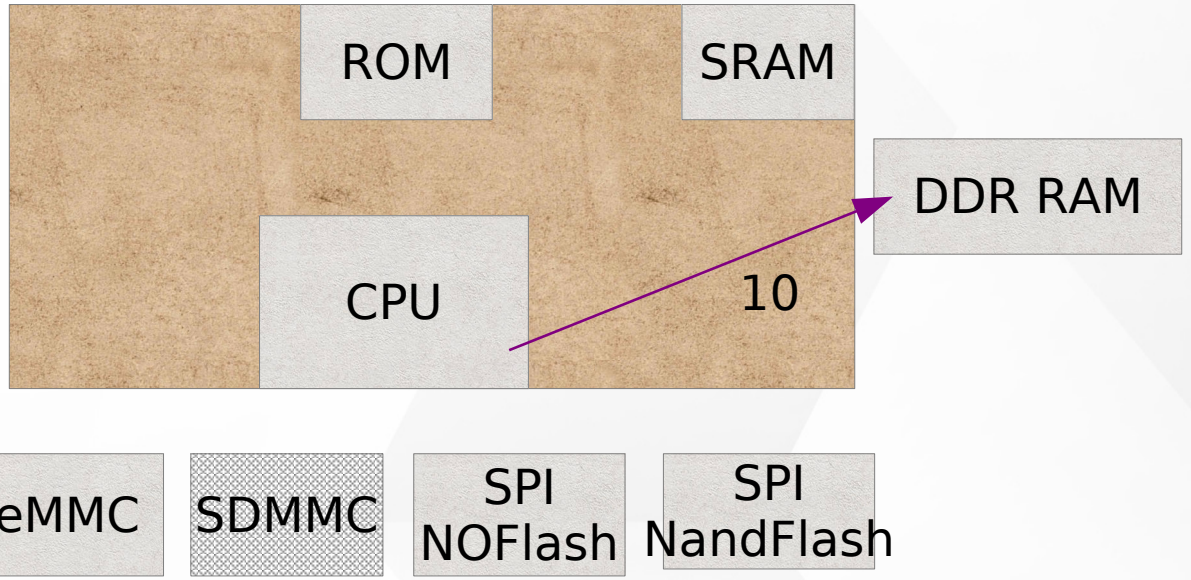


Run UBoot

RK3399 System Boot (10)

work in DDR RAM

Linux



Run Linux

Boot

- Power On BootROM code (work in **cache**)
  - Load BL1
- BL1 (work in **cache** - IDB_Loader)
  - **Initial simple exception vectors, PLL** (clock)
  - **Initial Multi-CPU**
  - Load BL2
- BL2 (work in **cache**)
  - **Initial DDR memory**
  - **Initial C environment** (stack, heap,)
  - Load BL31

Boot

- BL31 (**work in DDR**)
  - Initial exception vectors
  - Load BL32 (u-boot)

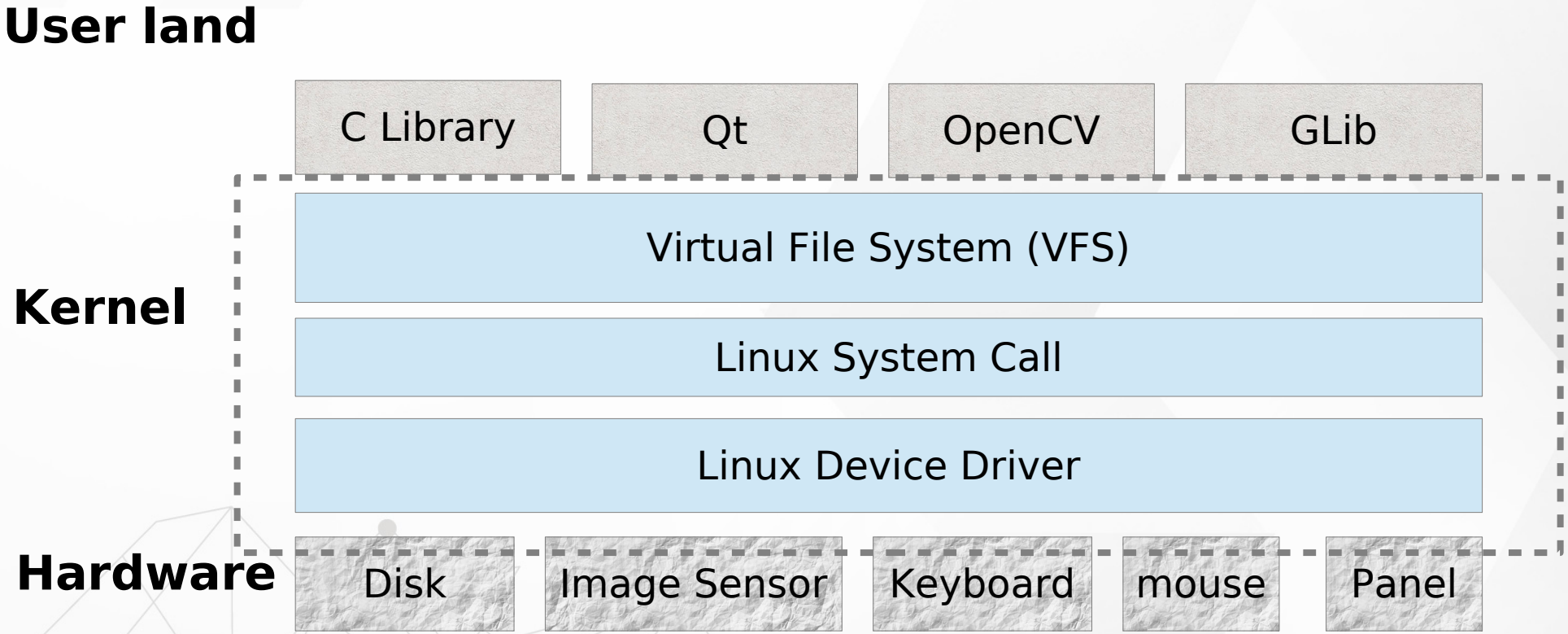
BL32 U-boot (**work in DDR**)

- **Initial storage device**
- **Load Linux Kernel**

Kernel (**work in DDR**)

- kernel/Documentation/arm64/booting.txt
- **Load RootFS**

Embedded Linux System



CH4 Basic Software & Tool

Software and Tool

- Open Source License
- Develop Tool
  - → Geany, gedit, vim
  - → Git
  - → diff, patch
- Build Code Tool
  - → ARM Toolchain
  - → make
  - → automake, autoconfig

Software and Tool

- Network
  - → nmcli, ethtool, wpa_supplicant
  - → SSH, SSHFS
  - → NFS
- Bluetooth
  - → hciconfig, bluetoothctl
- Media Software
  - → gstreamer
  - → ALSA Tool - aplay, arecord

Software and Tool

- Bus
  - → I2C – I2cset, i2cget, i2cdump
  - → USB – lsusb

Package Manage

Debian Package Management

- dpkg : command-line tool for handling packages
- dpkg --help
  - → [CMD] dpkg -i ${PACKAGE_NAME} : install packag
  - → [CMD] dpkg -r ${PACKAGE_NAME} : remove package

Ubuntu Package Management

- apt-get : command-line tool for handling packages
- apt-get --help
  - → [CMD] apt-get update
  - → [CMD] apt-get install ${PACKAGE_NAME}
  - → [CMD] apt-get remove ${PACKAGE_NAME}
  - → [CMD] apt-get autoremove
  - → [CMD] apt-get clean

Editor

Editor

- Geany
  - [CMD] sudo apt-get install geany
- Vim
  - [CMD] sudo apt-get install vim
- Gedit
  - [CMD] sudo apt-get install gedit

Diff Tool

diff and patch

- diff - compare files line by line
  - Create a patch file
    - diff -Nuar file_a file_b > c.patch
      - -N, treat absent files as empty
      - -a, --text
      - -u, output NUM (default 3) lines of unified context
      - -r, recursively compare any subdirectories found
- patch - apply a diff file to an original
  - apply a patch file
    - patch ./hello_1.c < ./tmp.patch
  - Reverse a patch file
    - patch -R ./hello_1.c < tmp.patch

Source Code Version Control

Git

- https://git-scm.com/book/zh-tw/v1/
- 版本控制
- 程式回溯
- 管理多人共同開發

Makefile

Makefile

- Simplify compile command
- Automation compile, linker program source
- It can update source in accordance with the dependence

Compile a Hello_World

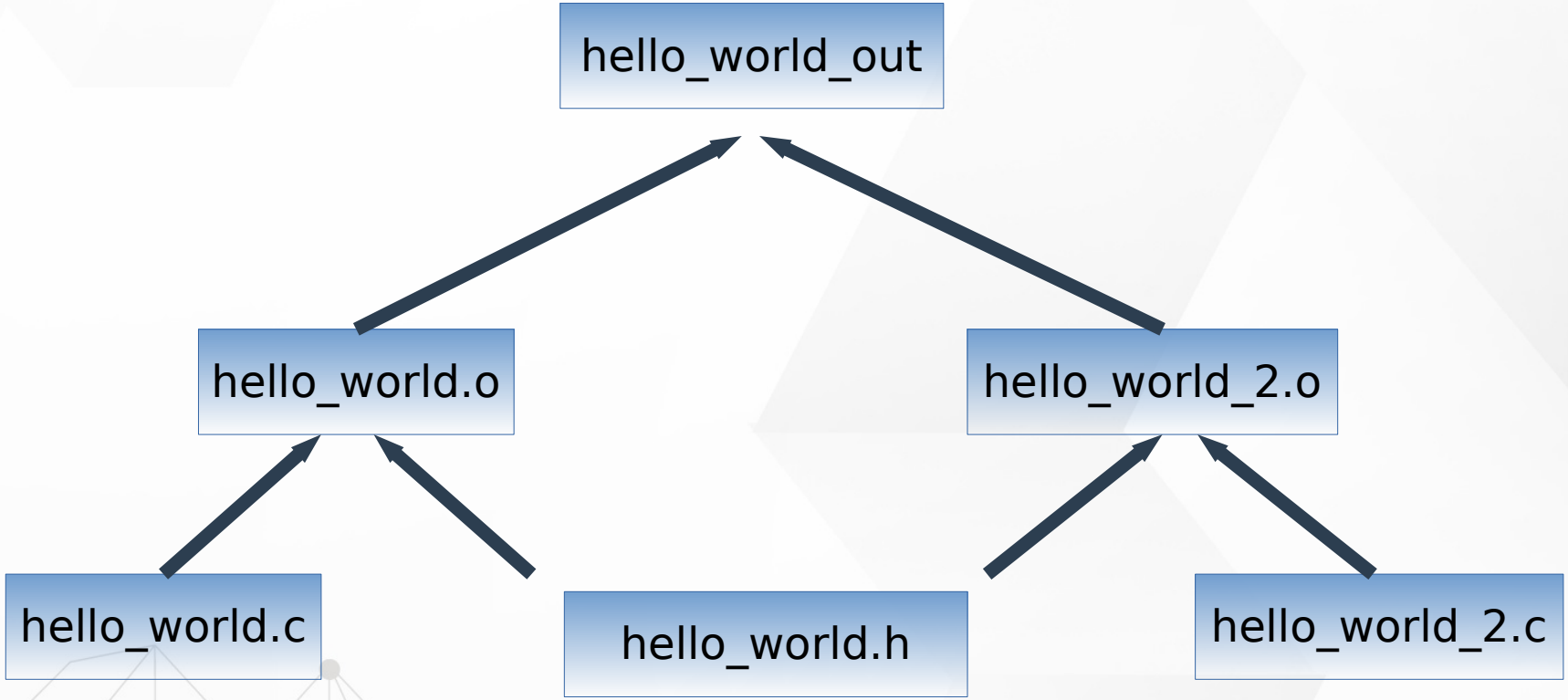
A	B	C
aarch64-linux-gnu-gcc	-o helloworld	./hello.c

A : ARM C Compile

B : ARM C Compile Parameter
(Output name)

C : C source code

Another Sample



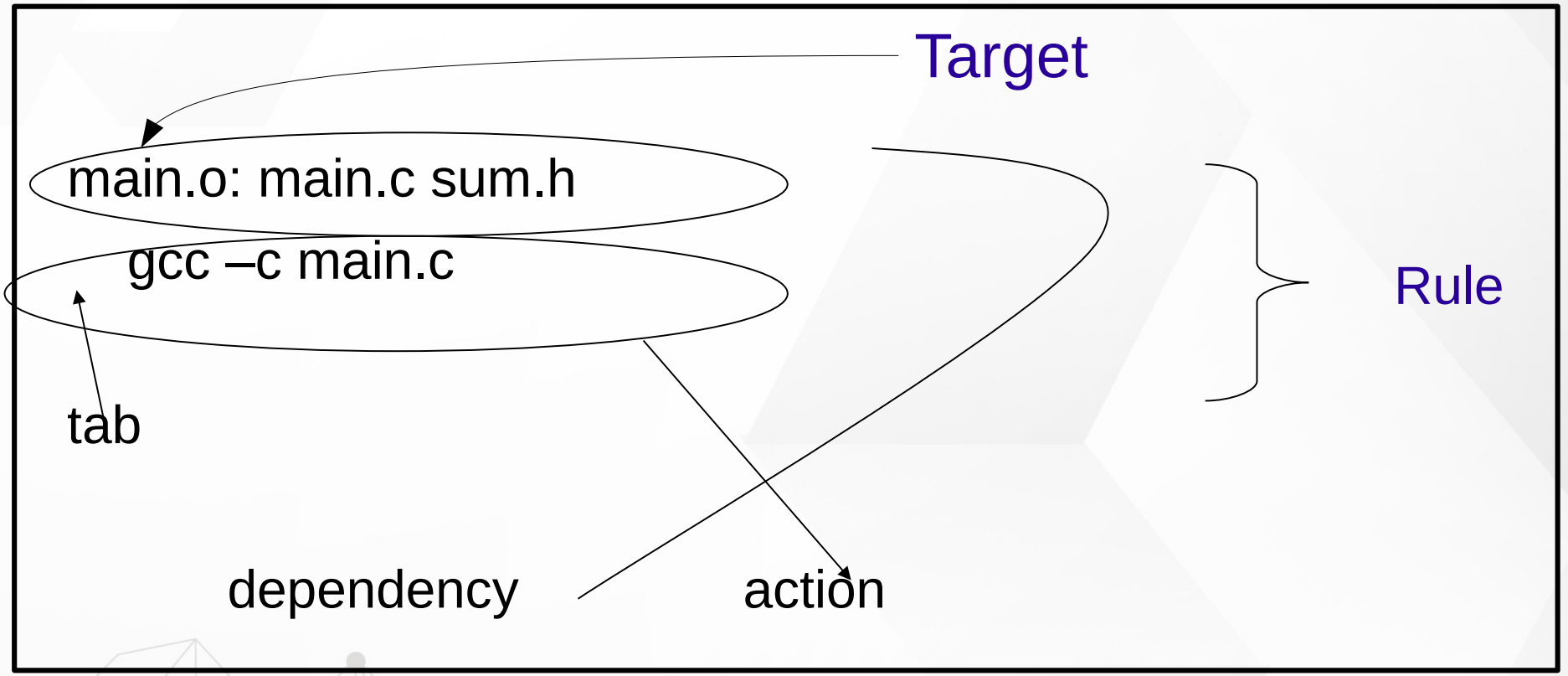
Compile Another Sample

- Step 1 : gcc -c hello_world.c
- Step 2 : gcc -c hello_world_2.c
- Step 3 : gcc -o hello_world hello_world.o hello_world_2.o

Another Sample - Makefile

```
CC=$(CROSS_COMPILE)gcc


all: hello_world

hello_world: hello_world.o hello_world_2.o
	$(CC) -o hello_world hello_world.o hello_world_2.o

hello_world.o: hello_world.c
	$(CC)  -c hello_world.c

hello_world_2.o: hello_world_2.c
	$(CC)  -c hello_world_2.c

clean:
	rm -r *.o
	rm hello_world
```

Rule Syntax



make 在編譯時，若發現 target 比較新，
也就是 dependencies 都比 target 舊，
那麼將不會重新建立 target，如此可以避免不必要的編譯動作

Rule Syntax

```
CC=$(CROSS_COMPILE)gcc

CFLAGS=-Wl,-Map,out.map -lpthread -lm

all: hello_world

hello_world: hello_world.o hello_world_2.o
	$(CC) -o hello_world hello_world.o hello_world_2.o

hello_world.o: hello_world.c
	$(CC) $(CFLAGS) -c hello_world.c

hello_world_2.o: hello_world_2.c
	$(CC) $(CFLAGS) -c hello_world_2.c

clean:
	rm -r *.o
	rm hello_world
```

hello_world_ex1

```
CC=$(CROSS_COMPILE)gcc

AA ='1234' '5678'
AA := 'DDDD'

$(info AA=$(AA))

CFLAGS=-Wl,-Map,out.map -lpthread -lm

all: hello_world

hello_world: hello_world.o
	$(CC) -o hello_world hello_world.o

hello_world.o: hello_world.c
	$(CC) $(CFLAGS) -c hello_world.c

clean:
	rm -r *.o
	rm hello_world
```

: Tab

Assignment Operators

=	定義一個 需做遞迴展開的 變數型態
:=	定義一個 立即運作的 變數型態
+=	將 指定值，續加在 原變數中
?=	如果之前 無任何設定該變數，即現在設定， 否則 跳過設定 (就是不做任何事)

Assignment Operators Sample 1

```
AA ='1234' '5678'
BB = ${AA}
AA = '789'
AA += 'ABCDE'
```

Output

```
AA='789' 'ABCDE'
BB='789' 'ABCDE'
```

Assignment Operators Sample 2

```
AA ='1234' '5678'
BB := ${AA}
AA = '789'
AA += 'ABCDE'
```

Output

```
AA='789' 'ABCDE'
BB='1234' '5678'
```

Assignment Operators Sample 3

```
AA ='1234' '5678'
BB := ${AA}
AA = '789'
AA ?= 'ABCDE'
```

Output

```
AA='789'
BB='1234' '5678'
```

Command-Line Options

- -C dir, --directory= dir
  - make changes the current working directory to dir before it does anything else. If the command line includes multiple -C options, each directory specified builds on the previous one
- -j [number] , --jobs[= number]
  - Run multiple commands in parallel

Bus Tool

Device Tool

- Real Time Clock
  - hwclock
    - -s, --hctosys set the system time from the RTC
    - -r, --show display the RTC time
- USB
  - lsusb
- Block Device
  - lsblk
- I2c-tools
  - I2cdump, i2cdetect, i2cget, i2cset

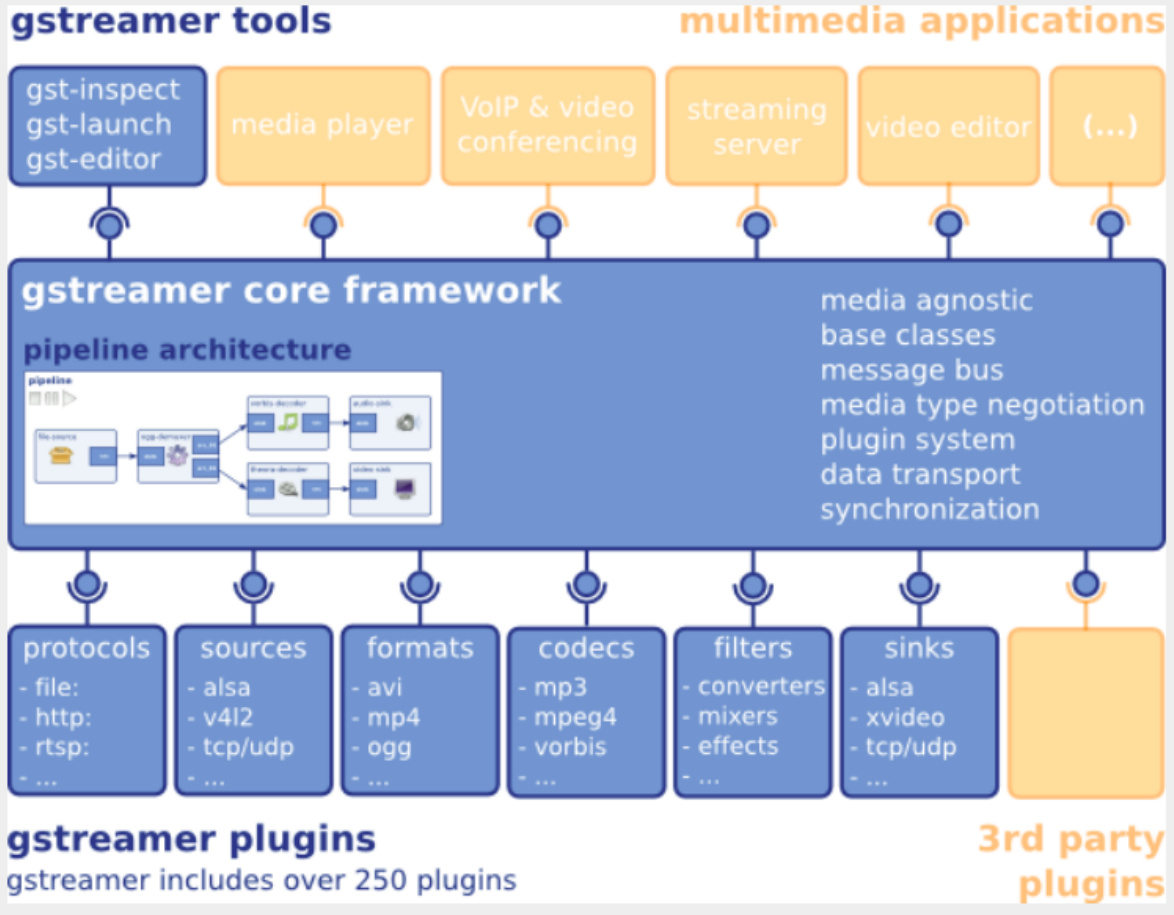
Media Tool

Gstreamer

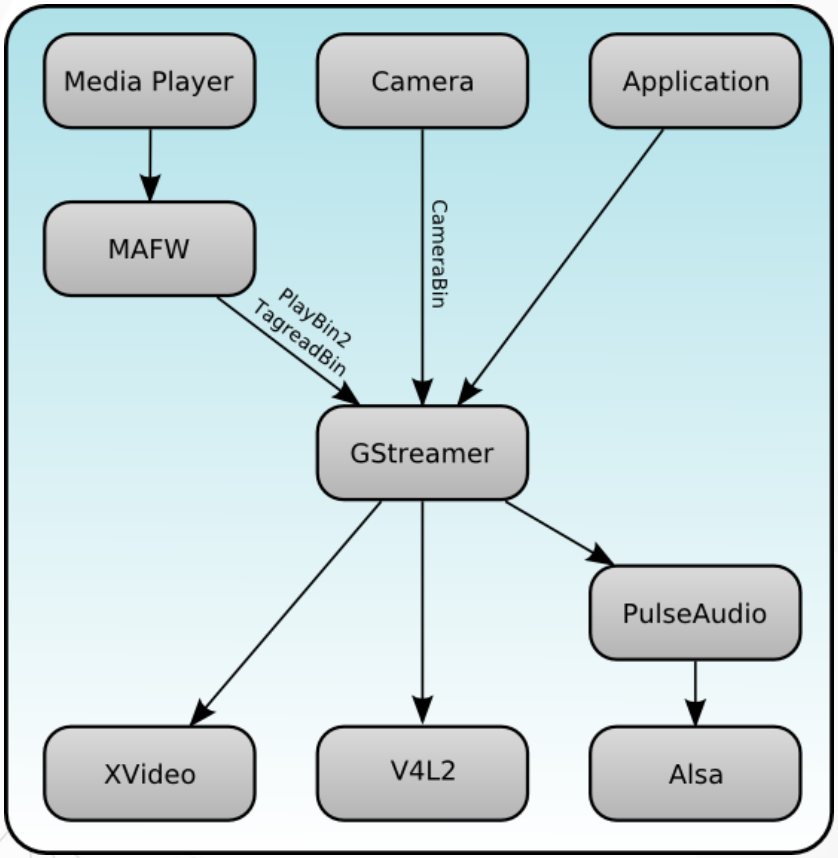
Open Source Multimedia Framework

- https://gstreamer.freedesktop.org/
- Documentation
  - Tutorials

Block Diagram



Block Diagram



http://maemo.org/development/sdks/maemo_5_beta_docs/using_multimedia_components/

Overview

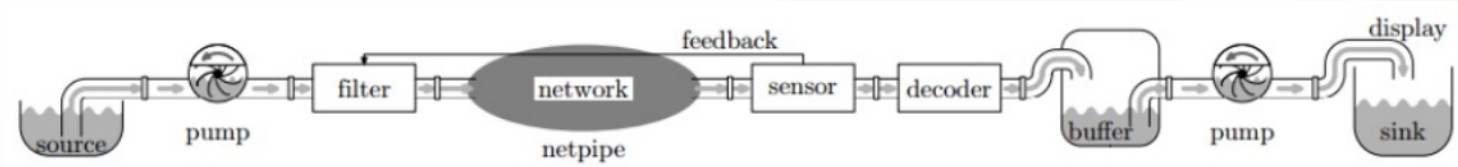
- GStreamer is a **framework** for creating streaming media applications
- The framework is based on **plugins** that will provide the various codec and other functionality

Overview

- **Gst-plugins-base:**
  an essential exemplary set of elements
- **Gst-plugins-good:**
  a set of good-quality plug-ins under LGPL
- **Gst-plugins-ugly:**
  a set of good-quality plug-ins that might pose distribution problems
- **Gst-plugins-bad:**
  a set of plug-ins that need more quality

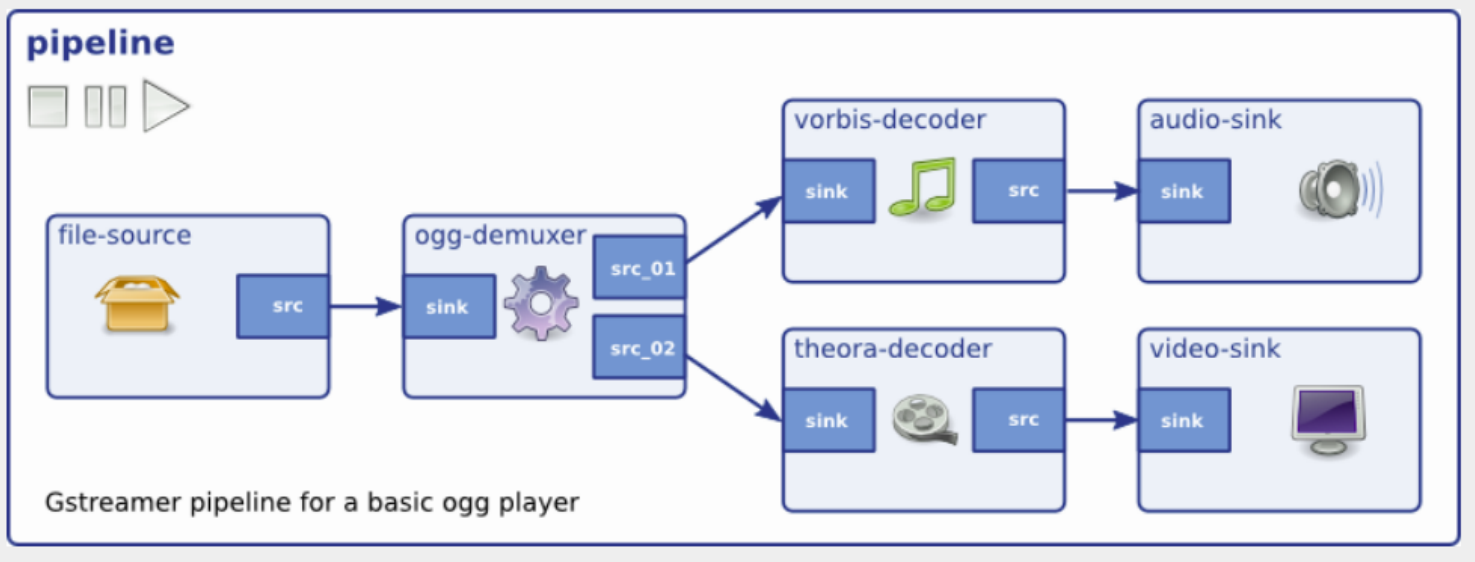
Gstreamer Pipe

A streamer pipe



https://www.slideshare.net/rodrigo_costa/gstreamer-an-overview

Gstreamer Pipe



Gstreamer - Tool

- Gst-inspect-1.0
  - Print supported plug-in
- Gst-launch-1.0
  - Gstreamer tester
- Gst-typefind-1.0
  - Check file for gstreamer plug-in type

Gstreamer - Tool

- Gst-inspect-1.0
  - Check what kind of videosink in Rockpi4b
  - gst-inspect-1.0 | grep -i videosink

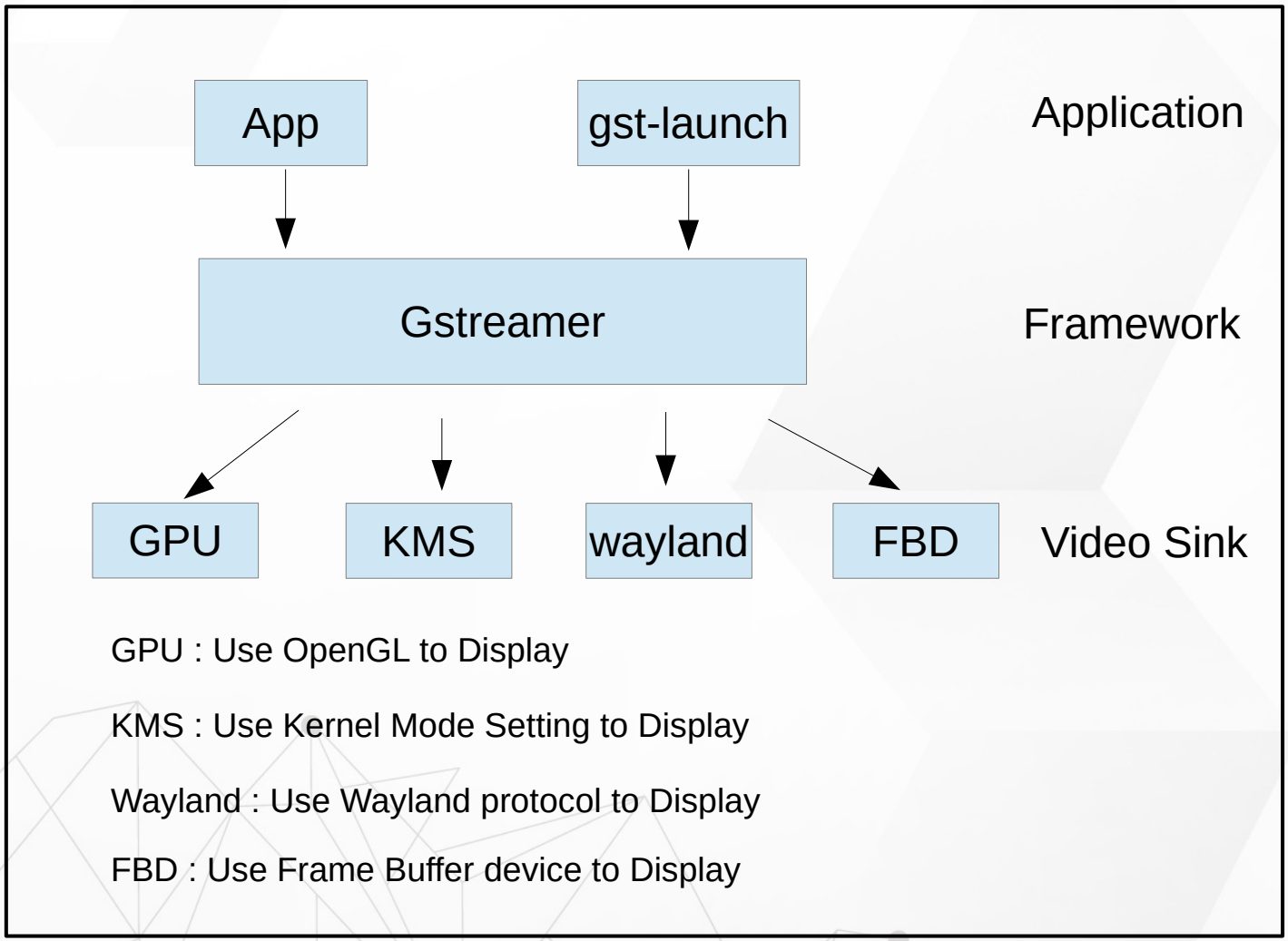
```
rock@rockpi4b:~$ gst-inspect-1.0 | grep -i videosink
debugutilsbad:  fakevideosink: Fake Video Sink
debugutilsbad:  fpsdisplaysink: Measure and show framerate on videosink
inter:  intervideosink: Internal video sink
decklink:  decklinkvideosink: Decklink Video Sink
autodetect:  autovideosink: Auto video sink
rock@rockpi4b:~$
```

Gstreamer - Tool

- Gst-typefind-1.0
  - Check Serenity-DVD-320x240.m4v what kind of file type in gstreamer
  - gst-typefind-1.0 ./Serenity-DVD-320x240.m4v

```
rock@rockpi4b:~$ gst-typefind-1.0 ./Serenity-DVD-320x240.m4v
./Serenity-DVD-320x240.m4v - video/quicktime, variant=(string)iso
```

Gstereamer - Video

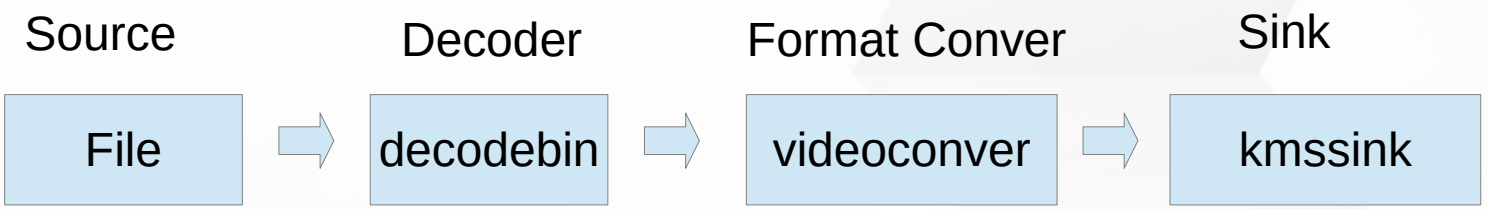


Play Video Test Stream


```
gst-launch-1.0 videotestsrc ! video/x-raw, width=1280, height=720 ! kmssink
```

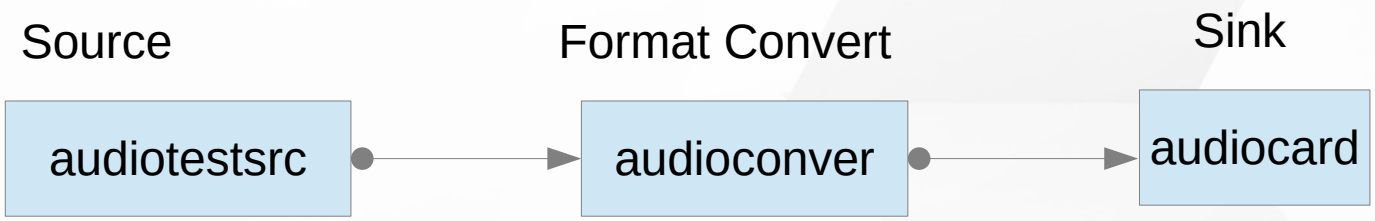
Play a H.264 video

```
gst-launch-1.0 filesrc location=/tmp/Serenity-DVD-320x240.m4v ! \
decodebin name=dec ! \
videoconvert ! \
kmssink
```



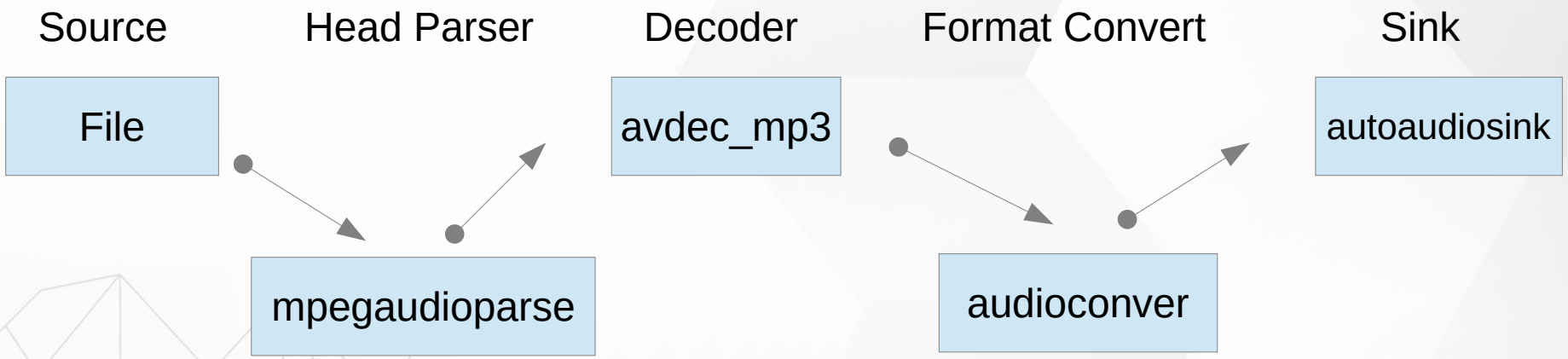
Play a Audio Test

```
gst-launch-1.0 audiotestsrc ! audioconvert ! alsasink device-name=rockchipes8316c
```



Play a MP3

```
gst-launch-1.0 filesrc location="/home/cadtc/audio/piano2-CoolEdit.mp3" ! \
mpegaudioparse ! \
mpg123audiodec ! \
audioconvert ! autoaudiosink
```



ALSA Tool

- ALSA Utile
  - **aplay**
    - Play a WAV file
  - **arecord**
    - Record a sound
  - **alsamixer**
    - A graph tool for adjusting audio gain
  - **amixer**
    - A console tool for adjusting audio gain

ALSA Tool

ALSA Utile

- **aplay**
  - aplay -Dsysdefault:CARD=rockchipes8316c /usr/share/sounds/alsa/Front_Center.wav
  - aplay -Dsysdefault:CARD=HDMICODEC /usr/share/sounds/alsa/Front_Center.wav
- **arecord**
  - arecord -Dhw:0,0 -r 44100 -t wav -f CD -d 5 /tmp/test.wav
- **alsamixer**
  - alasmixer
- **amixer**
  - amixer scontrols | less
  - amixer sget 'HP' 0%
  - amixer sset 'HP' 0%

WiFi and Network

Network Tool

- ifconfig → Network setting
- ping → Network package test
- iperf3 → perform network throughput tests
- dhcpc → used for automatic retrieving of

WIFI

- NetworkManager Command Line Tool
  - nmcli
- Show / manipulate wireless devices and their configuration
  - iw
- For connecting to a WPA/WPA2 network
  - wpa_supplicant

SSH

SSH

- Secure SHell protocol
- SSH Client
- SSH Server

SSH

- SSH Client
  - # sudo apt-get install ssh
  - References

NFS

NFS

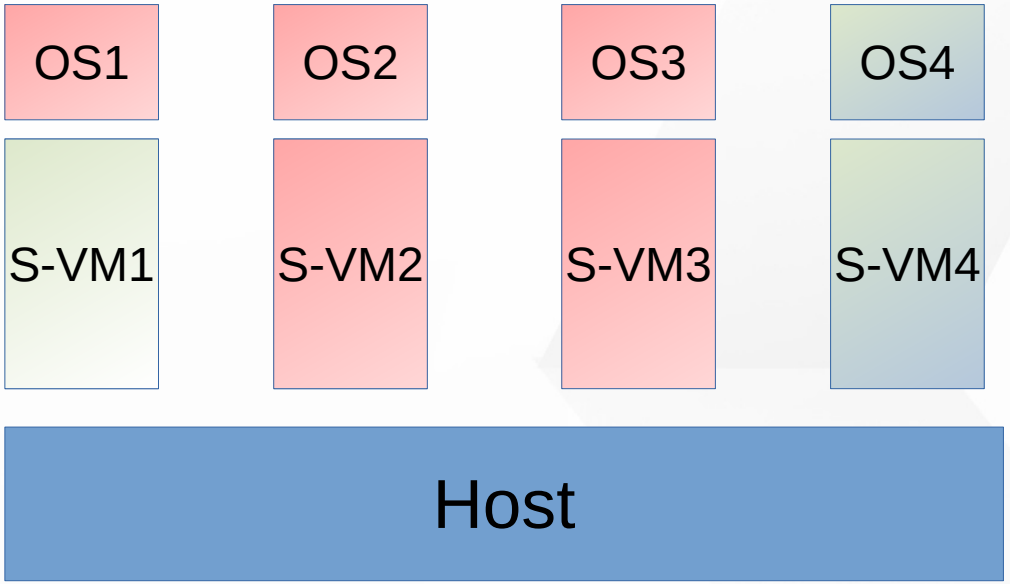
- Network File System
- NFS Client
- NFS Server

Docker

What is Docker

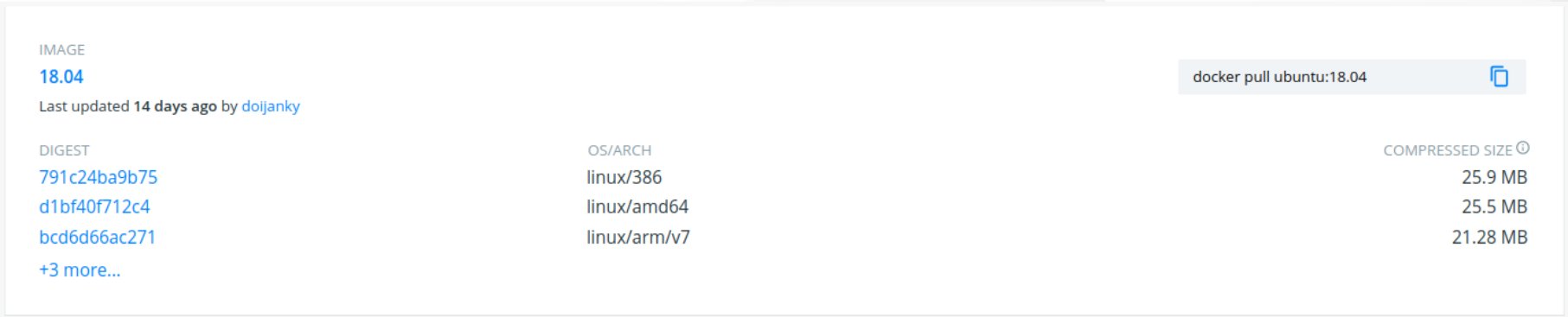
- https://docs.microsoft.com/zh-tw/dotnet/architecture/containerized-lifecycle/what-is-docker
- 簡單的說
  - Docker ： 模擬 作業系統的 “ User space”, 沒有底層
  - VirtualBox ： 模擬 整個 作業系統 , 包含 底層 與 硬體
- 優點
  - 對於開發 User space 的程式方便交換環境 , 快速
- 缺點
  - 無法處理 底層 或是 硬體 相關 部份

What is Docker



Dicker Hub

- https://hub.docker.com/
- https://hub.docker.com/_/ubuntu
- Ubuntu 官方 已在 Docker Hub 建立不同版本的 docker image
- 直接從 Docker Hub 下載需要的新環境
  - 例如 ：Ubuntu 1804



Install a Docker in Ubuntu

- Install docker in Ubuntu
  - $ sudo apt-get install docker
- Docker help
  - $ docker --help

How to use Docker

- Pull a Docker image that in Docker Hub
  - For example : ubuntu 18.04
  - $ docker pull ubuntu:18.04

```
slash@slash-HD631-Q87CRM:~$ dicker images
No command 'dicker' found, did you mean:
 Command 'ticker' from package 'ticker' (universe)
 Command 'docker' from package 'docker.io' (universe)
dicker: command not found
slash@slash-HD631-Q87CRM:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
slash@slash-HD631-Q87CRM:~$ docker pull ubuntu:18.04
18.04: Pulling from library/ubuntu
f08d8e2a3ba1: Extracting [>                                                  ]  294.9kB/26.7MB
3baa9cb2483b: Download complete
94e5ff4c0b15: Download complete
1860925334f9: Download complete
```

How to use Docker-2

- Check docker Image in local
  - $ docker images

```
slash@slash-HD631-Q87CRM:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              18.04               6526a1858e5d        2 weeks ago         64.2MB
slash@slash-HD631-Q87CRM:~$
```

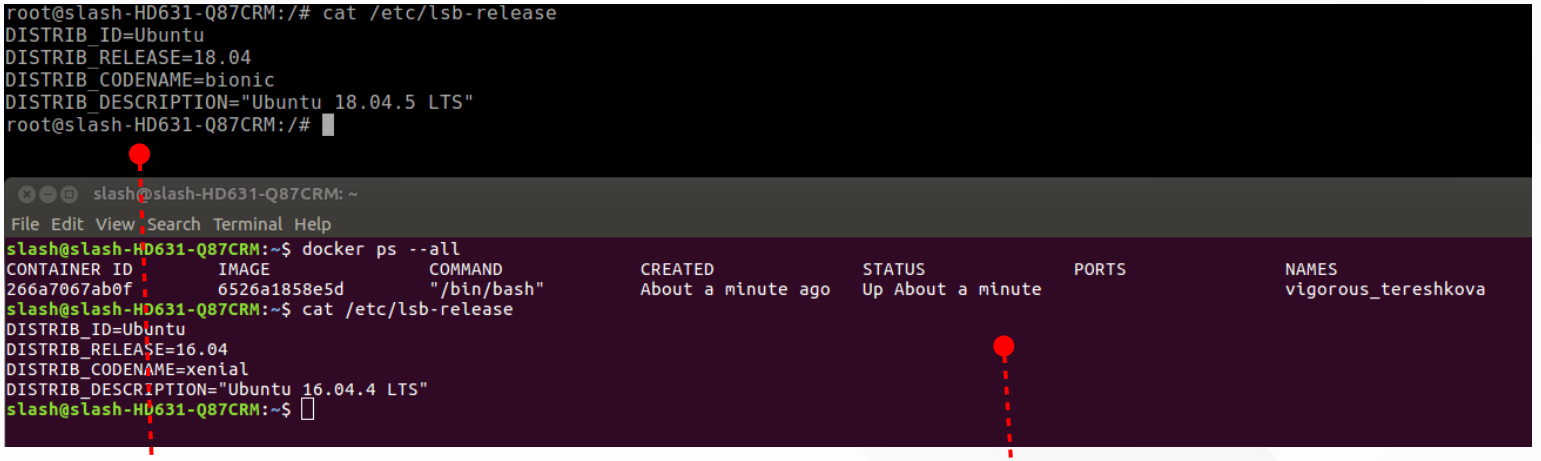
- start to run docker Image in local
  - $ docker run --help
  - $ docker run --net=host -i -t 6526a1858e5d /bin/bash

Enter docker content

```
slash@slash-HD631-Q87CRM:~$ docker run -i -t --net=host 6526a1858e5d /bin/bash
root@slash-HD631-Q87CRM:/#
```

How to use Docker-3

- Check docker content with docker command



Docker content (ubuntu 18.04)

Host (ubuntu 20.04)

How to use Docker-4

- Install package that you want in docker image
  - For example
    - $ apt-get install net-tools

```
root@slash-HD631-Q87CRM:/# apt-get install  net-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  net-tools
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 194 kB of archives.
After this operation, 803 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 net-tools amd64 1.60+git20161116.90da8a0-1ubuntu1 [194 kB]
Fetched 194 kB in 2s (115 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package net-tools.
(Reading database ... 4046 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20161116.90da8a0-1ubuntu1_amd64.deb ...
Unpacking net-tools (1.60+git20161116.90da8a0-1ubuntu1) ...
Setting up net-tools (1.60+git20161116.90da8a0-1ubuntu1) ...
root@slash-HD631-Q87CRM:/#
```

How to use Docker-5

- **Commit** a docker content and push it to Docker Hub
  - 1. check which docker content that you want to commit
    - $ docker ps --all

```
slash@slash-HD631-Q87CRM:~$ docker ps --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
266a7067ab0f        6526a1858e5d        "/bin/bash"         16 hours ago        Up 12 minutes                           vigorous_tereshkova
slash@slash-HD631-Q87CRM:~$
```

  - 2. check REPOSITORY and TAG of docker image current
    -

```
slash@slash-HD631-Q87CRM:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              18.04               6526a1858e5d        2 weeks ago         64.2MB
```

  - 3. commit docker content
    - $ docker commit -m "test" 266a7067ab0f ubuntu:18.04

```
slash@slash-HD631-Q87CRM:~$ docker commit -m "test" 266a7067ab0f ubuntu:18.04
sha256:769ce8b9f38081b4cedf764b51e070ba4032b8146f26cf5c5d6726a3f20b831a
slash@slash-HD631-Q87CRM:~$ rm -rf ./build-fb/^C
slash@slash-HD631-Q87CRM:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              18.04               769ce8b9f380        11 seconds ago      94.9MB
ubuntu              <none>              6526a1858e5d        2 weeks ago         64.2MB
```

How to use Docker-6

- **Import** a Local Docker Image (Load)
  - [CMD] docker load < docker_image_file.tar
- Save a Local Docker Image (Export)
  - [CMD] docker save ${IMAGE_ID} > docker_image_file.tar

How to use Docker-7

- Delete a docker **Content**
  - $ docker ps --all
  - $ docker rm ${Content ID}
- Delete a docker **Image**
  - $ docker images
  - $ docker rmi ${IMAGE ID}

A Sample Docker Usage

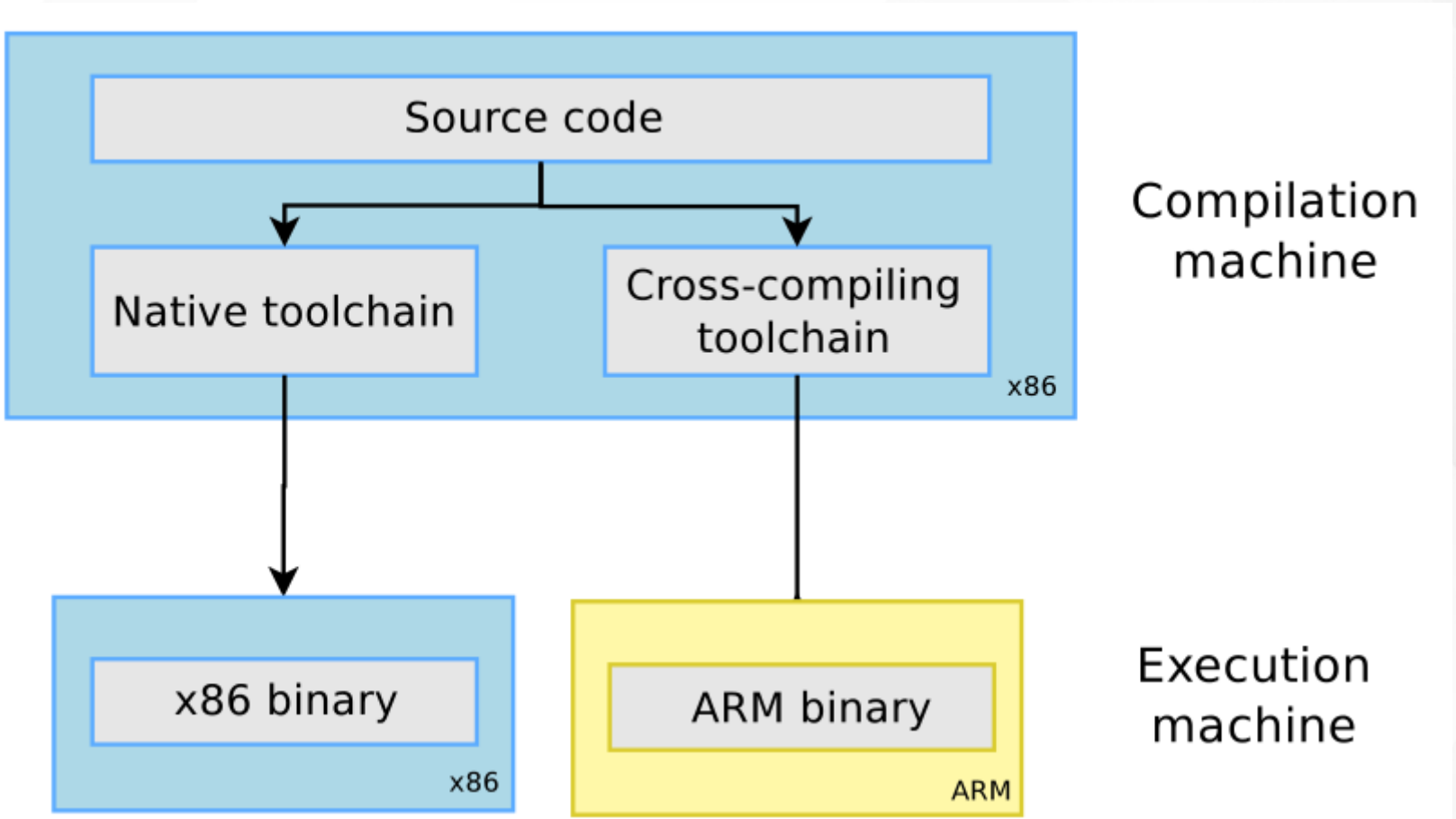
- **docker_info.sh**
- Set environment variables
  - -e --env
  - -e DISPLAY
- Execute application when login to docker content
  - -c --cpu-shares
  - -c "${DOCKER_BUILD_CMD}"
- Work directory
  - -w --workdir
  - -w ${DOCKER_WORKDIR}

A Sample Docker Usage

- Login with user name
  - -u --user
  - -u ${DOCKER_LOGIN_ID}
- Mount a local disk space in docker content
  - -v --volume
  - -v ${LOCAL_FOLDER}:${DOCKER_FOLDER}
- Allocate a pseudo-TTY
  - -t --tty
- Keep STDIN open even if not attached
  - -i --interactive

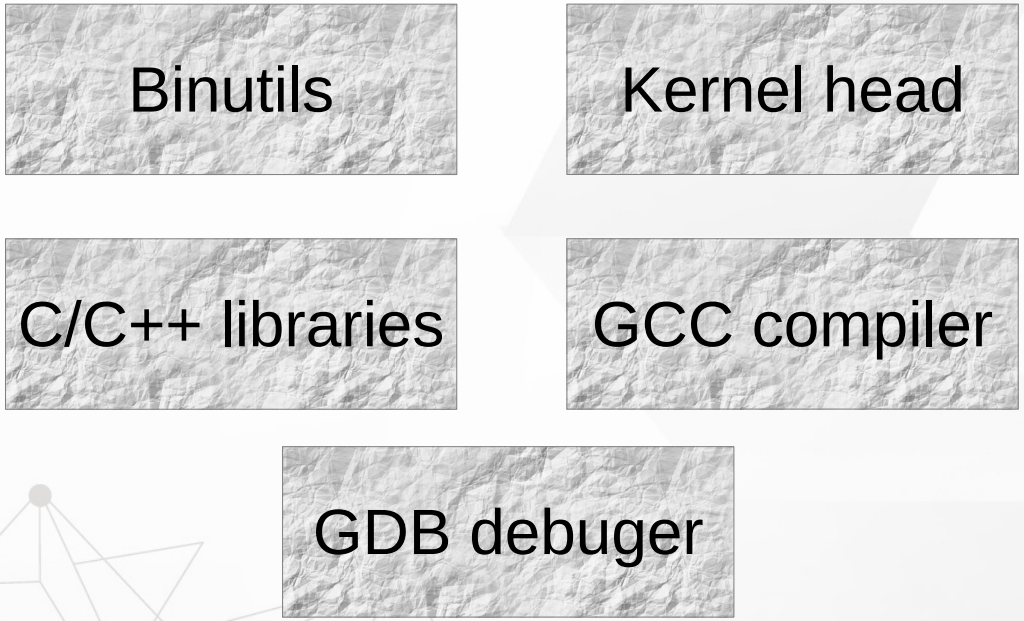
CH5 Cross Compilation Toolchain

Cross Compilation Tool-chain



GCC Components

- The GNU C Compiler
- The GNU Compiler Collection



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 (1)

- 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

GCC flag (2)

- arm-linux-gnueabihf-gcc –help
  - -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
    - -l links with a library file

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.

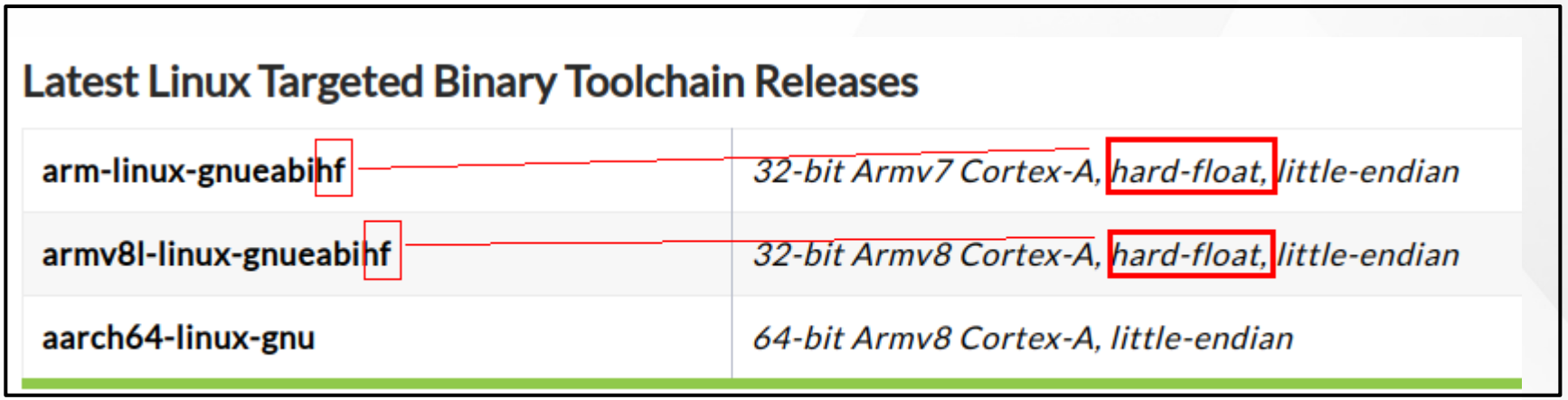
Floating point support

- 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

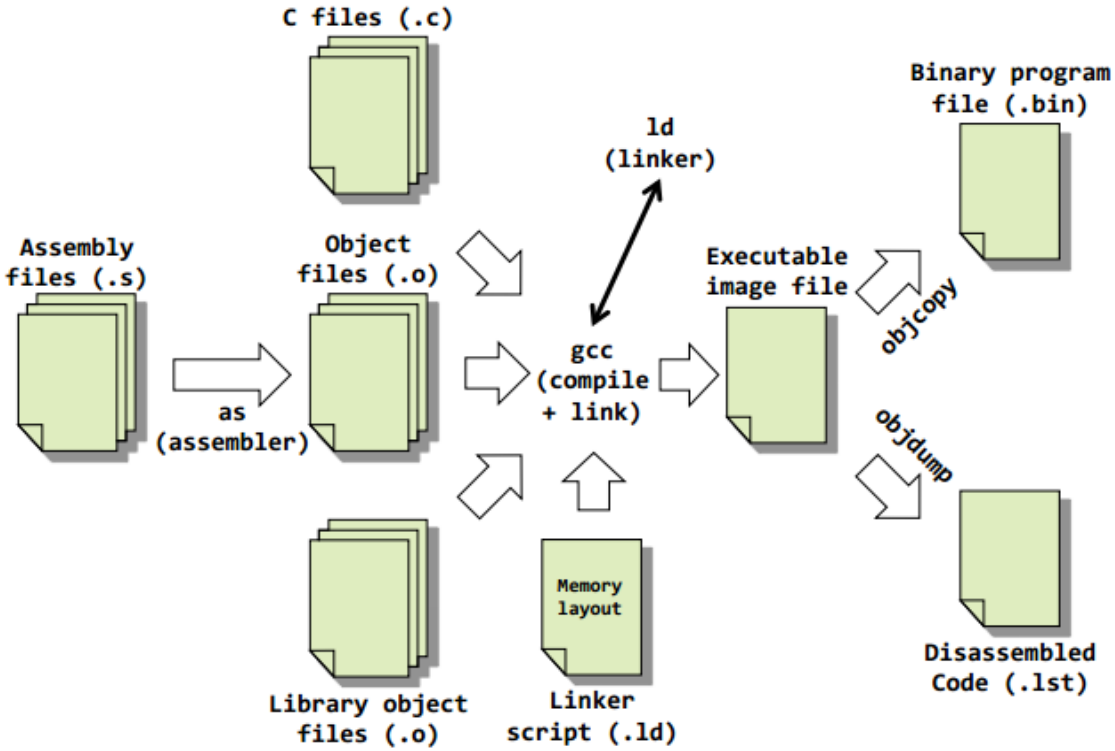
- Building a cross-compiling toolchain by ourself
  - Crosstool-NG
  - http://crosstool-ng.org/#introduction
- Pre-build toolchain
  - Linaro - https://www.linaro.org/downloads/
  - By Linux distribution -
    - sudo apt-get install gcc-arm-linux-gnueabi
  - BSP
  - CodeSourcery

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
```