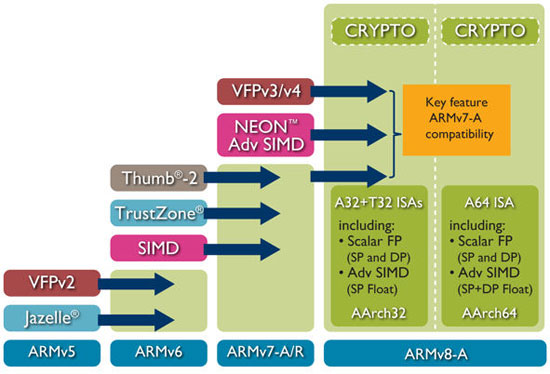
ARMv8

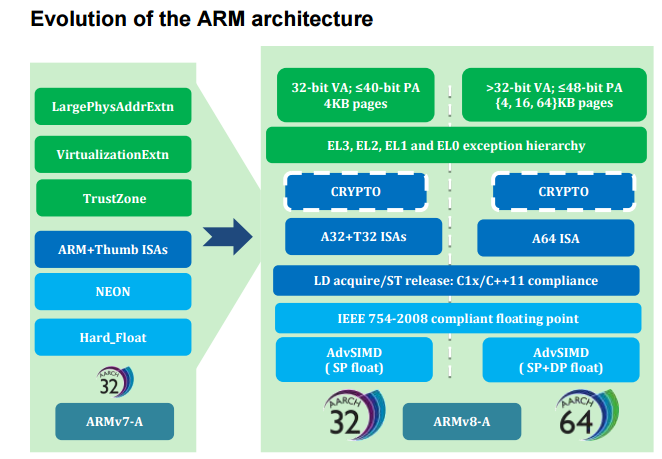
# 1 ARMv8架构介绍

1.1 ARMv8 概述

ARMv8（当前只有A系列，即ARMv8-A）架构，是ARM公司满足新需求而设计的一个架构，是近20年来，ARM架构变动最大的一次。它新引入了Execution State、Exception Level、Security State等新特性，已经和我们对旧的ARM架构的认知，有很大差距了。



ARMv8架构有一个重要的特点就是与之前的架构兼容。



## 1.1 执行态

1. AArch64: 64位执行态
   1. 提供31组64位通用寄存器
   2. 提供64位PC寄存器、SP寄存器与ELR寄存器
   3. 4种特权模式（EL0-EL3）
   4. 支持64位虚拟地址
   5. 定义一组PSTATE来保存PE状态
   6. 没有协处理器的概念
2. Aarch32: 32位执行态
   1. 提供16组32位的通用寄存器
   2. 提供一组ELR寄存器，作为从超级模式（hyp-mode）的exception返回用
   3. 提供A32（兼容ARMv7 ARM）与T32（兼容ARMv7 Thumb）两种指令集
   4. 使用32位虚拟地址
   5. 使用单一的CPSR来保存PE状态
   6. Aarch32只支持CP10，CP11，CP14和CP15

在A32与T32中做切换只需通过BX即可，但要在Aarch32与Aarch64之间做切换只能通过Exception。

# 2 AArch64指令集介绍

ARMv8-A\_Architecture\_Reference\_Manual P.111

# 3 系统级架构

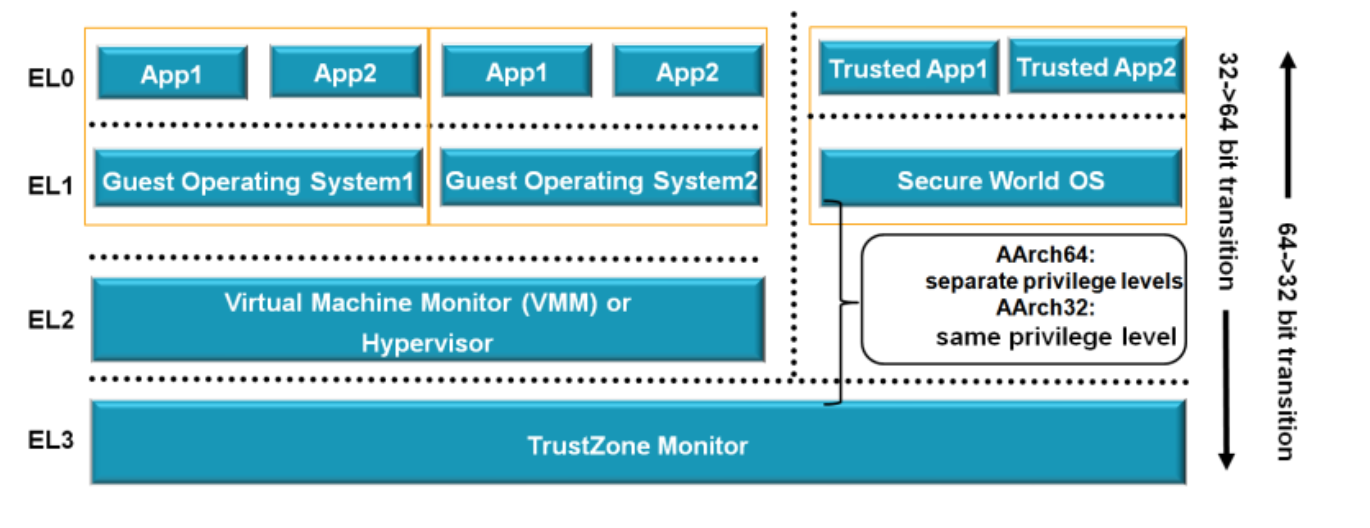
ARMv8-A\_Architecture\_Reference\_Manual P.1405

## 3.1 异常级（Exception Levels）

ARMv8-A\_Architecture\_Reference\_Manual P.1408

* ARMv8-A架构定义了4个异常级别，分别为EL0到EL3。其中数字越大，代表特权等级越高。
* EL0: 无特权模式(unprivileged)
* EL1: 作業系統核心模式(OS kernel mode)
* EL2: 虛擬機器監視器模式(Hypervisor mode)
* EL3: TrustZone® monitor mode
* 要提升到較高層級需要透過exceptions(如: 中斷、page faults等)。
* EL0 => EL1: SVC (system call)
* EL1 => EL2: HVC (hypervisor call)
* EL2 => EL3: SMC (secure monitor call)
* 在轉換時會將返回位址(return address)紀錄在例外連結暫存器ELR(Exception-Link-Register)。
* 每個EL會有個別的SP(stack pointer)
* 根據目前架構，由下層系統的Execution State決定上層系統所在模式
* 若下層系統為32bits則上層只能為32bits，反之若為64bits則上層可為32bits or 64bits

参考下图：



## 3.2 安全状态（Security state）

ARMv8-A架構提供兩種安全性狀態，他們有個別的實體記憶體定址空間(Secure physical address space)。

* 安全狀態(Secure state): PE可以存取安全及不安全的實體定址空間，有EL0.EL1.EL3
* 不安全狀態(Non-Secure state): 只能存取不安全的實體定址空間，有EL0.EL1.EL2

Extension.png (from: Hardware Support Virtualization (國立清華大學))

3.3 虚拟化

這邊提到的虛擬化為有實現EL2架構的系統。以下為其基礎模型: (manual D1.5 Virtualization)

* 一個跑在EL2的Hypervisor負責切換跑在EL1、EL0的virtual machines
* 一些跑在virtual machines上(在EL1中)的Guest OS
* 每個Guest OS上跑在EL0的應用程式

每個VM會被Hypervisor指定一個VMID。

EL2只會實現在 Non-secure state，並負責:

* 提供虛擬值給少數特定的暫存器(1)。Guset OS 或其上的應用程式讀取這些暫存器時會得到虛擬的值。
* Trapping: 當在做記憶體管理及存取其他大多數的暫存器((1)之外的)時會產生exception並由EL2處理。
* Routing interrupt: 將中斷分配給
* 現在的Guest OS
* 現在沒在執行的Guest OS
* hypervisor

(以上會在個別的章節特別探討)

實現EL2包含以下實作:

* Hypervisor Call (HVC) exception
* Traps to EL2
* 虚拟中断，包括:
  + Virtual SError
  + Virtual IRQ
  + Virtual FIQ
  + 所有虛擬中斷會由EL1處理
  + 每個虛擬中斷可由EL2個別啟用
  + 每個虛擬中斷都會有其對應的實體中斷
  + 當一個虛擬中斷被啟用時，其對應的實體中斷會由EL2處理(除非EL3指定他要處理)

# 4 虚拟内存系统架构

# 5 通用定时器

# 参考

1．[ARMv8](http://wiki.csie.ncku.edu.tw/embedded/ARMv8) ：台湾一位大学教师写的文章

2. [蜗窝科技](http://www.wowotech.net/sort/armv8a_arch)之系列文章