# Chapter A1 Arm 架构综述

1.Arm 架构是一种 load-store 架构。
Arm 处理器是典型的 RISC 处理器，只有 load 和 store 指令才能访问内存。数据处理指令仅对寄存器内容进行操作。
Armv8 是 Armv7 之后的下一个主要架构更新。它引入了 64 位体系结构，但保持了与现有 32 位体系结构的兼容性。它使用两种执行状态：
（1）AArch32
在 AArch32 状态下，代码可以访问 32 位通用寄存器。在 AArch32 状态下执行的代码只能使用 A32 和 T32 指令集。这种状态与 Armv7 - A 架构广泛兼容。
（2）AArch64
在 AArch64 状态下，代码可以访问 64 位通用寄存器。AArch64 状态只存在于 Armv8 架构中。在 AArch64 状态下执行的代码只能使用 A64 指令集。

2.Armv8 引入了一组新的 32 位指令，称为 A64，每条指令 32 位，以及新的编码和汇编语言。A64 仅在处理器处于 AArch64 状态时可用。它提供了与 A32 和 T32 指令集类似的功能，但**提供了对更大的虚拟地址空间的访问**，并进行了一些其他更改，包括**减少了条件约束**。
Armv8 还定义了一个可选的**加密扩展**。这个扩展提供了 A64 指令集中的加密和哈希指令。

3.执行 A64 指令的处理器处于 AArch64 状态。在这种状态下，指令可以**同时访问 64 位和 32 位寄存器**。
执行 A32 或 T32 指令的处理器处于 AArch32 状态。在这种状态下，指令**只能访问 32 位寄存器**，而不能访问 64 位寄存器。
基于 Armv8 架构的处理器可以运行为 AArch32 和 AArch64 状态构建的应用程序，但是**AArch32 和 AArch64 状态之间的变化只能在异常边界发生**。
Arm 编译器工具链为 AArch32 状态或 AArch64 状态构建映像。因此，使用 Arm 编译器工具链构建的映像**可以只包含 A32 和 T32 指令，也可以只包含 A64 指令**。
**处理器只能执行指令集中与其当前执行状态相匹配的指令**。AArch32 状态的处理器不能执行 A64 指令，AArch64 状态的处理器不能执行 A32 或 T32 指令。

4.**Advanced SIMD** 是一种 **64 位和 128 位**混合单指令多数据(**SIMD**)技术，它针对高级媒体和信号处理应用程序和嵌入式处理器。
Advanced SIMD 是作为基于 **arm 的处理器的一部分**实现的，但是它**有自己的执行流水线和与通用寄存器组不同的寄存器组**。
Advanced SIMD 指令在 A32 和 A64 都可用。A64 Advanced SIMD 的指令基于 A32 的指令，但略有不同：
（1）A64 有 32 个 128 位向量寄存器，而 A32 只有 16 个。
（2）不同的寄存器包装方案：在 A64 中，较小的寄存器占用较大寄存器的低阶位。例如，S31 映射到 D31 的位[31:0]。在 A32 中，较小的寄存器被打包成较大的寄存器。例如，S31 映射到 D15 的位[63:32]。
（3）A64 Advanced SIMD 指令支持单精度和双精度浮点数据类型和算术。A32 Advanced SIMD 指令只支持单精度浮点数据类型。

浮点硬件的寄存器组不同于 Arm 核心寄存器组，但浮点寄存器组与 SIMD 寄存器组共享。在 AArch32 状态下，与 VFPv4 相比，浮点支持基本没有变化，除了为了符合 IEEE 754 标准而增加了一些指令。

AArch64 状态下的浮点架构也是基于 VFPv4 的。主要区别如下：

（1） 同前面的（1）。

（2） 同前面的（2）。

（3） 浮点硬件的存在是强制的，不支持软件实现的浮点链接。

（4） AArch64 状态不支持早期版本的浮点架构，例如 VFPv2, VFPv3 和 VFPv4。

（5） 在 AArch32 和 AArch64 状态下都不支持 VFP 矢量模式，而是使用矢量浮点的 Advanced SIMD 指令。

（6） 增加了一些新的指令：①半精度和双精度直接转换 ②Load and store pair, replacing load and store multiple. ③融合了乘-加和乘-减 ④IEEE 754-2008 兼容性指令

# Chapter A3 AArch64 状态综述

1. AArch64 状态中的寄存器

Arm 处理器提供通用和专用寄存器。一些附加寄存器在 privileged 执行模式下可用。

在 AArch64 状态，有以下寄存器可用：

（1）31 个 64 位通用寄存器 X0-X30，其下半部分可访问为 W0-W30。

（2）四个栈指针寄存器 SP_EL0、SP_EL1、SP_EL2、SP_EL3。

（3）三个异常链接寄存器(exception link registers) ELR_EL1、ELR_EL2、ELR_EL3。

（3）三个保存的程序状态寄存器(saved program status registers) SPSR_EL1、SPSR_EL2、SPSR_EL3。

（3）一个程序计数器。

除了 SPSR_EL1、SPSR_EL2 和 SPSR_EL3 是 32 位之外，其他寄存器都是 64 位的。

大多数 A64 整数指令可以在 32 位或 64 位寄存器上操作。**W 表示 32 位，X 表示 64 位**。n 在 0-30 范围内的名称 Wn 和 Xn 指的是同一个寄存器。当使用 32 位形式的指令时，源寄存器的上 32 位将被忽略，而目标寄存器的上 32 位将被设置为零。

**没有名为 W31 或 X31 的寄存器**。根据指令的不同，寄存器 31 要么是**栈指针**，要么是**零寄存器**。当用作栈指针时，它指的是 SP，当用作零寄存器时，它指的是 32 位上下文中的 WZR 或 64 位上下文中的 XZR。

2. Exception levels

Armv8 架构定义了**四个异常级别**，从 EL0 到 EL3，其中 EL3 是最高的异常级别，拥有最大的执行权。在获取异常时，异常级别可以增加也可以保持不变，而在从异常返回时，异常级别可以减少也可以保持不变。

EL0: Applications.

EL1: OS kernels and associated functions that are typically described as privileged.

EL2: Hypervisor(管理程序).

EL3: Secure monitor.

当将异常提升到更高的异常级别时，执行状态可以保持不变，也可以从 AArch32 更改为 AArch64。当返回到较低的异常级别时，执行状态可以保持不变，也可以从 AArch64 更改为 AArch32。改变执行状态的唯一方法是从异常中获取或返回。

**在启动和重置时，处理器进入实现的最高异常级别**。此异常级别的**执行状态**是实现的属性，**可能由配置输入信号决定**。

对于 EL0 以外的异常级别，执行状态由一个或多个控制寄存器配置位决定。这些位只能在更高的异常级别上设置。对于 EL0，执行状态是作为返回到 EL0 的异常的一部分确定的，由执行返回的异常级别控制。

3. Link registers

当进行子例程调用时，**链接寄存器(LR)存储返回地址**。如果返回地址存储在堆栈上，它还可以用作通用寄存器。LR 映射到寄存器 30。

有三个异常链接寄存器 ELR_EL1、ELR_EL2 和 ELR_EL3，它们对应于每个异常级别。当出现异常时，目标异常级别的异常链接寄存器存储异常处理完成后要跳转到的返回地址。如果异常是从 AArch32 状态获取的，那么 ELR 中的前 32 位都被设置为零。**异常级别内的子例程调用使用 LR 存储子例程的返回地址**。

在一个异常级别上，如果启用了使用相同异常级别的中断，则必须确保**将 ELR 存储在堆栈上**，因为在执行中断时，它将被一个新的返回地址覆盖。

4. Stack Pointer register

SP 表示 64 位栈指针，SP_EL0 是 SP 的别名，不要将 SP 用作通用寄存器。只能在以下指令中使用 SP 作为操作数：

（1）作为加载和存储的基本寄存器。在这种情况下，在添加任何偏移量之前，必须对它进行四字对齐，否则会出现栈对齐异常。

（2）作为算术指令的源或目的地，但不能在设置条件标志的指令中用作目的地。

（3）在逻辑指令中，例如为了对齐。

对于 SP_EL1、SP_EL2 和 SP_EL3 这三个异常级别，每个都有一个单独的栈指针。**在异常级别中，您可以为该异常级别使用专用的栈指针，也可以使用 SP_EL0**。您可以使用 **SPSel** 寄存器来选择在异常级别中使用哪个栈指针。**栈指针的选择由附加到异常级别名称的字母 t 或 h 表示**，例如 EL0t 或 EL3h。t 后缀表示异常级别使用 SP_EL0，h 后缀表示异常级别使用 SP_ELx，其中 x 是当前异常级别号。EL0 总是使用 SP_EL0，因此不能有 h 后缀。

5. 在 AArch64 状态中预先声明的核心寄存器名称

| 寄存器名称 | 含义 |
|---|---|
| **W0-W30** | 32 位通用寄存器。 |
| **X0-X30** | 64 位通用寄存器。 |
| **WZR** | 32 位 RAZ/WI 寄存器。这是寄存器 31 在 32 位上下文中用作零寄存器时的名称。 |
| **XZR** | 64 位 RAZ/WI 寄存器。这是寄存器 31 在 64 位上下文中用作零寄存器时的名称。 |
| **WSP** | 32 位栈指针。这是寄存器 31 在 32 位上下文中作为栈指针时的名称。 |
| **SP** | 64 位栈指针。这是寄存器 31 在 64 位上下文中用作栈指针时的名称。 |
| **LR** | 链接寄存器。这是 X30 的同义词。 |

6. 在 AArch64 状态中预先声明的扩展寄存器名称

| 寄存器名称 | 含义 |
|---|---|
| **V0-V31** | 128 位 Advanced SIMD 向量寄存器。 |
| **Q0-Q31** | 128 位标量的 Advanced SIMD 寄存器。 |
| **D0-D31** | 64 位标量的 Advanced SIMD 寄存器，浮点双精度寄存器。 |
| **S0-S31** | 32 位标量的 Advanced SIMD 寄存器，浮点单精度寄存器。 |
| **H0-H31** | 16 位标量的 Advanced SIMD 寄存器，浮点半精度寄存器。 |
| **B0-B31** | 8 位标量的 Advanced SIMD 寄存器。 |

7. 程序计数器

程序计数器(PC)包含当前执行指令的地址。它按所执行指令的大小递增，通常为四个字节。**PC** 不是一个通用寄存器，**不能显式地访问**它。以下类型的指令隐式地读取它：

（1）计算 PC-relative 地址的指令。

（2）PC-relative literal loads.

（3）Direct branches to a PC-relative label.

（4）Branch and link instructions, which store it in the procedure link register.

能往 PC 写数据的指令只有：（1）条件分支和无条件分支。（2）异常生成和异常返回。

分支指令将目的地址加载到 PC 中。

8. AArch64 状态中的条件执行

在 AArch64 状态中，NZCV 寄存器保存 **N、Z、C 和 V 条件标志的副本**。处理器使用它们来决定是否执行条件指令。NZCV 寄存器以位的形式包含标志[31:28]。

使用 **MSR 和 MRS 指令**，可以在所有异常级别设置和访问条件标志。

在 A64 中只有很少的指令可以设置或测试条件标志，唯一有条件执行的指令是条**件分支 B.cond**，如果条件为 false，则表现为 NOP。

9. The Q flag in AArch64 state

在 AArch64 状态下，不能读或写 Q 标志，因为在 **A64 中没有操作在通用寄存器上的饱和算术指令**（饱和运算，就是当运算结果大于一个上限或小于一个下限时，结果就等于上限或是下限）。

**Advanced SIMD 饱和算术指令在浮点状态寄存器(FPSR)中设置 QC 位来表示饱和已经发生**。可以通过 Q 助记符修饰符标识这些指令，例如 SQADD。

10. Process State

进程状态字段包括：（1）N，Z，C 和 V 条件标志(NZCV)。（2）当前寄存器宽度(nRW)。（3）栈指针选择位(SPSel)。（4）中断禁用标志(DAIF)。（5）当前异常级别(EL)。（6）单步进程状态位(SS)。（7）非法异常返回状态位(IL)。

可以用 **MSR 写**：（1）NZCV 寄存器中的 N、Z、C 和 V 标志。（2）中断在 DAIF 寄存器中禁用标志。（3）在 EL1 或更高层级，SPSel 寄存器中的 SP 选择位。

可以用 **MRS 读**:（1）NZCV 寄存器中的 N、Z、C 和 V 标志。（2）中断在 DAIF 寄存器中禁用标志。（3）当前寄存器（EL1 或更高层级）中的异常级别位。（4）在 EL1 或更高层级，SPSel 寄存器中的 SP 选择位。

当异常发生时，**与当前异常级别关联的所有进程状态字段都存储在与目标异常级别 SPSR 关联的单个寄存器中**。只能从 SPSR 访问 SS、IL 和 nRW 位。

11. Saved Program Status Registers (SPSR) in AArch64 state

**保存的程序状态寄存器(SPSRs)是 32 位寄存器**，当一个异常被带到使用 AArch64 状态的异常级别时，它**存储当前异常级别的进程状态**。这**允许在处理异常后恢复进程状态**。

在 AArch64 状态下，每个目标异常级别都有自己的 SPSR：SPSR_EL1、SPSR_EL2、SPSR_EL3。SPSRs 存储以下信息：（1）NZCV 标记。（2）中断禁用标志(DAIF)。（3）寄存器宽度。（4）执行模式。（5）IL 和 SS 位。

12. A64 指令集概述

| 指令组 | 描述 |
| --- | --- |
| **分支和控制** | （1）分支到子程序并从子程序返回。（2）向后分支，形成循环。（3）在条件结构中向前分支。（4）从异常中生成和返回。 |
| **数据处理** | 这些指令在通用寄存器上操作。它们可以对两个寄存器的内容执行加法、减法或位逻辑等操作，并将结果放在第三个寄存器中。它们还可以对单个寄存器中的值进行操作，或者对寄存器中的值和指令 |

| | |
|---|---|
| | 中提供的立即数进行操作。<br>A64 包括有符号和无符号 32 位和 64 位的乘法和除法指令。 |
| **寄存器 LOAD 和 STORE** | 这些指令从存储器加载或存储单个寄存器或一对寄存器的值。可以加载或存储一个 64 位双字、32 位字、16 位半字或 8 位字节，或者**一对字或双字**。字节和半字负载可以是符号扩展的，也可以是零扩展的，以填充 32 位寄存器。<br>还可以将有符号字节、半字或字加载到 64 位寄存器中并进行符号扩展，或者将一对有符号的字加载到两个 64 位寄存器中。 |
| **系统寄存器访问** | 这些指令将系统寄存器的内容移动到通用寄存器或从通用寄存器移动到通用寄存器。 |