目录

- 前言 1.1
- ARM概述 1.2
  - 调用规范 1.2.1
- ARM寄存器 1.3
  - 寄存器命名 1.3.1
  - 寄存器架构 1.3.2
  - 常用寄存器 1.3.3
    - PC 1.3.3.1
    - IP 1.3.3.2
    - SP 1.3.3.3
    - LR 1.3.3.4
    - 状态寄存器 1.3.3.5
      - CPSR 1.3.3.5.1
  - 其他寄存器 1.3.4
  - AArch64=ARM64 1.3.5
    - 特殊用途寄存器 1.3.5.1
      - NZCV 1.3.5.1.1
- ARM汇编指令 1.4
  - 指令列表 1.4.1
    - 按字母 1.4.1.1
    - 操作码 1.4.1.2
  - 常用汇编指令 1.4.2
    - 赋值 1.4.2.1
    - 内存操作 1.4.2.2
    - 比较 1.4.2.3
    - 分支跳转 1.4.2.4
    - 条件选择 1.4.2.5
    - 寻址 1.4.2.6
    - 算数运算 1.4.2.7
    - 逻辑运算 1.4.2.8
    - SVC系统调用 1.4.2.9
    - 其他 1.4.2.10
- ARM常见用法和通用规则 1.5
  - 函数调用 1.5.1

跳转指令 1.5.2
条件执行 1.5.3
cond条件码 1.5.3.1
Pre-index和Post-index 1.5.4
flexible second operand 1.5.5
工具 1.6
指令和二进制opcode互转 1.6.1
附录 1.7
X86汇编 1.7.1
参考资料 1.7.2

最流行汇编语言：ARM

- 最新版本：`v0.8.0`
- 更新时间：`20240925`

简介

整理最流行的汇编语言ARM的相关内容。包括先概述，以及调用规范；ARM寄存器，包括寄存器架构、常用寄存器，比如IP和状态寄存器，以及其他寄存器；再解释ARM汇编指令，包括ARM汇编指令列表，包括按字母表顺序排列的、带操作码的；以及各种常用汇编指令，包括赋值、内存操作、比较、分支跳转、条件选择、寻址、算数运算、逻辑运算、SVC系统调用以及其他；继续整理了ARM常用用法和通用规则，包括函数调用、跳转指令、条件执行、Pre-Index和Post-Index、fleiable second oprand；最后附录上X86汇编。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

HonKit源码

- crifan/popular_assembly_arm: 最流行汇编语言：ARM

如何使用此HonKit源码去生成发布为电子书

详见：crifan/honkit_template: demo how to use crifan honkit template and demo

在线浏览

- 最流行汇编语言：ARM book.crifan.org
- 最流行汇编语言：ARM crifan.github.io

离线下载阅读

- 最流行汇编语言：ARM PDF
- 最流行汇编语言：ARM ePub
- 最流行汇编语言：ARM Mobi

版权和用途说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

其他

作者的其他电子书

本人 `crifan` 还写了其他 `150+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

关于作者

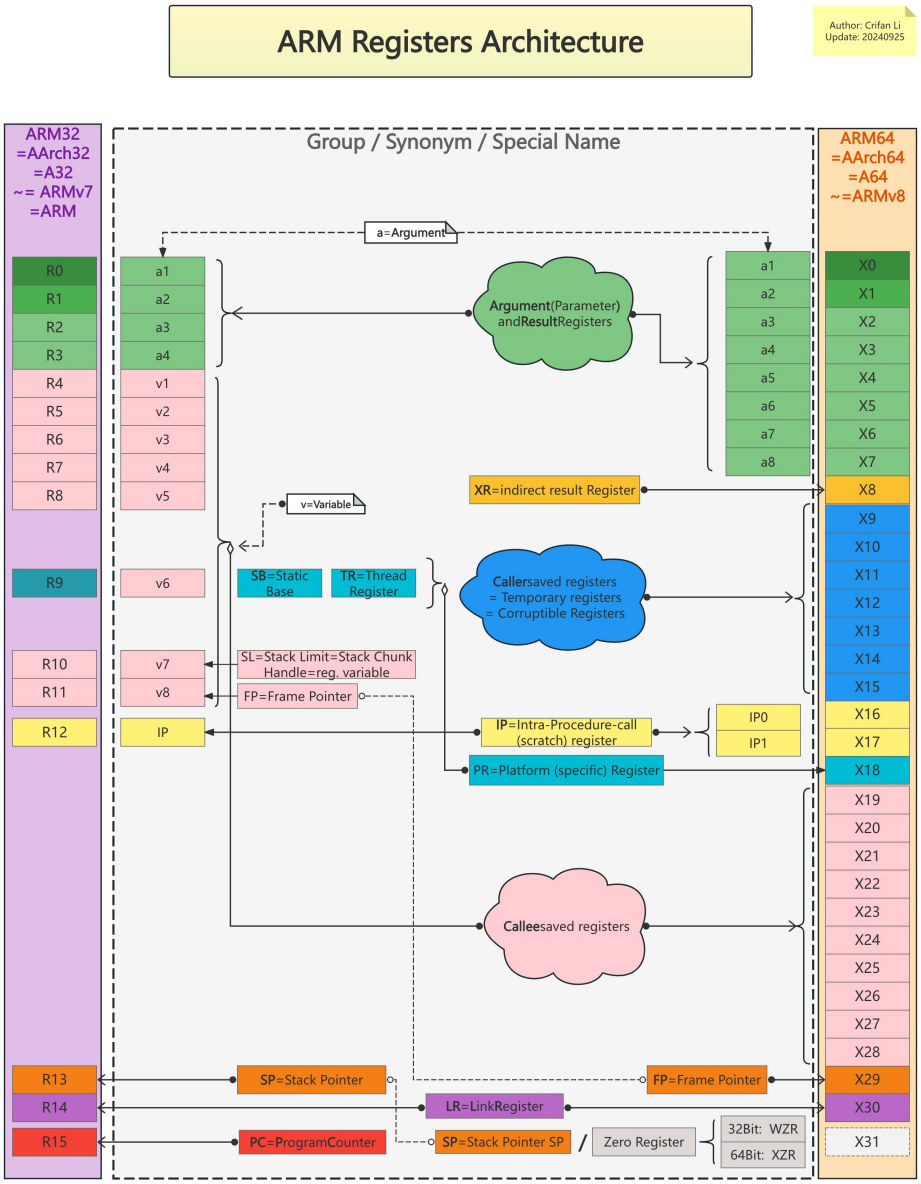
关于作者更多介绍，详见：

关于CrifanLi李茂 – 在路上

ARM概述

- 【整理】ARM汇编指令和架构
- 【整理】ARM架构：栈Stack
- ARMv8-A has a 64-bit architecture called AArch64

- ARM
  - 指令集
    - ARM本身有不同的：指令集
      - ARM32：AArch32 = 32位 的 ARM的A32 和 Thumb的T32
      - ARM64：AArch64 = 64位 的 A64指令集
    - 指令集会使用到寄存器，而寄存器的使用，要符合：`调用规范`
  - 调用规范
    - 包括
      - ARM32：AAPCS
      - ARM64：AAPCS64
    - 含义
      - PCS=Procedure Call Standard
        - specifies
          - Which registers are used to pass arguments into the function.
          - Which registers are used to return a value to the function doing the calling, known as the caller.
          - Which registers the function being called, which is known as the callee, can corrupt.
          - Which registers the callee cannot corrupt.
    - 概述
      - 自己整理的：
        - 本地查看

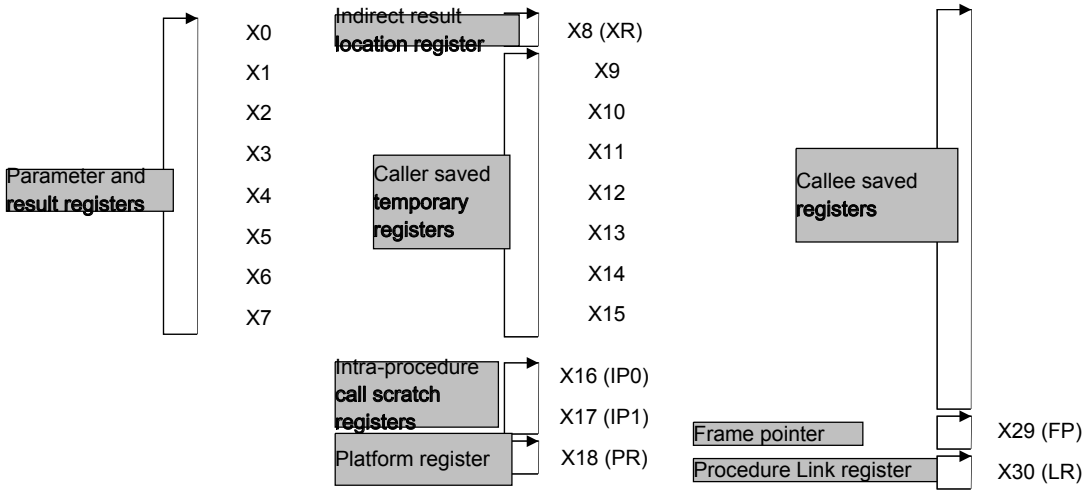


- 在线浏览
  - ARM寄存器架构 | ProcessOn免费在线作图
- AAPCS

Register	Synonym	Special	Role in the procedure call standard
r15		PC	The Program Counter.
r14		LR	The Link Register.
r13		SP	The Stack Pointer.
r12		IP	The Intra-Procedure-call scratch register.
r11	v8		Variable-register 8.
r10	v7		Variable-register 7.
r9		v6 SB TR	Platform register. The meaning of this register is defined by the platform standard.
r8	v5		Variable-register 5.
r7	v4		Variable register 4.
r6	v3		Variable register 3.
r5	v2		Variable register 2.
r4	v1		Variable register 1.
r3	a4		Argument / scratch register 4.
r2	a3		Argument / scratch register 3.
r1	a2		Argument / result / scratch register 2.
r0	a1		Argument / result / scratch register 1.

Table 2, Core registers and AAPCS usage

- AAPCS64 ~= AArch64

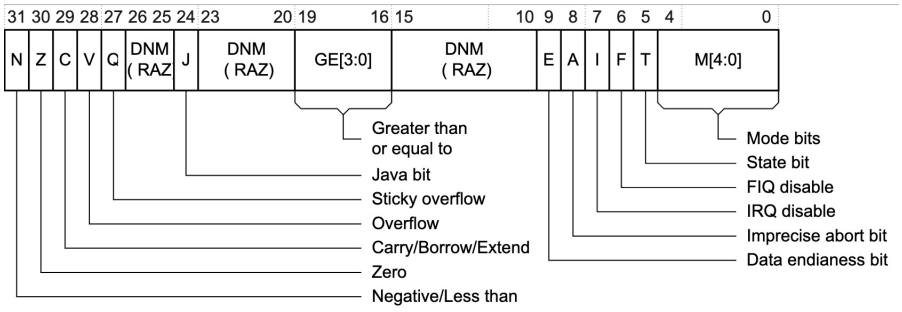


- 详解
  - 特殊的寄存器，有专门的名称
    - `PC` = `Program Counter` = `程序计数器`
      - 用途：记录当前CPU的地址，总是指向正在“取指”的指令
        - 流水线（一般）使用三个阶段，因此指令分为三个阶段执行
          - 1、取指：从存储器装载一条指令
          - 2、译码：识别将要被执行的指令
          - 3、执行：处理 指令并将结果写回寄存器
      - 对应寄存器
        - ARM32：`R15`
        - ARM64：`X31`
    - `LR` = `Link Register` = `链接寄存器`
      - 用途：用于函数调用function calls
      - 对应寄存器

        - ARM32：`R14`
        - ARM64：`X30`
    - `SP` = `Stack Pointer` = 栈指针
        - ARM32：`R13`
    - `FP` = `Frame Pointer` = 帧指针
        - ARM32：R11？
        - ARM64：`X29`
    - `IP` == `Intra-Procedure` = `Intra-Procedure-call (scratch) register`
        - ARM32：
            - R12
        - ARM64
            - IP0 (X16) = first intra-procedure-call scratch register
            - IP1 (X17) = second intra-procedure-call temporary register
    - PR=Platform (specific) Register=平台相关寄存器
        - ARM32：R9
            - v6 = an additional callee-saved variable register
                - A virtual platform that has no need for such a special register may designate r9 as an additional callee-saved variable register, v6
            - SB=Static Base：in a position-independent data model
            - TR=Thread Register：in an environment with thread-local storage
        - ARM64：X18
    - XR=indirect result Register=间接结果寄存器
        - 用途：保存函数调用返回的（结构体等非普通的数值的）结果（的指针地址）
            - 对比：x0、x1等保存函数返回的普通的数值类型的结果，即单个寄存器能保存的值
        - 对应寄存器
            - ARM32：无?
            - ARM64：X8
- 其他寄存器也有专门的大的分类
    - Parameter and Result Registers
        - 用途：作为参数和结果
            - 传递函数参数：调用其他（子）函数时，用x0、x1、x2等作为参数
            - 保存函数结果：函数调用返回后，结果保存在x0等寄存器中
        - 对应寄存器
            - ARM32：R0-R3
            - ARM64：X0-X7
        - 特殊
            - 一般保存返回结果，只用x0
                - 极个别情况下，才用到：x1，返回第二个函数结果
    - Caller和Callee
        - Caller-saved registers=调用者负责保存的寄存器=temporary registers=临时寄存器=Corruptible Registers=容易被破坏的寄存器
            - 用途：
                - 函数调用方 再调用 被调用函数之前 负责保存X9-X15
                - 调用子函数之前，如果这些寄存器中有保存值，记得要保存起来，即保

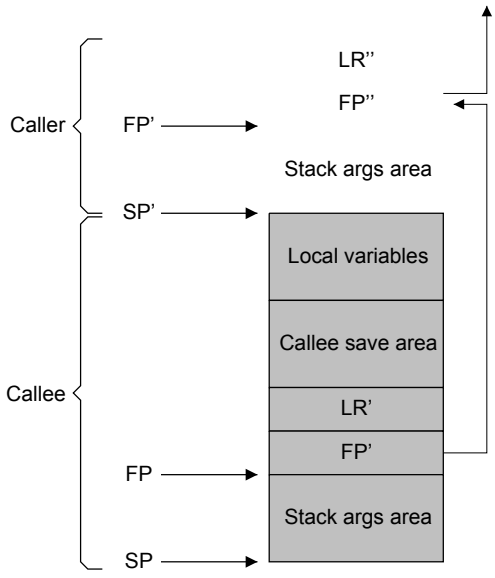
存上下文
- 函数调用后，再恢复这些寄存器，就恢复了现场
  - 汇编代码就可以继续运行了
- 对应寄存器
  - ARM32：好像就没有这个说法?
  - ARM64：X9-X15
- Callee-saved Registers=被调用者负责保存的寄存器
  - 用途
    - 被调用函数，在被调用后，刚要执行之前，负责保存X19-X29
  - 对应寄存器
    - ARM32：R4-R11
      - 除了R9特殊是PR=Platform (specific) Register
    - ARM64：X19-X28
- 其他说明
  - 寄存器保存值的类型
    - 写成W，比如W0,、W1等：用于保存integer整数
    - 写成D，比如D0、D1等：用于保存double双精度浮点数

- 其他相关
  - 除了上述介绍的普通寄存器之外，还有特殊寄存器
    - 状态寄存器
      - `CPSR` = `Current Program Status Register`
        - 旧称：`APSR` = `Application Program Status Register`
        - CPSR的bit位含义

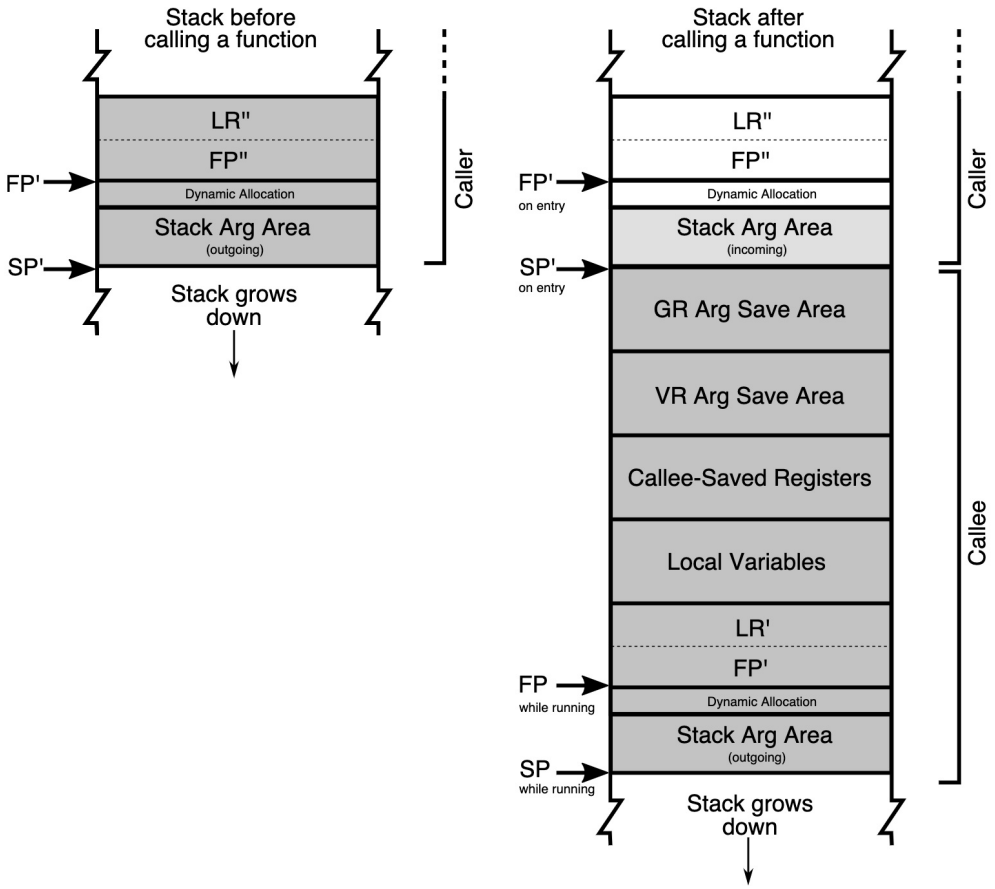


      - `SPSR` = `Saved Program Status Register`
        - 注：部分模式时才可用
  - 寄存器的叫法=概念
    - Global register
      - A register whose value is neither saved nor destroyed by a subroutine. The value may be updated, but only in a manner defined by the execution environment.
    - Scratch register / temporary register
      - A register used to hold an intermediate value during a calculation (usually, such values are not named in the program source and have a limited lifetime).
    - Variable register / v-register
      - A register used to hold the value of a variable, usually one local to a routine, and often named in the source code.
  - ARM指令是三级流水线，取指，译指，执行时同时执行的
    - 现在PC指向的是正在取指的地址
      - 假设在ARM状态 下，一个指令占4个字节

        - 那么
            - cpu正在译指的指令地址是PC-4
            - cpu正在执行的指令地址是PC-8
            - 下一个指令的值就是：PC+4
    - -》也就是说PC所指向的地址和现在所执行的指令地址相差8
        - 当突然发生中断的时候，保存的是PC的地址
        - 如果返回的时候返回PC，那么中间就有一个指令没有执行，所以用
            - SUB pc lr-irq #4
- 函数调用 相关
    - caller和callee

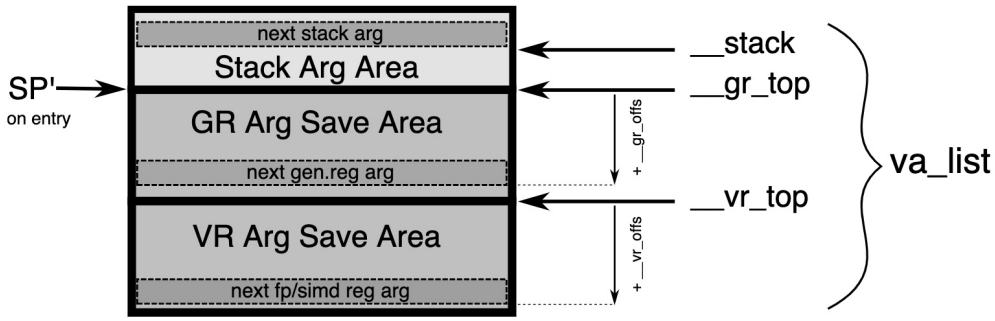


    - Example stack frame layout



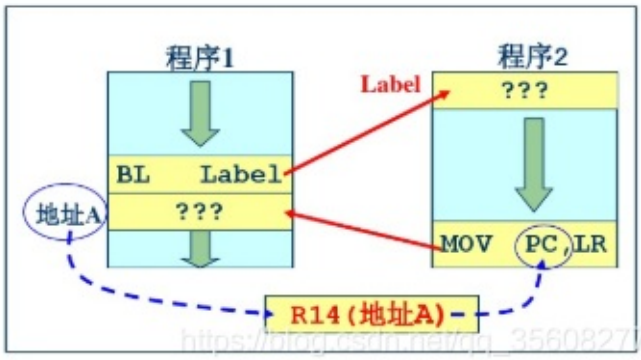
- Example stack frame layout

- The va_list



- The va_list

- LR寄存器 用法 图示



- 相关名字概念
  - prologue vs epilogue
    - prologue code = start of routine = 开场代码
      - 被调用函数的最开始一段代码要做的事情

        - epilogue code = 结尾代码
            - 被调用函数的最后一段代码要做的事情
    - caller-safe register vs callee-safe register
        - caller-safe register
            - caller负责保存的寄存器
                - 比如当r12用作scratch register时，calle被调用函数内，有可能会使用r12，改变r12的值，则此时caller函数调用者就要在调用calle之前，自己保存好r12的值
                    - -》Scratch register 也叫做caller-safe register
        - callee-safe register
            - calle负责保存的寄存器

调用规范

- 【已解决】ARM的ARM64汇编语法和寄存器、调用规范等基础知识
- 【已解决】ARM64中寄存器用法规范
- 【整理】ARM程序调用规范AAPCS：举例解释
- 【已解决】IDA中xsp和xbp是什么意思如何定位地址

背景知识

- ARM模式和寄存器
  - ARM处理器有7种（运行）模式
    - usr = User = 用户模式
    - fiq = Fast interrupt = 快速中断模式
    - irq= Interrupt = 普通中断模式
    - svc = Supervisor = 管理模式
    - abt = Abort = 数据访问中止模式
    - usr = System = 系统模式
      - User用户模式和System系统模式共用一套寄存器
    - und = Undefined = 未定义指令中止模式
  - 概述
    - Organization of general-purpose registers and Program Status Registers

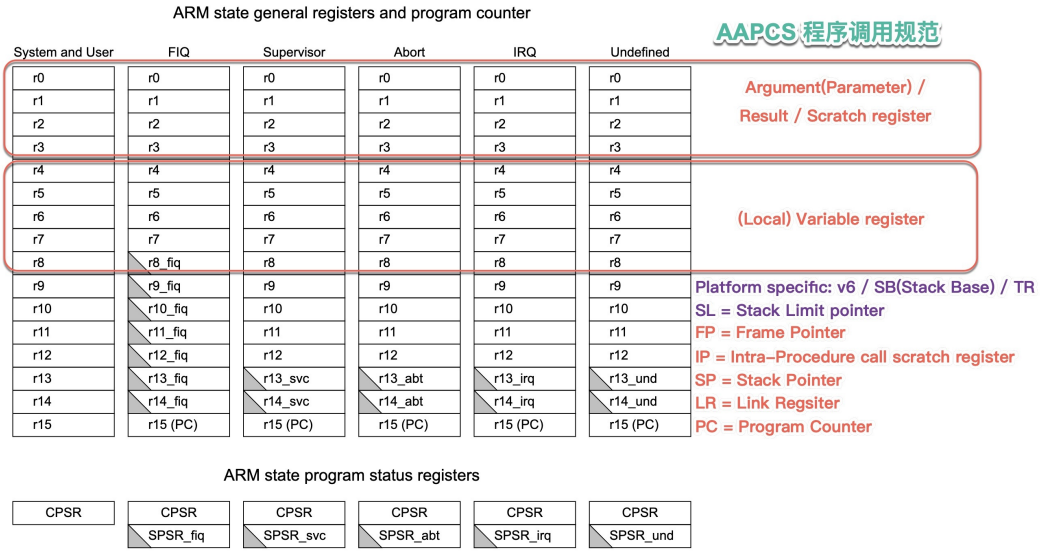
Application level view | System level views | Privileged modes | Exception modes

	User mode	System mode	Supervisor mode	Monitor mode ‡	Abort mode	Undefined mode	IRQ mode	FIQ mode
R0	R0_usr							
R1	R1_usr							
R2	R2_usr							
R3	R3_usr							
R4	R4_usr							
R5	R5_usr							
R6	R6_usr							
R7	R7_usr							
R8	R8_usr							R8_fiq
R9	R9_usr							R9_fiq
R10	R10_usr							R10_fiq
R11	R11_usr							R11_fiq
R12	R12_usr							R12_fiq
SP	SP_usr		SP_svc	SP_mon ‡	SP_abt	SP_und	SP_irq	SP_fiq
LR	LR_usr		LR_svc	LR_mon ‡	LR_abt	LR_und	LR_irq	LR_fiq
PC	PC							
APSR	CPSR							
			SPSR_svc	SPSR_mon ‡	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

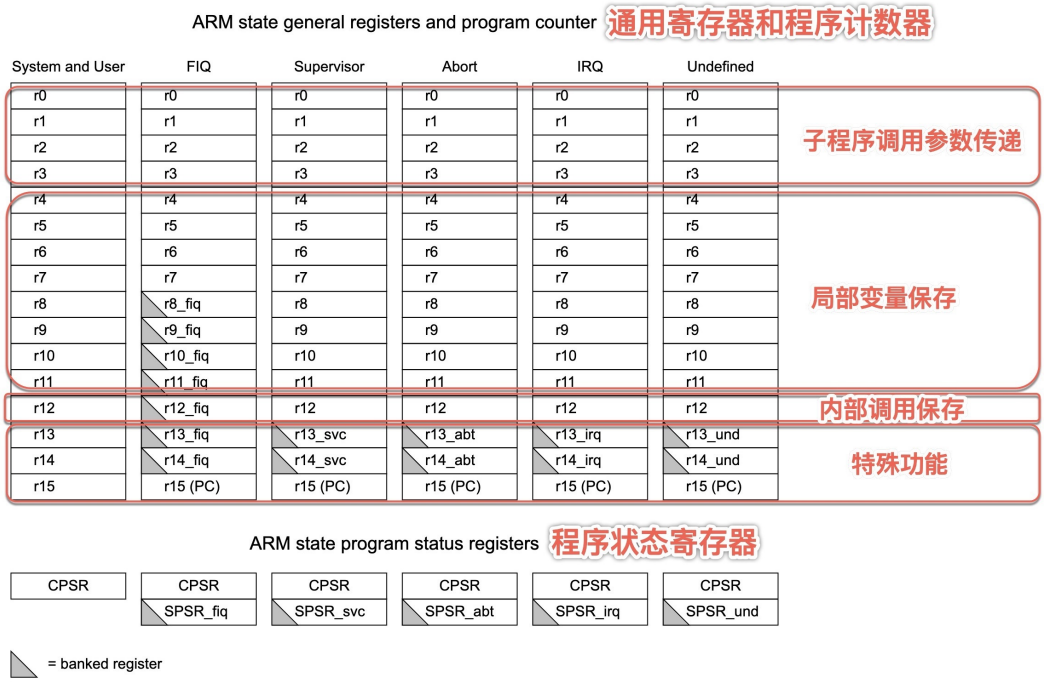
‡ Monitor mode and the associated banked registers are implemented only as part of the Security Extensions

AAPCS(Procedure Call Standard for the Arm Architecture) 程序调用规范 = ARM寄存器用法 用途

- `AAPCS` = `Procedure Call Standard for the Arm Architecture` = 程序调用规范
  - 含义：描述了ARM寄存器用途和用法
  - 图=概述
    - ARM状态的寄存器组织结构 + 寄存器用法



      - = banked register



      - = banked register

    - 核心寄存器和AAPCS用法

Register	Synonym	Special	Role in the procedure call standard
r15		PC	The Program Counter.
r14		LR	The Link Register.
r13		SP	The Stack Pointer.
r12		IP	The Intra-Procedure-call scratch register.
r11	v8		Variable-register 8.
r10	v7		Variable-register 7.
r9		v6 SB TR	Platform register. The meaning of this register is defined by the platform standard.
r8	v5		Variable-register 5.
r7	v4		Variable register 4.
r6	v3		Variable register 3.
r5	v2		Variable register 2.
r4	v1		Variable register 1.
r3	a4		Argument / scratch register 4.
r2	a3		Argument / scratch register 3.
r1	a2		Argument / result / scratch register 2.
r0	a1		Argument / result / scratch register 1.

Table 2, Core registers and AAPCS usage

- r9：平台寄存器: v6 / SB /TR
- r11 = SP
- r12 = IP

Table 6.1: Table 2, Core registers and AAPCS usage

Register	Synonym	Special	Role in the procedure call standard
r15		PC	The Program Counter.
r14		LR	The Link Register.
r13		SP	The Stack Pointer.
r12		IP	The Intra-Procedure-call scratch register.
r11	v8	FP	Frame Pointer or Variable-register 8.
r10	v7		Variable-register 7.
r9		v6 SB TR	Platform register. The meaning of this register is defined by the platform standard.
r8	v5		Variable-register 5.
r7	v4		Variable-register 4.
r6	v3		Variable-register 3.
r5	v2		Variable-register 2.
r4	v1		Variable-register 1.
r3	a4		Argument / scratch register 4.
r2	a3		Argument / scratch register 3.
r1	a2		Argument / result / scratch register 2.
r0	a1		Argument / result / scratch register 1.

Register	Synonym	Special	Role in the procedure call standard
r15	-	pc	Program counter.
r14	-	lr	Link register.
r13	-	sp	Stack pointer.
r12	-	ip	Intra-procedure-call scratch register.
r11	v8	-	ARM-state variable register 8.
r10	v7	sl	ARM-state variable register 7. Stack limit pointer in stack-checked variants.
r9	v6	sb	ARM-state variable register 6. Static base in RWPI variants.
r8	v5	-	ARM-state variable register 5.
r7	v4	-	Variable register 4.
r6	v3	-	Variable register 3.
r5	v2	-	Variable register 2.
r4	v1	-	Variable register 1.
r3	a4	-	Argument/result/scratch register 4.
r2	a3	-	Argument/result/scratch register 3.
r1	a2	-	Argument/result/scratch register 2.
r0	a1	-	Argument/result/scratch register 1.

- 从不同版本可以看出
  - 一般来说，函数返回值只用r0-r1
    - 偶尔也可以用r2-r3

- 文字
  - ARM Developer Suite Developer Guide 的 ATPCS
    - r0-r3
      - Use registers r0-r3 to pass parameter values into routines, and to pass result values out. You can refer to r0-r3 as a1-a4 to make this usage apparent. See Parameter passing. Between subroutine calls you can use r0-r3 for any purpose. A called routine does not have to restore r0-r3 before returning. A calling routine must preserve the contents of r0-r3 if it needs them again.
    - r4-r11
      - Use registers r4-r11 to hold the values of a routine's local variables. You can refer to them as v1-v8 to make this usage apparent. In Thumb state, in most instructions you can only use registers r4-r7 for local variables. A called routine must restore the values of these registers before returning, if it has used them.
    - r12
      - Register r12 is the intra-call scratch register, ip. It is used in this role in procedure linkage veneers, for example interworking veneers. Between procedure calls you can use it for any purpose. A called routine does not need to restore r12 before returning.
    - r13
      - Register r13 is the stack pointer, sp. You must not use it for any other purpose. The value held in sp on exit from a called routine must be the same as it was on entry.
    - r14

        - Register r14 is the link register, lr. If you save the return address, you can use r14 for other purposes between calls.
      - r15
        - Register r15 is the program counter, pc. It cannot be used for any other purpose.
    - Procedure Call Standard for the Arm Architecture - ABI 2020Q2 documentation
      - r0-r3
        - The first four registers r0-r3 (a1-a4) are used to pass argument values into a subroutine and to return a result value from a function. They may also be used to hold intermediate values within a routine (but, in general, only between subroutine calls).
      - r12
        - Register r12 (IP) may be used by a linker as a scratch register between a routine and any subroutine it calls (for details, see Use of IP by the linker). It can also be used within a routine to hold intermediate values between subroutine calls.
      - r11
        - In some variants r11 (FP) may be used as a frame pointer in order to chain frame activation records into a linked list.
      - r9
        - The role of register r9 is platform specific. A virtual platform may assign any role to this register and must document this usage. For example, it may designate it as the static base (SB) in a position-independent data model, or it may designate it as the thread register (TR) in an environment with thread-local storage. The usage of this register may require that the value held is persistent across all calls. A virtual platform that has no need for such a special register may designate r9 as an additional callee-saved variable register, v6.
      - r4-r8, r10, r11
        - Typically, the registers r4-r8, r10 and r11 (v1-v5, v7 and v8) are used to hold the values of a routine's local variables. Of these, only v1-v4 can be used uniformly by the whole Thumb instruction set, but the AAPCS does not require that Thumb code only use those registers.
        - A subroutine must preserve the contents of the registers r4-r8, r10, r11 and SP (and r9 in PCS variants that designate r9 as v6).
      - r12-r15
        - In all variants of the procedure call standard, registers r12-r15 have special roles. In these roles they are labeled IP, SP, LR and PC.
  - 相关资料
    - github.com
      - ARM-software/abi-aa: Application Binary Interface for the Arm® Architecture (github.com)
        - aapcs32
          - abi-aa/aapcs32.rst at main · ARM-software/abi-aa (github.com)
        - aapcs64
          - abi-aa/aapcs64.rst at main · ARM-software/abi-aa (github.com)
        - Releases · ARM-software/abi-aa (github.com)
          - ARM64
            - ABI for the Arm 64-bit Architecture

        - Procedure Call Standard for the Arm 64-bit Architecture
            - pdf
                - https://github.com/ARM-software/abi-aa/releases/download/2021Q1/aapcs64.pdf
            - html
                - https://github.com/ARM-software/abi-aa/blob/2bcab1e3b22d55170c563c3c7940134089176746/aapcs64/aapcs64.rst
- arm.com
    - 和软件开发相关资料的入口
        - Develop Software – Arm Developer
    - 各种软件资料
        - System Architectures | Application Binary Interface (ABI) – Arm Developer
            - Procedure Call Standard for the Arm 64-bit Architecture

ARM寄存器

ARM寄存器命名

ARM中，根据bit位宽，分：32位和64位，分别叫：

- `32位` = `ARM32`
- `64位` = `ARM64`

其中：

- 寄存器
  - 英文：`Register`
    - 缩写：`Reg`
      - ARM中简称：`R`

此处整理ARM中寄存的命名和叫法：

- ARM
  - 寄存器= `Register` = `Reg`
  - 通用叫法：`R`
    - 所以一般叫做：`R1` 、`R2` 、...、`R15` 、`R16` 、...、`R31`
    - `ARM32`：`W`
      - 一共有 `16` 个，所以分别叫：
        - `X1` 、`X2` 、。。。。、`X15`
      - 但是由于历史原因：`ARM32` 中也常叫做：`R`
        - 所以 `ARM32` 中也常用：`R1` 、`R2` 、...、`R15`
    - `ARM64`：`X`
      - 一共有 `32` 个，所以分别叫：
        - `X1` 、`X2` 、`X31`

寄存器架构

ARM根据位数分32位和64位，每种架构都有很多寄存器，以及对应的特定用法和叫法。

对于ARM寄存器的总体架构，整理如下：

- 在线浏览
  - ARM寄存器架构 | ProcessOn免费在线作图
- 本地查看

2024-09-25 10:03:55

常用寄存器

TODO:把x0到x31 把其中比较常用的，有需要单独解释的寄存器，整理出来

PC

- `PC` = `Program Counter` = `程序计数器`
  - 对应寄存器
    - `AArch32` = `ARM32` == `R15`
    - `AArch64` = `ARM64` != `X31`
      - AArch64中有PC，但不是通用寄存器X31
        - 无法直接访问到PC的值
        - 只有其他一些操作才能访问PC的值
          - 读取PC
            - 计算相对地址
          - 写入PC
            - 特殊的跳转类的指令
  - 用途：记录当前CPU的地址，总是指向正在“取指”的指令
    - 流水线（一般）使用三个阶段，因此指令分为三个阶段执行
      - 1、取指：从存储器装载一条指令
      - 2、译码：识别将要被执行的指令
      - 3、执行：处理 指令并将结果写回寄存器
    - -》
      - PC总是指向正在“取指”的指令
        - 而不是指向正在“执行”的指令 或 正在“译码”的指令
      - 一般来说，人们习惯性约定 将“正在执行的指令作为参考点”，称之为当前第一条指令，因此PC总是指向第三条指令
      - 当ARM状态时，每条指令为4字节长，所以PC始终指向该指令地址加8字节的地址，即：PC值=当前程序执行位置+8
    - -> ARM指令是三级流水线，取指，译指，执行时同时执行的
      - 现在PC指向的是正在取指的地址
        - 假设在ARM状态 下，一个指令占4个字节
          - 那么
            - cpu正在译指的指令地址是PC-4
            - cpu正在执行的指令地址是PC-8
            - 下一个指令的值就是：PC+4
      - -》也就是说PC所指向的地址和现在所执行的指令地址相差8
        - 当突然发生中断的时候，保存的是PC的地址
        - 如果返回的时候返回PC，那么中间就有一个指令没有执行，所以用
          - SUB pc lr-irq #4

IP

- 【已解决】ARM中的R12寄存器IP

SP

- `R13` = `SP` = `Stack Pointer` = `栈指针` = `堆栈指针`
  - 每一种异常模式都有其自己独立的r13，它通常指向异常模式所专用的堆栈，也就是说五种异常模式、非异常模式（用户模式和系统模式），都有各自独立的堆栈，用不同的堆栈指针来索引。这样当ARM进入异常模式的时候，程序就可以把一般通用寄存器压入堆栈，返回时再出栈，保证了各种模式下程序的状态的完整性

LR

- `LR` = `Link Register` = `链接寄存器` = `Subroutine Link Register` = `子程序连接寄存器`
  - 作用=用途：用于函数调用function calls
    - 当发生跳转时，记录原先地址（用于后续返回地址）
  - ARM32
    - 英文概述
      - Register r14 is used as the subroutine Link Register (LR).
      - Register r14 receives the return address when a Branch with Link (BL or BLX) instruction is executed.
      - You can treat r14 as a general-purpose register at all other times. The corresponding banked registers r14_svc, r14_irq, r14_fiq, r14_abt, and r14_und are similarly used to hold the return values when interrupts and exceptions arise, or when BL or BLX instructions are executed within interrupt or exception routines.
    - 在ARM汇编语言程序设计时，R14和LR通用
    - ARM的每种模式都有各自专用的R14 寄存器=LR
      - R14
        - User用户模式和System系统模式共用一个LR
      - R14_fiq
      - R14_irq
      - R14_svc
      - R14_abt
      - R14_und
    - 注意
      - 程序设计者要清晰处理器的模式与相应寄存器的对应关系，都是使用 R14，但不同模式下的R14 不是同一个物理资源，其内容很可能不同
  - 对应寄存器
    - ARM32：`R14`
    - ARM64：`X30`

跳转逻辑

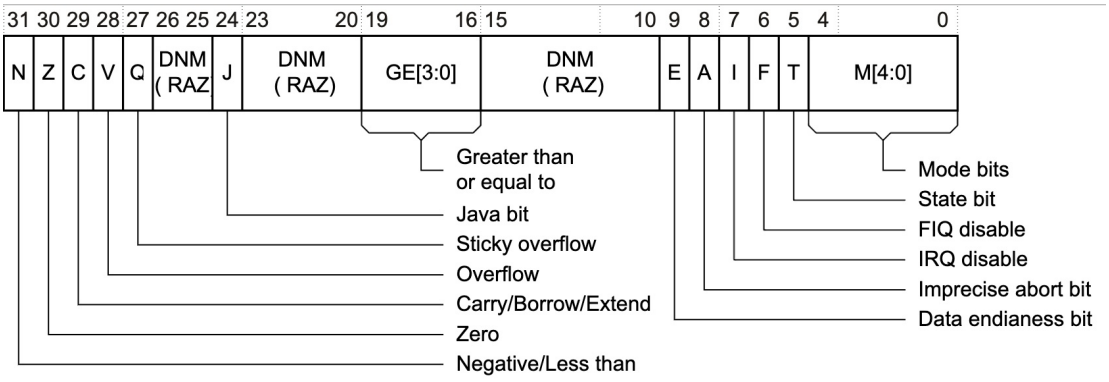
- 有几种情况会发生跳转，以及对应的逻辑是
  - 子程序的调用（和返回）
    - 作用：保存子程序返回地址
    - 相关指令
      - BL
      - BLX
    - 内部逻辑
      - （ARM处理器）执行（BL或BLX）跳转指令时，（ARM处理器）自动把返回地址放入r14中
        - LR的值 = 当前（BL或BLX）BL指令地址 - 4 = PC - 4
    - 子程序返回时
      - 核心逻辑：把R14=LR赋值到PC

        - 三种方法
            - MOV PC, LR
            - BX LR
                - 或：BL LR
            - 把R14压栈再出栈
                - 在子程序入口处使用以下指令将R14存入堆栈
                    - `STMFD SP!,{<Regs>,LR}`
                        - == `stmfd sp!, {lr} ?`
                - 对应的，使用以下指令可以完成子程序的返回
                    - `LDMFD SP!, {<Regs>,LR}`
                        - == `ldmfd sp!, {pc} ?`
    - 中断和异常（Interrupt and Exception）
        - 常见异常
            - fiq
            - irq
        - 作用：跳转到异常处理=异常响应
        - ARM处理器发生异常时，异常模式的r14用来保存异常返回地址，即 PC值给LR
            - 将r14入栈可以处理嵌套中断

状态寄存器

- 【已解决】ARM汇编指令：CPSR当前程序状态寄存器和标志位

CPSR

- `CPSR` = `Current Program Status Register` = 当前程序状态寄存器
  - 旧称：`APSR` = `Application Program Status Register`
  - CPSR的bit位含义
    - 

其他寄存器

TODO：

- 【已解决】ARM汇编寄存器：TPIDRRO_EL0
- 【已解决】ARM寄存器：q0 q1
- 【已解决】ARM寄存器：q0 q1 v0 S0 H0 B0

AArch64=ARM64

特殊用途寄存器

AArch64=ARM64中，有一系列的：

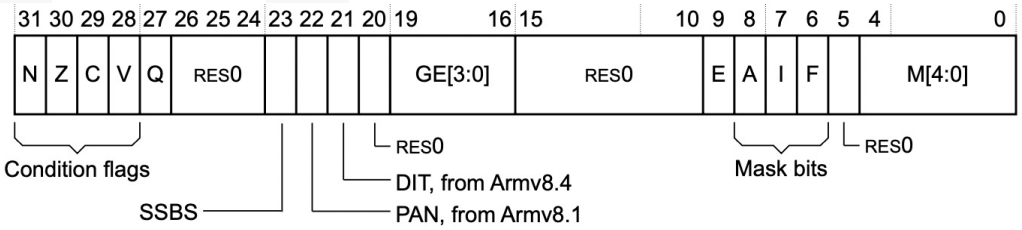
- `特殊用途寄存器 = Special-purpose registers`
  - 包括

特殊用途寄存器	对应PSTATE中的字段/域
`NZCV`	`N, Z, C, V`
`DAIF`	`D, A, I, F`
`CurrentEL`	`EL`
`SPSel`	`SP`
`PAN`	`PAN`
`UAO`	`UAO`
`DIT`	`DIT`
`SSBS`	`SSBS`
`TCO`	`TCO`

- 用法
  - 读取和写入，用特殊的指令
    - `MSR`
      - 举例
        - NZCV
          - `MSR NZCV, <Xt>`
    - `MRS`
      - 举例
        - NZCV
          - `MRS <Xt>, NZCV`

NZCV

- `NZCV`
  - 含义
    - `N` = `Negative` Condition flag= `借位标志`
    - `Z` = `Zero` Condition flag= `0`
    - `C` = `Carry` Condition flag=进位标志
    - `V` = `Overflow` Condition flag=溢出标志
  - ARM32
    - 是 `CPSR` 状态寄存器保存了 `NZCV` 的值



    - ARM64
      - 具体表示和写法
        - 要表示 `PSTATE` 中的某一标志，比如 `进位` ，可以写成 `PSTATE.C`
      - 读取和写入，用特殊的指令
        - `MSR`

```
MSR NZCV, Xt
```

        - `MRS`

```
MRS Xt , NZCV
```

      - ARM手册

C5.2.9 NZCV, Condition Flags

The NZCV characteristics are:

Purpose

Allows access to the condition flags.

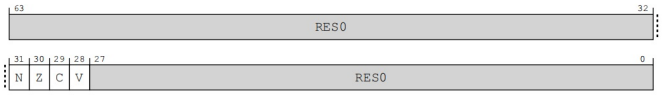
Configurations

There are no configuration notes.

Attributes

NZCV is a 64-bit register.

Field descriptions



Bits [63:32]

Reserved, RES0.

N, bit [31]

Negative condition flag. Set to 1 if the result of the last flag-setting instruction was negative.

Z, bit [30]

Zero condition flag. Set to 1 if the result of the last flag-setting instruction was zero, and to 0 otherwise. A result of zero often indicates an equal result from a comparison.

C, bit [29]

Carry condition flag. Set to 1 if the last flag-setting instruction resulted in a carry condition, for example an unsigned overflow on an addition.

V, bit [28]

Overflow condition flag. Set to 1 if the last flag-setting instruction resulted in an overflow condition, for example a signed overflow on an addition.

Bits [27:0]

Reserved, RES0.

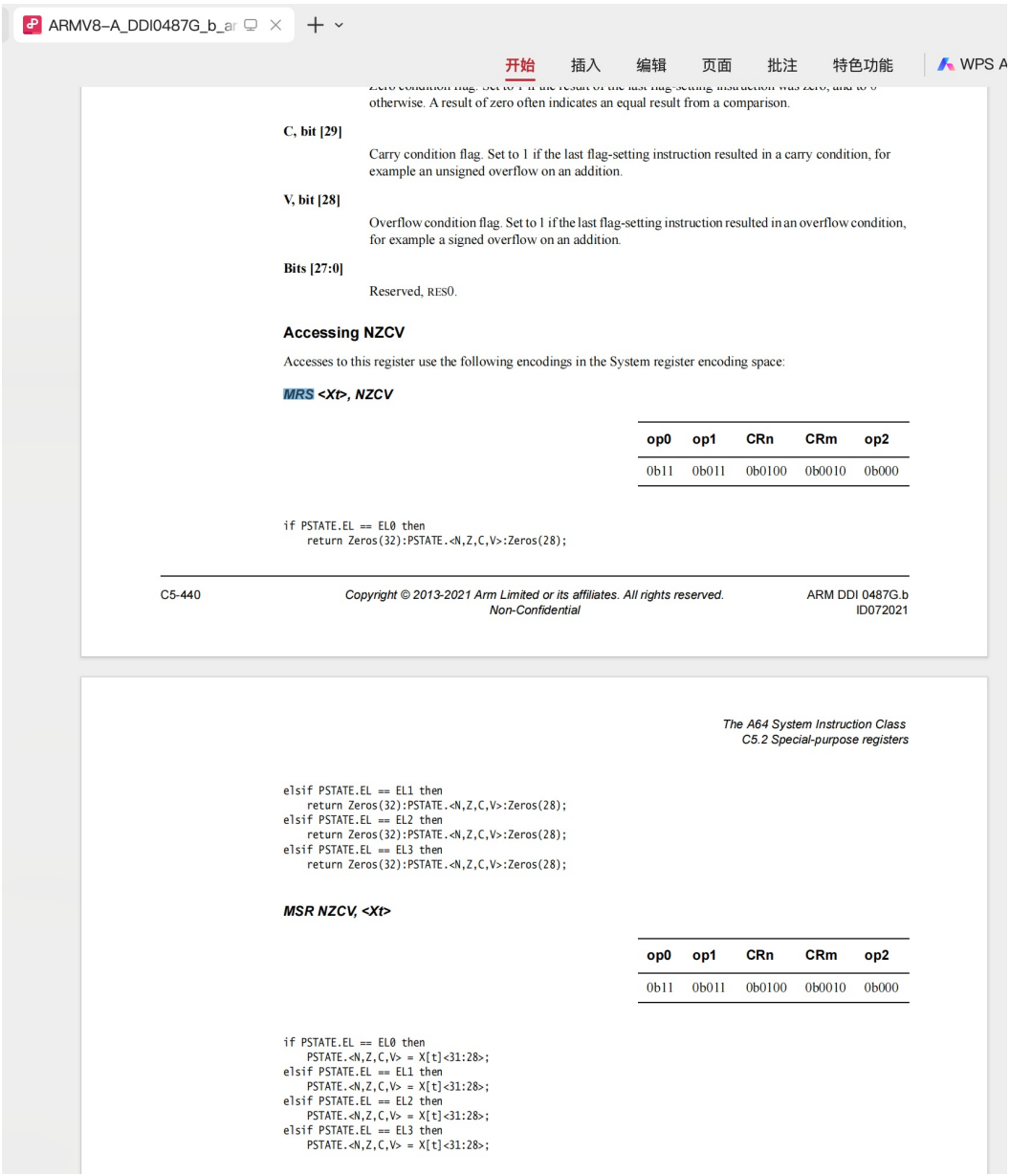
Accessing NZCV

Accesses to this register use the following encodings in the System register encoding space:

MRS <Xt>, NZCV

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b000

```
if PSTATE.EL == EL0 then
    return Zeros(32):PSTATE.<N,Z,C,V>:Zeros(28);
```



otherwise. A result of zero often indicates an equal result from a comparison.

C, bit [29]

Carry condition flag. Set to 1 if the last flag-setting instruction resulted in a carry condition, for example an unsigned overflow on an addition.

V, bit [28]

Overflow condition flag. Set to 1 if the last flag-setting instruction resulted in an overflow condition, for example a signed overflow on an addition.

Bits [27:0]

Reserved, RES0.

Accessing NZCV

Accesses to this register use the following encodings in the System register encoding space:

***MRS** <Xt>, NZCV*

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b000

```
if PSTATE.EL == EL0 then
    return Zeros(32):PSTATE.<N,Z,C,V>:Zeros(28);
```



```
elsif PSTATE.EL == EL1 then
    return Zeros(32):PSTATE.<N,Z,C,V>:Zeros(28);
elsif PSTATE.EL == EL2 then
    return Zeros(32):PSTATE.<N,Z,C,V>:Zeros(28);
elsif PSTATE.EL == EL3 then
    return Zeros(32):PSTATE.<N,Z,C,V>:Zeros(28);
```

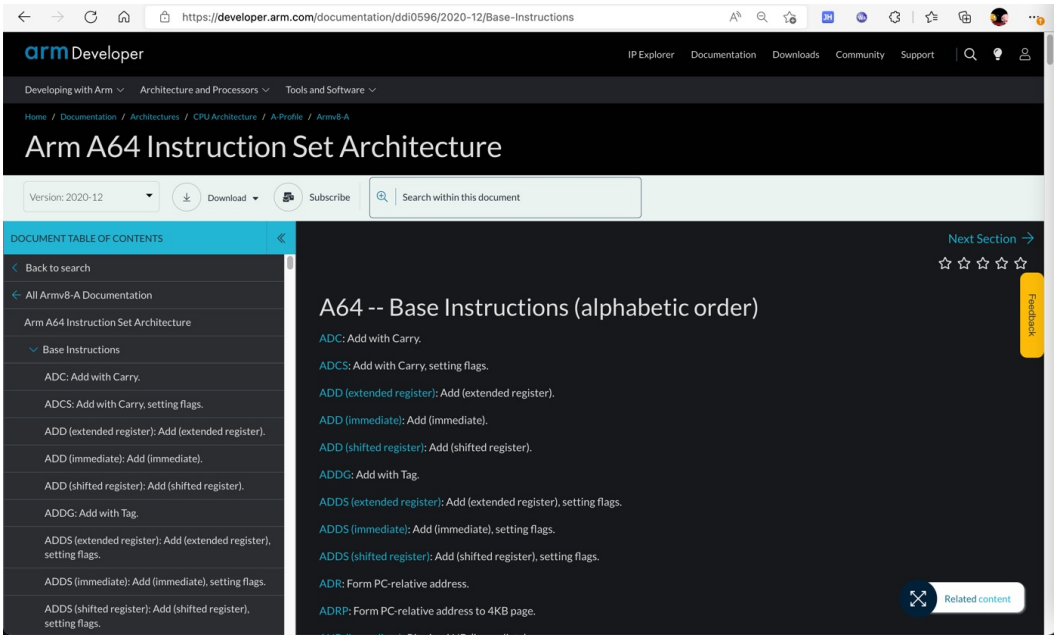
***MSR** NZCV, <Xt>*

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b000

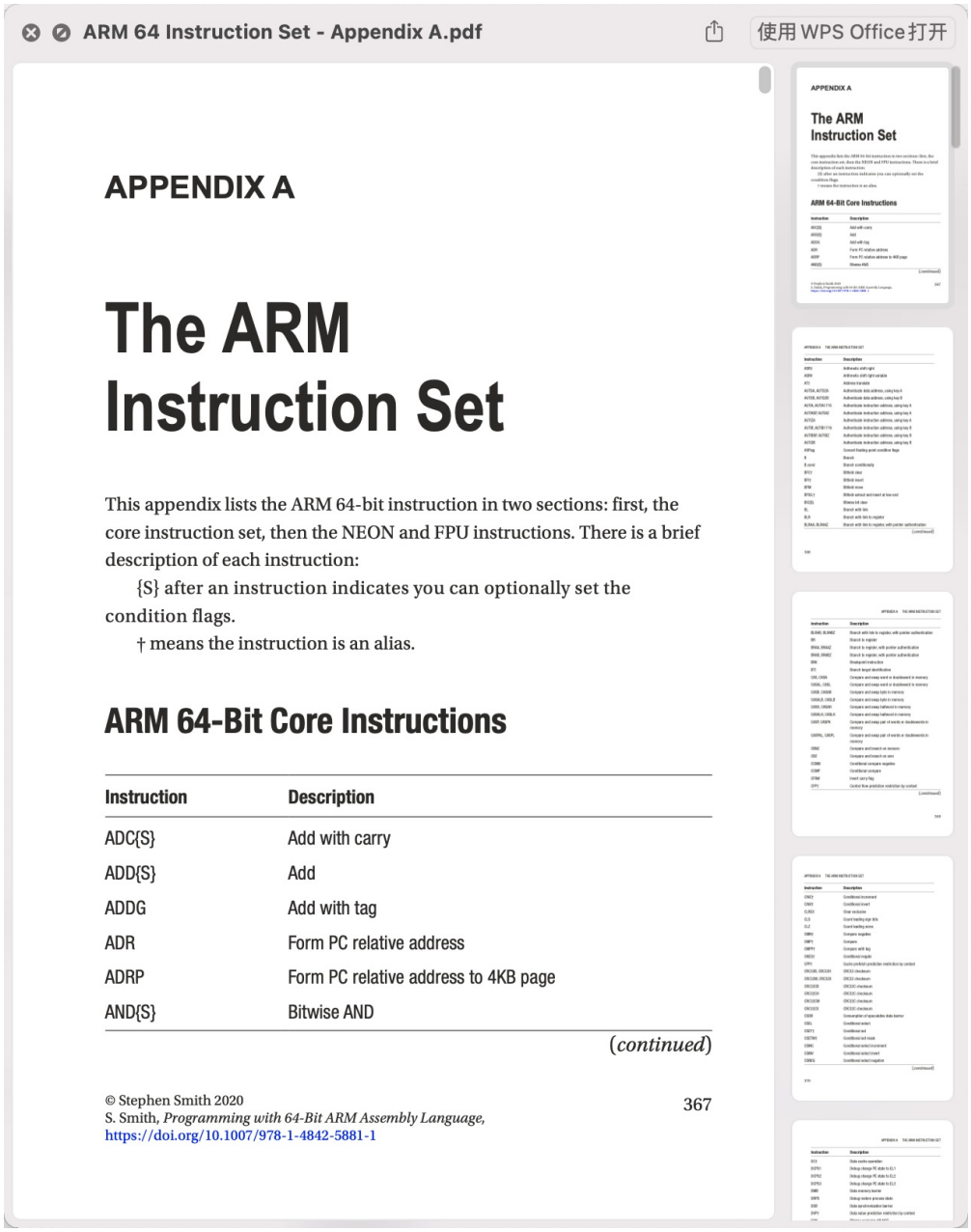
```
if PSTATE.EL == EL0 then
    PSTATE.<N,Z,C,V> = X[t]<31:28>;
elsif PSTATE.EL == EL1 then
    PSTATE.<N,Z,C,V> = X[t]<31:28>;
elsif PSTATE.EL == EL2 then
    PSTATE.<N,Z,C,V> = X[t]<31:28>;
elsif PSTATE.EL == EL3 then
    PSTATE.<N,Z,C,V> = X[t]<31:28>;
```

ARM汇编指令

ARM汇编指令列表

- 资料来源
  - 很全的指令的列表
    - A Guide to ARM64 / AArch64 Assembly on Linux with Shellcodes and Cryptography | modexp (wordpress.com)
  - 官网的专业介绍
    - Arm A64 Instruction Set Architecture
    - 

  - ARM 64 汇编指令集
    - 包含
      - 核心指令
      - NEON和FPU指令集
    - 详见
      - https://link.springer.com/content/pdf/bbm%3A978-1-4842-5881-1%2F1.pdf



按字母排序的ARM汇编指令列表

参考：

Arm A-profile A64 Instruction Set Architecture - 2024-06

的指令列表：

A64 -- Base Instructions (alphabetic order)

```
ABS: Absolute value.
ADC: Add with carry.
ADCS: Add with carry, setting flags.
ADD (extended register): Add (extended register).
ADD (immediate): Add (immediate).
ADD (shifted register): Add (shifted register).
ADDG: Add with tag.
ADDPT: Add checked pointer.
ADDS (extended register): Add (extended register), setting flags.
ADDS (immediate): Add (immediate), setting flags.
ADDS (shifted register): Add (shifted register), setting flags.
ADR: Form PC relative address.
ADRP: Form PC relative address to 4KB page.
AND (immediate): Bitwise AND (immediate).
AND (shifted register): Bitwise AND (shifted register).
ANDS (immediate): Bitwise AND (immediate), setting flags.
ANDS (shifted register): Bitwise AND (shifted register), setting flags.
ASR (immediate): Arithmetic shift right (immediate): an alias of SBFM.
ASR (register): Arithmetic shift right (register): an alias of ASRV.
ASRV: Arithmetic shift right variable.
AT: Address translate: an alias of SYS.
AUTDA, AUTDZA: Authenticate data address, using key A.
AUTDB, AUTDZB: Authenticate data address, using key B.
AUTIA, AUTIA1716, AUTIASP, AUTIAZ, AUTIZA: Authenticate instruction address, using key
A.
AUTIA171615: Authenticate instruction address, using key A.
AUTIASPPC: Authenticate return address using key A, using an immediate offset.
AUTIASPPCR: Authenticate return address using key A, using a register.
AUTIB, AUTIB1716, AUTIBSP, AUTIBZ, AUTIZB: Authenticate instruction address, using key
B.
AUTIB171615: Authenticate instruction address, using key B.
AUTIBSPPC: Authenticate return address using key B, using an immediate offset.
AUTIBSPPCR: Authenticate return address using key B, using a register.
AXFLAG: Convert floating point condition flags from Arm to external format.
B: Branch.
B.cond: Branch conditionally.
BC.cond: Branch consistent conditionally.
BFC: Bitfield clear: an alias of BFM.
BFI: Bitfield insert: an alias of BFM.
BFM: Bitfield move.
BFXIL: Bitfield extract and insert at low end: an alias of BFM.
BIC (shifted register): Bitwise bit clear (shifted register).
```

BICS (shifted register): Bitwise bit clear (shifted register), setting flags.
BL: Branch with link.
BLR: Branch with link to register.
BLRAA, BLRAAZ, BLRAB, BLRABZ: Branch with link to register, with pointer authentication.
BR: Branch to register.
BRAA, BRAAZ, BRAB, BRABZ: Branch to register, with pointer authentication.
BRB: Branch record buffer: an alias of SYS.
BRK: Breakpoint instruction.
BTI: Branch target identification.
CAS, CASA, CASAL, CASL: Compare and swap word or doubleword in memory.
CASB, CASAB, CASALB, CASLB: Compare and swap byte in memory.
CASH, CASAH, CASALH, CASLH: Compare and swap halfword in memory.
CASP, CASPA, CASPAL, CASPL: Compare and swap pair of words or doublewords in memory.
CBNZ: Compare and branch on nonzero.
CBZ: Compare and branch on zero.
CCMN (immediate): Conditional compare negative (immediate).
CCMN (register): Conditional compare negative (register).
CCMP (immediate): Conditional compare (immediate).
CCMP (register): Conditional compare (register).
CFINV: Invert carry flag.
CFP: Control flow prediction restriction by context: an alias of SYS.
CHKFEAT: Check feature status.
CINC: Conditional increment: an alias of CSINC.
CINV: Conditional invert: an alias of CSINV.
CLRBHB: Clear branch history.
CLREX: Clear exclusive.
CLS: Count leading sign bits.
CLZ: Count leading zeros.
CMN (extended register): Compare negative (extended register): an alias of ADDS (extended register).
CMN (immediate): Compare negative (immediate): an alias of ADDS (immediate).
CMN (shifted register): Compare negative (shifted register): an alias of ADDS (shifted register).
CMP (extended register): Compare (extended register): an alias of SUBS (extended register).
CMP (immediate): Compare (immediate): an alias of SUBS (immediate).
CMP (shifted register): Compare (shifted register): an alias of SUBS (shifted register).
CMPP: Compare with tag: an alias of SUBPS.
CNEG: Conditional negate: an alias of CSNEG.
CNT: Count bits.
COSP: Clear other speculative prediction restriction by context: an alias of SYS.
CPP: Cache prefetch prediction restriction by context: an alias of SYS.
CPYFP, CPYFM, CPYFE: Memory copy forward only.
CPYFPN, CPYFMN, CPYFEN: Memory copy forward only, reads and writes non temporal.
CPYFPRN, CPYFMRN, CPYFERN: Memory copy forward only, reads non temporal.
CPYFPRT, CPYFMRT, CPYFERT: Memory copy forward only, reads unprivileged.
CPYFPRTN, CPYFMRTN, CPYFERTN: Memory copy forward only, reads unprivileged, reads and writes non temporal.
CPYFPRTRN, CPYFMRTRN, CPYFERTRN: Memory copy forward only, reads unprivileged and non temporal.
CPYFPRTWN, CPYFMRTWN, CPYFERTWN: Memory copy forward only, reads unprivileged, writes non temporal.
CPYFPT, CPYFMT, CPYFET: Memory copy forward only, reads and writes unprivileged.
CPYFPTN, CPYFMTN, CPYFETN: Memory copy forward only, reads and writes unprivileged and

```
non temporal.
CPYFPTRN, CPYFMTRN, CPYFETRN: Memory copy forward only, reads and writes unprivileged,
reads non temporal.
CPYFPTWN, CPYFMTWN, CPYFETWN: Memory copy forward only, reads and writes unprivileged,
writes non temporal.
CPYFPWN, CPYFMWN, CPYFEWN: Memory copy forward only, writes non temporal.
CPYFPWT, CPYFMWT, CPYFEWT: Memory copy forward only, writes unprivileged.
CPYFPWTN, CPYFMWTN, CPYFEWTN: Memory copy forward only, writes unprivileged, reads and
writes non temporal.
CPYFPWTRN, CPYFMWTRN, CPYFEWTRN: Memory copy forward only, writes unprivileged, reads n
on temporal.
CPYFPWTWN, CPYFMWTWN, CPYFEWTWN: Memory copy forward only, writes unprivileged and non
temporal.
CPYP, CPYM, CPYE: Memory copy.
CPYPN, CPYMN, CPYEN: Memory copy, reads and writes non temporal.
CPYPRN, CPYMRN, CPYERN: Memory copy, reads non temporal.
CPYPRT, CPYMRT, CPYERT: Memory copy, reads unprivileged.
CPYPRTN, CPYMRTN, CPYERTN: Memory copy, reads unprivileged, reads and writes non tempor
al.
CPYPRTRN, CPYMRTRN, CPYERTRN: Memory copy, reads unprivileged and non temporal.
CPYPRTWN, CPYMRTWN, CPYERTWN: Memory copy, reads unprivileged, writes non temporal.
CPYPT, CPYMT, CPYET: Memory copy, reads and writes unprivileged.
CPYPTN, CPYMTN, CPYETN: Memory copy, reads and writes unprivileged and non temporal.
CPYPTRN, CPYMTRN, CPYETRN: Memory copy, reads and writes unprivileged, reads non tempor
al.
CPYPTWN, CPYMTWN, CPYETWN: Memory copy, reads and writes unprivileged, writes non tempo
ral.
CPYPWN, CPYMWN, CPYEWN: Memory copy, writes non temporal.
CPYPWT, CPYMWT, CPYEWT: Memory copy, writes unprivileged.
CPYPWTN, CPYMWTN, CPYEWTN: Memory copy, writes unprivileged, reads and writes non tempo
ral.
CPYPWTRN, CPYMWTRN, CPYEWTRN: Memory copy, writes unprivileged, reads non temporal.
CPYPWTWN, CPYMWTWN, CPYEWTWN: Memory copy, writes unprivileged and non temporal.
CRC32B, CRC32H, CRC32W, CRC32X: CRC32 checksum.
CRC32CB, CRC32CH, CRC32CW, CRC32CX: CRC32C checksum.
CSDB: Consumption of speculative data barrier.
CSEL: Conditional select.
CSET: Conditional set: an alias of CSINC.
CSETM: Conditional set mask: an alias of CSINV.
CSINC: Conditional select increment.
CSINV: Conditional select invert.
CSNEG: Conditional select negation.
CTZ: Count trailing zeros.
DC: Data cache operation: an alias of SYS.
DCPS1: Debug change PE state to EL1.
DCPS2: Debug change PE state to EL2.
DCPS3: Debug change PE state to EL3.
DGH: Data gathering hint.
DMB: Data memory barrier.
DRPS: Debug restore PE state.
DSB: Data synchronization barrier.
DVP: Data value prediction restriction by context: an alias of SYS.
EON (shifted register): Bitwise exclusive OR NOT (shifted register).
EOR (immediate): Bitwise exclusive OR (immediate).
EOR (shifted register): Bitwise exclusive OR (shifted register).
ERET: Exception return.
```

```
ERETAA, ERETAB: Exception return, with pointer authentication.
ESB: Error synchronization barrier.
EXTR: Extract register.
GCSB: Guarded Control Stack barrier.
GCSPOPCX: Guarded Control Stack pop and compare exception return record: an alias of SY
S.
GCSPOPM: Guarded Control Stack pop: an alias of SYSL.
GCSPOPX: Guarded Control Stack pop exception return record: an alias of SYS.
GCSPUSHM: Guarded Control Stack push: an alias of SYS.
GCSPUSHX: Guarded Control Stack push exception return record: an alias of SYS.
GCSSS1: Guarded Control Stack switch stack 1: an alias of SYS.
GCSSS2: Guarded Control Stack switch stack 2: an alias of SYSL.
GCSSTR: Guarded Control Stack store.
GCSSTTR: Guarded Control Stack unprivileged store.
GMI: Tag mask insert.
HINT: Hint instruction.
HLT: Halt instruction.
HVC: Hypervisor call.
IC: Instruction cache operation: an alias of SYS.
IRG: Insert random tag.
ISB: Instruction synchronization barrier.
LD64B: Single copy atomic 64 byte Load.
LDADD, LDADDA, LDADDAL, LDADDL: Atomic add on word or doubleword in memory.
LDADDB, LDADDAB, LDADDALB, LDADDLB: Atomic add on byte in memory.
LDADDH, LDADDAH, LDADDALH, LDADDLH: Atomic add on halfword in memory.
LDAPR: Load acquire RCpc register.
LDAPRB: Load acquire RCpc register byte.
LDAPRH: Load acquire RCpc register halfword.
LDAPUR: Load acquire RCpc register (unscaled).
LDAPURB: Load acquire RCpc register byte (unscaled).
LDAPURH: Load acquire RCpc register halfword (unscaled).
LDAPURSB: Load acquire RCpc register signed byte (unscaled).
LDAPURSH: Load acquire RCpc register signed halfword (unscaled).
LDAPURSW: Load acquire RCpc register signed word (unscaled).
LDAR: Load acquire register.
LDARB: Load acquire register byte.
LDARH: Load acquire register halfword.
LDAXP: Load acquire exclusive pair of registers.
LDAXR: Load acquire exclusive register.
LDAXRB: Load acquire exclusive register byte.
LDAXRH: Load acquire exclusive register halfword.
LDCLR, LDCLRA, LDCLRAL, LDCLRL: Atomic bit clear on word or doubleword in memory.
LDCLRB, LDCLRAB, LDCLRALB, LDCLRLB: Atomic bit clear on byte in memory.
LDCLRH, LDCLRAH, LDCLRALH, LDCLRLH: Atomic bit clear on halfword in memory.
LDCLRP, LDCLRPA, LDCLRPAL, LDCLRPL: Atomic bit clear on quadword in memory.
LDEOR, LDEORA, LDEORAL, LDEORL: Atomic exclusive OR on word or doubleword in memory.
LDEORB, LDEORAB, LDEORALB, LDEORLB: Atomic exclusive OR on byte in memory.
LDEORH, LDEORAH, LDEORALH, LDEORLH: Atomic exclusive OR on halfword in memory.
LDG: Load Allocation Tag.
LDGM: Load tag multiple.
LDIAPP: Load Acquire RCpc ordered pair of registers.
LDLAR: Load LOAcquire register.
LDLARB: Load LOAcquire register byte.
LDLARH: Load LOAcquire register halfword.
LDNP: Load pair of registers, with non temporal hint.
LDP: Load pair of registers.
```

```
LDPSW: Load pair of registers signed word.
LDR (immediate): Load register (immediate).
LDR (literal): Load register (literal).
LDR (register): Load register (register).
LDRAA, LDRAB: Load register, with pointer authentication.
LDRB (immediate): Load register byte (immediate).
LDRB (register): Load register byte (register).
LDRH (immediate): Load register halfword (immediate).
LDRH (register): Load register halfword (register).
LDRSB (immediate): Load register signed byte (immediate).
LDRSB (register): Load register signed byte (register).
LDRSH (immediate): Load register signed halfword (immediate).
LDRSH (register): Load register signed halfword (register).
LDRSW (immediate): Load register signed word (immediate).
LDRSW (literal): Load register signed word (literal).
LDRSW (register): Load register signed word (register).
LDSET, LDSETA, LDSETAL, LDSETL: Atomic bit set on word or doubleword in memory.
LDSETB, LDSETAB, LDSETALB, LDSETLB: Atomic bit set on byte in memory.
LDSETH, LDSETAH, LDSETALH, LDSETLH: Atomic bit set on halfword in memory.
LDSETP, LDSETPA, LDSETPAL, LDSETPL: Atomic bit set on quadword in memory.
LDSMAX, LDSMAXA, LDSMAXAL, LDSMAXL: Atomic signed maximum on word or doubleword in memo
ry.
LDSMAXB, LDSMAXAB, LDSMAXALB, LDSMAXLB: Atomic signed maximum on byte in memory.
LDSMAXH, LDSMAXAH, LDSMAXALH, LDSMAXLH: Atomic signed maximum on halfword in memory.
LDSMIN, LDSMINA, LDSMINAL, LDSMINL: Atomic signed minimum on word or doubleword in memo
ry.
LDSMINB, LDSMINAB, LDSMINALB, LDSMINLB: Atomic signed minimum on byte in memory.
LDSMINH, LDSMINAH, LDSMINALH, LDSMINLH: Atomic signed minimum on halfword in memory.
LDTR: Load register (unprivileged).
LDTRB: Load register byte (unprivileged).
LDTRH: Load register halfword (unprivileged).
LDTRSB: Load register signed byte (unprivileged).
LDTRSH: Load register signed halfword (unprivileged).
LDTRSW: Load register signed word (unprivileged).
LDUMAX, LDUMAXA, LDUMAXAL, LDUMAXL: Atomic unsigned maximum on word or doubleword in me
mory.
LDUMAXB, LDUMAXAB, LDUMAXALB, LDUMAXLB: Atomic unsigned maximum on byte in memory.
LDUMAXH, LDUMAXAH, LDUMAXALH, LDUMAXLH: Atomic unsigned maximum on halfword in memory.
LDUMIN, LDUMINA, LDUMINAL, LDUMINL: Atomic unsigned minimum on word or doubleword in me
mory.
LDUMINB, LDUMINAB, LDUMINALB, LDUMINLB: Atomic unsigned minimum on byte in memory.
LDUMINH, LDUMINAH, LDUMINALH, LDUMINLH: Atomic unsigned minimum on halfword in memory.
LDUR: Load register (unscaled).
LDURB: Load register byte (unscaled).
LDURH: Load register halfword (unscaled).
LDURSB: Load register signed byte (unscaled).
LDURSH: Load register signed halfword (unscaled).
LDURSW: Load register signed word (unscaled).
LDXP: Load exclusive pair of registers.
LDXR: Load exclusive register.
LDXRB: Load exclusive register byte.
LDXRH: Load exclusive register halfword.
LSL (immediate): Logical shift left (immediate): an alias of UBFM.
LSL (register): Logical shift left (register): an alias of LSLV.
LSLV: Logical shift left variable.
LSR (immediate): Logical shift right (immediate): an alias of UBFM.
```

```
LSR (register): Logical shift right (register): an alias of LSRV.
LSRV: Logical shift right variable.
MADD: Multiply add.
MADDPT: Multiply add checked pointer.
MNEG: Multiply negate: an alias of MSUB.
MOV (bitmask immediate): Move (bitmask immediate): an alias of ORR (immediate).
MOV (inverted wide immediate): Move (inverted wide immediate): an alias of MOVN.
MOV (register): Move (register): an alias of ORR (shifted register).
MOV (to from SP): Move (to from SP): an alias of ADD (immediate).
MOV (wide immediate): Move (wide immediate): an alias of MOVZ.
MOVK: Move wide with keep.
MOVN: Move wide with NOT.
MOVZ: Move wide with zero.
MRRS: Move System register to two adjacent general purpose registers.
MRS: Move System register to general purpose register.
MSR (immediate): Move immediate value to special register.
MSR (register): Move general purpose register to System register.
MSRR: Move two adjacent general purpose registers to System register.
MSUB: Multiply subtract.
MSUBPT: Multiply subtract checked pointer.
MUL: Multiply: an alias of MADD.
MVN: Bitwise NOT: an alias of ORN (shifted register).
NEG (shifted register): Negate (shifted register): an alias of SUB (shifted register).
NEGS: Negate, setting flags: an alias of SUBS (shifted register).
NGC: Negate with carry: an alias of SBC.
NGCS: Negate with carry, setting flags: an alias of SBCS.
NOP: No operation.
ORN (shifted register): Bitwise OR NOT (shifted register).
ORR (immediate): Bitwise OR (immediate).
ORR (shifted register): Bitwise OR (shifted register).
PACDA, PACDZA: Pointer Authentication Code for data address, using key A.
PACDB, PACDZB: Pointer Authentication Code for data address, using key B.
PACGA: Pointer Authentication Code, using generic key.
PACIA, PACIA1716, PACIASP, PACIAZ, PACIZA: Pointer Authentication Code for instruction
address, using key A.
PACIA171615: Pointer Authentication Code for instruction address, using key A.
PACIASPPC: Pointer Authentication Code for return address, using key A.
PACIB, PACIB1716, PACIBSP, PACIBZ, PACIZB: Pointer Authentication Code for instruction
address, using key B.
PACIB171615: Pointer Authentication Code for instruction address, using key B.
PACIBSPPC: Pointer Authentication Code for return address, using key B.
PACM: Pointer authentication modifier.
PACNBIASPPC: Pointer Authentication Code for return address, using key A, not a branch
target.
PACNBIBSPPC: Pointer Authentication Code for return address, using key B, not a branch
target.
PRFM (immediate): Prefetch memory (immediate).
PRFM (literal): Prefetch memory (literal).
PRFM (register): Prefetch memory (register).
PRFUM: Prefetch memory (unscaled offset).
PSB: Profiling synchronization barrier.
PSSBB: Physical speculative store bypass barrier: an alias of DSB.
RBIT: Reverse bits.
RCWCAS, RCWCASA, RCWCASL, RCWCASAL: Read check write compare and swap doubleword in mem
ory.
RCWCASP, RCWCASPA, RCWCASPL, RCWCASPAL: Read check write compare and swap quadword in m
```

emory.
RCWCLR, RCWCLRA, RCWCLRL, RCWCLRAL: Read check write atomic bit clear on doubleword in memory.
RCWCLRP, RCWCLRPA, RCWCLRPL, RCWCLRPAL: Read check write atomic bit clear on quadword in memory.
RCWSCAS, RCWSCASA, RCWSCASL, RCWSCASAL: Read check write software compare and swap doubleword in memory.
RCWSCASP, RCWSCASPA, RCWSCASPL, RCWSCASPAL: Read check write software compare and swap quadword in memory.
RCWSCLR, RCWSCLRA, RCWSCLRL, RCWSCLRAL: Read check write software atomic bit clear on doubleword in memory.
RCWSCLRP, RCWSCLRPA, RCWSCLRPL, RCWSCLRPAL: Read check write software atomic bit clear on quadword in memory.
RCWSET, RCWSETA, RCWSETL, RCWSETAL: Read check write atomic bit set on doubleword in memory.
RCWSETP, RCWSETPA, RCWSETPL, RCWSETPAL: Read check write atomic bit set on quadword in memory.
RCWSSET, RCWSSETA, RCWSSETL, RCWSSETAL: Read check write software atomic bit set on doubleword in memory.
RCWSSETP, RCWSSETPA, RCWSSETPL, RCWSSETPAL: Read check write software atomic bit set on quadword in memory.
RCWSSWP, RCWSSWPA, RCWSSWPL, RCWSSWPAL: Read check write software swap doubleword in memory.
RCWSSWPP, RCWSSWPPA, RCWSSWPPL, RCWSSWPPAL: Read check write software swap quadword in memory.
RCWSWP, RCWSWPA, RCWSWPL, RCWSWPAL: Read check write swap doubleword in memory.
RCWSWPP, RCWSWPPA, RCWSWPPL, RCWSWPPAL: Read check write swap quadword in memory.
RET: Return from subroutine.
RETAA, RETAB: Return from subroutine, with pointer authentication.
RETAASPPC, RETABSPPC: Return from subroutine, with enhanced pointer authentication return using an immediate offset.
RETAASPPCR, RETABSPPCR: Return from subroutine, with enhanced pointer authentication return using a register.
REV: Reverse bytes.
REV16: Reverse bytes in 16 bit halfwords.
REV32: Reverse bytes in 32 bit words.
REV64: Reverse bytes: an alias of REV.
RMIF: Rotate, mask insert flags.
ROR (immediate): Rotate right (immediate): an alias of EXTR.
ROR (register): Rotate right (register): an alias of RORV.
RORV: Rotate right variable.
RPRFM: Range prefetch memory.
SB: Speculation barrier.
SBC: Subtract with carry.
SBCS: Subtract with carry, setting flags.
SBFIZ: Signed bitfield insert in zeros: an alias of SBFM.
SBFM: Signed bitfield move.
SBFX: Signed bitfield extract: an alias of SBFM.
SDIV: Signed divide.
SETF8, SETF16: Evaluation of 8 bit or 16 bit flag values.
SETGP, SETGM, SETGE: Memory set with tag setting.
SETGPN, SETGMN, SETGEN: Memory set with tag setting, non temporal.
SETGPT, SETGMT, SETGET: Memory set with tag setting, unprivileged.
SETGPTN, SETGMTN, SETGETN: Memory set with tag setting, unprivileged and non temporal.
SETP, SETM, SETE: Memory set.
SETPN, SETMN, SETEN: Memory set, non temporal.

SETPT, SETMT, SETET: Memory set, unprivileged.
SETPTN, SETMTN, SETETN: Memory set, unprivileged and non-temporal.
SEV: Send event.
SEVL: Send event local.
SMADDL: Signed multiply add long.
SMAX (immediate): Signed maximum (immediate).
SMAX (register): Signed maximum (register).
SMC: Secure monitor call.
SMIN (immediate): Signed minimum (immediate).
SMIN (register): Signed minimum (register).
SMNEGL: Signed multiply negate long: an alias of SMSUBL.
SMSTART: Enables access to Streaming SVE mode and SME architectural state: an alias of MSR (immediate).
SMSTOP: Disables access to Streaming SVE mode and SME architectural state: an alias of MSR (immediate).
SMSUBL: Signed multiply subtract long.
SMULH: Signed multiply high.
SMULL: Signed multiply long: an alias of SMADDL.
SSBB: Speculative store bypass barrier: an alias of DSB.
ST2G: Store Allocation Tags.
ST64B: Single copy atomic 64 byte store without status result.
ST64BV: Single copy atomic 64 byte store with status result.
ST64BV0: Single copy atomic 64 byte EL0 store with status result.
STADD, STADDL: Atomic add on word or doubleword in memory, without return: an alias of LDADD, LDADDA, LDADDAL, LDADDL.
STADDB, STADDLB: Atomic add on byte in memory, without return: an alias of LDADDB, LDADDAB, LDADDALB, LDADDLB.
STADDH, STADDLH: Atomic add on halfword in memory, without return: an alias of LDADDH, LDADDAH, LDADDALH, LDADDLH.
STCLR, STCLRL: Atomic bit clear on word or doubleword in memory, without return: an alias of LDCLR, LDCLRA, LDCLRAL, LDCLRL.
STCLRB, STCLRLB: Atomic bit clear on byte in memory, without return: an alias of LDCLRB, LDCLRAB, LDCLRALB, LDCLRLB.
STCLRH, STCLRLH: Atomic bit clear on halfword in memory, without return: an alias of LDCLRH, LDCLRAH, LDCLRALH, LDCLRLH.
STEOR, STEORL: Atomic exclusive OR on word or doubleword in memory, without return: an alias of LDEOR, LDEORA, LDEORAL, LDEORL.
STEORB, STEORLB: Atomic exclusive OR on byte in memory, without return: an alias of LDEORB, LDEORAB, LDEORALB, LDEORLB.
STEORH, STEORLH: Atomic exclusive OR on halfword in memory, without return: an alias of LDEORH, LDEORAH, LDEORALH, LDEORLH.
STG: Store Allocation Tag.
STGM: Store Allocation Tag multiple.
STGP: Store Allocation Tag and pair of registers.
STILP: Store release ordered pair of registers.
STLLR: Store LORelease register.
STLLRB: Store LORelease register byte.
STLLRH: Store LORelease register halfword.
STLR: Store release register.
STLRB: Store release register byte.
STLRH: Store release register halfword.
STLUR: Store release register (unscaled).
STLURB: Store release register byte (unscaled).
STLURH: Store release register halfword (unscaled).
STLXP: Store release exclusive pair of registers.
STLXR: Store release exclusive register.

STLXRB: Store release exclusive register byte.
STLXRH: Store release exclusive register halfword.
STNP: Store pair of registers, with non temporal hint.
STP: Store pair of registers.
STR (immediate): Store register (immediate).
STR (register): Store register (register).
STRB (immediate): Store register byte (immediate).
STRB (register): Store register byte (register).
STRH (immediate): Store register halfword (immediate).
STRH (register): Store register halfword (register).
STSET, STSETL: Atomic bit set on word or doubleword in memory, without return: an alias of LDSET, LDSETA, LDSETAL, LDSETL.
STSETB, STSETLB: Atomic bit set on byte in memory, without return: an alias of LDSETB, LDSETAB, LDSETALB, LDSETLB.
STSETH, STSETLH: Atomic bit set on halfword in memory, without return: an alias of LDSETH, LDSETAH, LDSETALH, LDSETLH.
STSMAX, STSMAXL: Atomic signed maximum on word or doubleword in memory, without return: an alias of LDSMAX, LDSMAXA, LDSMAXAL, LDSMAXL.
STSMAXB, STSMAXLB: Atomic signed maximum on byte in memory, without return: an alias of LDSMAXB, LDSMAXAB, LDSMAXALB, LDSMAXLB.
STSMAXH, STSMAXLH: Atomic signed maximum on halfword in memory, without return: an alias of LDSMAXH, LDSMAXAH, LDSMAXALH, LDSMAXLH.
STSMIN, STSMINL: Atomic signed minimum on word or doubleword in memory, without return: an alias of LDSMIN, LDSMINA, LDSMINAL, LDSMINL.
STSMINB, STSMINLB: Atomic signed minimum on byte in memory, without return: an alias of LDSMINB, LDSMINAB, LDSMINALB, LDSMINLB.
STSMINH, STSMINLH: Atomic signed minimum on halfword in memory, without return: an alias of LDSMINH, LDSMINAH, LDSMINALH, LDSMINLH.
STTR: Store register (unprivileged).
STTRB: Store register byte (unprivileged).
STTRH: Store register halfword (unprivileged).
STUMAX, STUMAXL: Atomic unsigned maximum on word or doubleword in memory, without return: an alias of LDUMAX, LDUMAXA, LDUMAXAL, LDUMAXL.
STUMAXB, STUMAXLB: Atomic unsigned maximum on byte in memory, without return: an alias of LDUMAXB, LDUMAXAB, LDUMAXALB, LDUMAXLB.
STUMAXH, STUMAXLH: Atomic unsigned maximum on halfword in memory, without return: an alias of LDUMAXH, LDUMAXAH, LDUMAXALH, LDUMAXLH.
STUMIN, STUMINL: Atomic unsigned minimum on word or doubleword in memory, without return: an alias of LDUMIN, LDUMINA, LDUMINAL, LDUMINL.
STUMINB, STUMINLB: Atomic unsigned minimum on byte in memory, without return: an alias of LDUMINB, LDUMINAB, LDUMINALB, LDUMINLB.
STUMINH, STUMINLH: Atomic unsigned minimum on halfword in memory, without return: an alias of LDUMINH, LDUMINAH, LDUMINALH, LDUMINLH.
STUR: Store register (unscaled).
STURB: Store register byte (unscaled).
STURH: Store register halfword (unscaled).
STXP: Store exclusive pair of registers.
STXR: Store exclusive register.
STXRB: Store exclusive register byte.
STXRH: Store exclusive register halfword.
STZ2G: Store Allocation Tags, zeroing.
STZG: Store Allocation Tag, zeroing.
STZGM: Store Allocation Tag and zero multiple.
SUB (extended register): Subtract (extended register).
SUB (immediate): Subtract (immediate).
SUB (shifted register): Subtract (shifted register).

SUBG: Subtract with tag.
SUBP: Subtract pointer.
SUBPS: Subtract pointer, setting flags.
SUBPT: Subtract checked pointer.
SUBS (extended register): Subtract (extended register), setting flags.
SUBS (immediate): Subtract (immediate), setting flags.
SUBS (shifted register): Subtract (shifted register), setting flags.
SVC: Supervisor call.
SWP, SWPA, SWPAL, SWPL: Swap word or doubleword in memory.
SWPB, SWPAB, SWPALB, SWPLB: Swap byte in memory.
SWPH, SWPAH, SWPALH, SWPLH: Swap halfword in memory.
SWPP, SWPPA, SWPPAL, SWPPL: Swap quadword in memory.
SXTB: Signed extend byte: an alias of SBFM.
SXTH: Sign extend halfword: an alias of SBFM.
SXTW: Sign extend word: an alias of SBFM.
SYS: System instruction.
SYSL: System instruction with result.
SYSP: 128 bit system instruction.
TBNZ: Test bit and branch if nonzero.
TBZ: Test bit and branch if zero.
TCANCEL: Cancel current transaction.
TCOMMIT: Commit current transaction.
TLBI: TLB invalidate operation: an alias of SYS.
TLBIP: TLB invalidate pair operation: an alias of SYSP.
TRCIT: Trace instrumentation: an alias of SYS.
TSB: Trace synchronization barrier.
TST (immediate): Test bits (immediate): an alias of ANDS (immediate).
TST (shifted register): Test (shifted register): an alias of ANDS (shifted register).
TSTART: Start transaction.
TTEST: Test transaction state.
UBFIZ: Unsigned bitfield insert in zeros: an alias of UBFM.
UBFM: Unsigned bitfield move.
UBFX: Unsigned bitfield extract: an alias of UBFM.
UDF: Permanently undefined.
UDIV: Unsigned divide.
UMADDL: Unsigned multiply add long.
UMAX (immediate): Unsigned maximum (immediate).
UMAX (register): Unsigned maximum (register).
UMIN (immediate): Unsigned minimum (immediate).
UMIN (register): Unsigned minimum (register).
UMNEGL: Unsigned multiply negate long: an alias of UMSUBL.
UMSUBL: Unsigned multiply subtract long.
UMULH: Unsigned multiply high.
UMULL: Unsigned multiply long: an alias of UMADDL.
UXTB: Unsigned extend byte: an alias of UBFM.
UXTH: Unsigned extend halfword: an alias of UBFM.
WFE: Wait for event.
WFET: Wait for event with timeout.
WFI: Wait for interrupt.
WFIT: Wait for interrupt with timeout.
XAFLAG: Convert floating point condition flags from external format to Arm format.
XPACD, XPACI, XPACLRI: Strip Pointer Authentication Code.
YIELD: Yield.

A64 -- SIMD and Floating-point Instructions (alphabetic order)

ABS: Absolute value (vector).
ADD (vector): Add (vector).
ADDHN, ADDHN2: Add returning high narrow.
ADDP (scalar): Add pair of elements (scalar).
ADDP (vector): Add pairwise (vector).
ADDV: Add across vector.
AESD: AES single round decryption.
AESE: AES single round encryption.
AESIMC: AES inverse mix columns.
AESMC: AES mix columns.
AND (vector): Bitwise AND (vector).
BCAX: Bit clear and exclusive OR.
BF1CVTL, BF1CVTL2, BF2CVTL, BF2CVTL2: 8 bit floating point convert to BFloat16 (vector)
.
BFCVT: Floating point convert from single precision to BFloat16 format (scalar).
BFCVTN, BFCVTN2: Floating point convert from single precision to BFloat16 format (vecto
r).
BFDOT (by element): BFloat16 floating point dot product (vector, by element).
BFDOT (vector): BFloat16 floating point dot product (vector).
BFMLALB, BFMLALT (by element): BFloat16 floating point widening multiply add long (by e
lement).
BFMLALB, BFMLALT (vector): BFloat16 floating point widening multiply add long (vector).
BFMMLA: BFloat16 floating point matrix multiply accumulate into 2x2 matrix.
BIC (vector, immediate): Bitwise bit clear (vector, immediate).
BIC (vector, register): Bitwise bit clear (vector, register).
BIF: Bitwise insert if false.
BIT: Bitwise insert if true.
BSL: Bitwise select.
CLS (vector): Count leading sign bits (vector).
CLZ (vector): Count leading zero bits (vector).
CMEQ (register): Compare bitwise equal (vector).
CMEQ (zero): Compare bitwise equal to zero (vector).
CMGE (register): Compare signed greater than or equal (vector).
CMGE (zero): Compare signed greater than or equal to zero (vector).
CMGT (register): Compare signed greater than (vector).
CMGT (zero): Compare signed greater than zero (vector).
CMHI (register): Compare unsigned higher (vector).
CMHS (register): Compare unsigned higher or same (vector).
CMLE (zero): Compare signed less than or equal to zero (vector).
CMLT (zero): Compare signed less than zero (vector).
CMTST: Compare bitwise test bits nonzero (vector).
CNT: Population count per byte.
DUP (element): Duplicate vector element to vector or scalar.
DUP (general): Duplicate general purpose register to vector.
EOR (vector): Bitwise exclusive OR (vector).
EOR3: Three way exclusive OR.
EXT: Extract vector from pair of vectors.
F1CVTL, F1CVTL2, F2CVTL, F2CVTL2: 8 bit floating point convert to half precision (vecto
r).
FABD: Floating point absolute difference (vector).
FABS (scalar): Floating point absolute value (scalar).
FABS (vector): Floating point absolute value (vector).
FACGE: Floating point absolute compare greater than or equal (vector).

```
FACGT: Floating point absolute compare greater than (vector).
FADD (scalar): Floating point add (scalar).
FADD (vector): Floating point add (vector).
FADDP (scalar): Floating point add pair of elements (scalar).
FADDP (vector): Floating point add pairwise (vector).
FAMAX: Floating point absolute maximum.
FAMIN: Floating point absolute minimum.
FCADD: Floating point complex add.
FCCMP: Floating point conditional quiet compare (scalar).
FCCMPE: Floating point conditional signaling compare (scalar).
FCMEQ (register): Floating point compare equal (vector).
FCMEQ (zero): Floating point compare equal to zero (vector).
FCMGE (register): Floating point compare greater than or equal (vector).
FCMGE (zero): Floating point compare greater than or equal to zero (vector).
FCMGT (register): Floating point compare greater than (vector).
FCMGT (zero): Floating point compare greater than zero (vector).
FCMLA: Floating point complex multiply accumulate.
FCMLA (by element): Floating point complex multiply accumulate (by element).
FCMLE (zero): Floating point compare less than or equal to zero (vector).
FCMLT (zero): Floating point compare less than zero (vector).
FCMP: Floating point quiet compare (scalar).
FCMPE: Floating point signaling compare (scalar).
FCSEL: Floating point conditional select (scalar).
FCVT: Floating point convert precision (scalar).
FCVTAS (scalar): Floating point convert to signed integer, rounding to nearest with tie
s to away (scalar).
FCVTAS (vector): Floating point convert to signed integer, rounding to nearest with tie
s to away (vector).
FCVTAU (scalar): Floating point convert to unsigned integer, rounding to nearest with t
ies to away (scalar).
FCVTAU (vector): Floating point convert to unsigned integer, rounding to nearest with t
ies to away (vector).
FCVTL, FCVTL2: Floating point convert to higher precision long (vector).
FCVTMS (scalar): Floating point convert to signed integer, rounding toward minus infini
ty (scalar).
FCVTMS (vector): Floating point convert to signed integer, rounding toward minus infini
ty (vector).
FCVTMU (scalar): Floating point convert to unsigned integer, rounding toward minus infi
nity (scalar).
FCVTMU (vector): Floating point convert to unsigned integer, rounding toward minus infi
nity (vector).
FCVTN (half precision to 8 bit floating point): Half precision to 8 bit floating point
convert and narrow (vector).
FCVTN, FCVTN2 (double to single precision, single to half precision): Floating point co
nvert to lower precision narrow (vector).
FCVTN, FCVTN2 (single precision to 8 bit floating point): Single precision to 8 bit flo
ating point convert and narrow (vector).
FCVTNS (scalar): Floating point convert to signed integer, rounding to nearest with tie
s to even (scalar).
FCVTNS (vector): Floating point convert to signed integer, rounding to nearest with tie
s to even (vector).
FCVTNU (scalar): Floating point convert to unsigned integer, rounding to nearest with t
ies to even (scalar).
FCVTNU (vector): Floating point convert to unsigned integer, rounding to nearest with t
ies to even (vector).
FCVTPS (scalar): Floating point convert to signed integer, rounding toward plus infinit
```

```
y (scalar).
FCVTPS (vector): Floating point convert to signed integer, rounding toward plus infinit
y (vector).
FCVTPU (scalar): Floating point convert to unsigned integer, rounding toward plus infin
ity (scalar).
FCVTPU (vector): Floating point convert to unsigned integer, rounding toward plus infin
ity (vector).
FCVTXN, FCVTXN2: Floating point convert to lower precision narrow, rounding to odd (vec
tor).
FCVTZS (scalar, fixed point): Floating point convert to signed fixed point, rounding to
ward zero (scalar).
FCVTZS (scalar, integer): Floating point convert to signed integer, rounding toward zer
o (scalar).
FCVTZS (vector, fixed point): Floating point convert to signed fixed point, rounding to
ward zero (vector).
FCVTZS (vector, integer): Floating point convert to signed integer, rounding toward zer
o (vector).
FCVTZU (scalar, fixed point): Floating point convert to unsigned fixed point, rounding
toward zero (scalar).
FCVTZU (scalar, integer): Floating point convert to unsigned integer, rounding toward z
ero (scalar).
FCVTZU (vector, fixed point): Floating point convert to unsigned fixed point, rounding
toward zero (vector).
FCVTZU (vector, integer): Floating point convert to unsigned integer, rounding toward z
ero (vector).
FDIV (scalar): Floating point divide (scalar).
FDIV (vector): Floating point divide (vector).
FDOT (8 bit floating point to half precision, by element): 8 bit floating point dot pro
duct to half precision (vector, by element).
FDOT (8 bit floating point to half precision, vector): 8 bit floating point dot product
 to half precision (vector).
FDOT (8 bit floating point to single precision, by element): 8 bit floating point dot p
roduct to single precision (vector, by element).
FDOT (8 bit floating point to single precision, vector): 8 bit floating point dot produ
ct to single precision (vector).
FJCVTZS: Floating point Javascript convert to signed fixed point, rounding toward zero.
FMADD: Floating point fused multiply add (scalar).
FMAX (scalar): Floating point maximum (scalar).
FMAX (vector): Floating point maximum (vector).
FMAXNM (scalar): Floating point maximum number (scalar).
FMAXNM (vector): Floating point maximum number (vector).
FMAXNMP (scalar): Floating point maximum number of pair of elements (scalar).
FMAXNMP (vector): Floating point maximum number pairwise (vector).
FMAXNMV: Floating point maximum number across vector.
FMAXP (scalar): Floating point maximum of pair of elements (scalar).
FMAXP (vector): Floating point maximum pairwise (vector).
FMAXV: Floating point maximum across vector.
FMIN (scalar): Floating point minimum (scalar).
FMIN (vector): Floating point minimum (vector).
FMINNM (scalar): Floating point minimum number (scalar).
FMINNM (vector): Floating point minimum number (vector).
FMINNMP (scalar): Floating point minimum number of pair of elements (scalar).
FMINNMP (vector): Floating point minimum number pairwise (vector).
FMINNMV: Floating point minimum number across vector.
FMINP (scalar): Floating point minimum of pair of elements (scalar).
FMINP (vector): Floating point minimum pairwise (vector).
```

FMINV: Floating point minimum across vector.
FMLA (by element): Floating point fused multiply add to accumulator (by element).
FMLA (vector): Floating point fused multiply add to accumulator (vector).
FMLAL, FMLAL2 (by element): Floating point fused multiply add long to accumulator (by element).
FMLAL, FMLAL2 (vector): Floating point fused multiply add long to accumulator (vector).
FMLALB, FMLALT (by element): 8 bit floating point multiply add long to half precision (vector, by element).
FMLALB, FMLALT (vector): 8 bit floating point multiply add long to half precision (vector).
FMLALLBB, FMLALLBT, FMLALLTB, FMLALLTT (by element): 8 bit floating point multiply add long long to single precision (vector, by element).
FMLALLBB, FMLALLBT, FMLALLTB, FMLALLTT (vector): 8 bit floating point multiply add long long to single precision (vector).
FMLS (by element): Floating point fused multiply subtract from accumulator (by element).
FMLS (vector): Floating point fused multiply subtract from accumulator (vector).
FMLSL, FMLSL2 (by element): Floating point fused multiply subtract long from accumulator (by element).
FMLSL, FMLSL2 (vector): Floating point fused multiply subtract long from accumulator (vector).
FMOV (general): Floating point move to or from general purpose register without conversion.
FMOV (register): Floating point move register without conversion.
FMOV (scalar, immediate): Floating point move immediate (scalar).
FMOV (vector, immediate): Floating point move immediate (vector).
FMSUB: Floating point fused multiply subtract (scalar).
FMUL (by element): Floating point multiply (by element).
FMUL (scalar): Floating point multiply (scalar).
FMUL (vector): Floating point multiply (vector).
FMULX: Floating point multiply extended.
FMULX (by element): Floating point multiply extended (by element).
FNEG (scalar): Floating point negate (scalar).
FNEG (vector): Floating point negate (vector).
FNMADD: Floating point negated fused multiply add (scalar).
FNMSUB: Floating point negated fused multiply subtract (scalar).
FNMUL (scalar): Floating point multiply negate (scalar).
FRECPE: Floating point reciprocal estimate.
FRECPS: Floating point reciprocal step.
FRECPX: Floating point reciprocal exponent (scalar).
FRINT32X (scalar): Floating point round to 32 bit integer, using current rounding mode (scalar).
FRINT32X (vector): Floating point round to 32 bit integer, using current rounding mode (vector).
FRINT32Z (scalar): Floating point round to 32 bit integer toward zero (scalar).
FRINT32Z (vector): Floating point round to 32 bit integer toward zero (vector).
FRINT64X (scalar): Floating point round to 64 bit integer, using current rounding mode (scalar).
FRINT64X (vector): Floating point round to 64 bit integer, using current rounding mode (vector).
FRINT64Z (scalar): Floating point round to 64 bit integer toward zero (scalar).
FRINT64Z (vector): Floating point round to 64 bit integer toward zero (vector).
FRINTA (scalar): Floating point round to integral, to nearest with ties to away (scalar).
FRINTA (vector): Floating point round to integral, to nearest with ties to away (vector).

```
FRINTI (scalar): Floating point round to integral, using current rounding mode (scalar)
.
FRINTI (vector): Floating point round to integral, using current rounding mode (vector)
.
FRINTM (scalar): Floating point round to integral, toward minus infinity (scalar).
FRINTM (vector): Floating point round to integral, toward minus infinity (vector).
FRINTN (scalar): Floating point round to integral, to nearest with ties to even (scalar
).
FRINTN (vector): Floating point round to integral, to nearest with ties to even (vector
).
FRINTP (scalar): Floating point round to integral, toward plus infinity (scalar).
FRINTP (vector): Floating point round to integral, toward plus infinity (vector).
FRINTX (scalar): Floating point round to integral exact, using current rounding mode (s
calar).
FRINTX (vector): Floating point round to integral exact, using current rounding mode (v
ector).
FRINTZ (scalar): Floating point round to integral, toward zero (scalar).
FRINTZ (vector): Floating point round to integral, toward zero (vector).
FRSQRTE: Floating point reciprocal square root estimate.
FRSQRTS: Floating point reciprocal square root step.
FSCALE: Floating point adjust exponent by vector.
FSQRT (scalar): Floating point square root (scalar).
FSQRT (vector): Floating point square root (vector).
FSUB (scalar): Floating point subtract (scalar).
FSUB (vector): Floating point subtract (vector).
INS (element): Insert vector element from another vector element.
INS (general): Insert vector element from general purpose register.
LD1 (multiple structures): Load multiple single element structures to one, two, three,
or four registers.
LD1 (single structure): Load one single element structure to one lane of one register.
LD1R: Load one single element structure and replicate to all lanes (of one register).
LD2 (multiple structures): Load multiple 2 element structures to two registers.
LD2 (single structure): Load single 2 element structure to one lane of two registers.
LD2R: Load single 2 element structure and replicate to all lanes of two registers.
LD3 (multiple structures): Load multiple 3 element structures to three registers.
LD3 (single structure): Load single 3 element structure to one lane of three registers.
LD3R: Load single 3 element structure and replicate to all lanes of three registers.
LD4 (multiple structures): Load multiple 4 element structures to four registers.
LD4 (single structure): Load single 4 element structure to one lane of four registers.
LD4R: Load single 4 element structure and replicate to all lanes of four registers.
LDAP1 (SIMD FP): Load acquire RCpc one single element structure to one lane of one regi
ster.
LDAPUR (SIMD FP): Load acquire RCpc SIMD FP register (unscaled offset).
LDNP (SIMD FP): Load pair of SIMD FP registers, with non temporal hint.
LDP (SIMD FP): Load pair of SIMD FP registers.
LDR (immediate, SIMD FP): Load SIMD FP register (immediate offset).
LDR (literal, SIMD FP): Load SIMD FP register (PC relative literal).
LDR (register, SIMD FP): Load SIMD FP register (register offset).
LDUR (SIMD FP): Load SIMD FP register (unscaled offset).
LUTI2: Lookup table read with 2 bit indices.
LUTI4: Lookup table read with 4 bit indices.
MLA (by element): Multiply add to accumulator (vector, by element).
MLA (vector): Multiply add to accumulator (vector).
MLS (by element): Multiply subtract from accumulator (vector, by element).
MLS (vector): Multiply subtract from accumulator (vector).
MOV (element): Move vector element to another vector element: an alias of INS (element)
```

```
.
MOV (from general): Move general purpose register to a vector element: an alias of INS
(general).
MOV (scalar): Move vector element to scalar: an alias of DUP (element).
MOV (to general): Move vector element to general purpose register: an alias of UMOV.
MOV (vector): Move vector: an alias of ORR (vector, register).
MOVI: Move immediate (vector).
MUL (by element): Multiply (vector, by element).
MUL (vector): Multiply (vector).
MVN: Bitwise NOT (vector): an alias of NOT.
MVNI: Move inverted immediate (vector).
NEG (vector): Negate (vector).
NOT: Bitwise NOT (vector).
ORN (vector): Bitwise inclusive OR NOT (vector).
ORR (vector, immediate): Bitwise inclusive OR (vector, immediate).
ORR (vector, register): Bitwise inclusive OR (vector, register).
PMUL: Polynomial multiply.
PMULL, PMULL2: Polynomial multiply long.
RADDHN, RADDHN2: Rounding add returning high narrow.
RAX1: Rotate and exclusive OR.
RBIT (vector): Reverse bit order (vector).
REV16 (vector): Reverse elements in 16 bit halfwords (vector).
REV32 (vector): Reverse elements in 32 bit words (vector).
REV64: Reverse elements in 64 bit doublewords (vector).
RSHRN, RSHRN2: Rounding shift right narrow (immediate).
RSUBHN, RSUBHN2: Rounding subtract returning high narrow.
SABA: Signed absolute difference and accumulate.
SABAL, SABAL2: Signed absolute difference and accumulate long.
SABD: Signed absolute difference.
SABDL, SABDL2: Signed absolute difference long.
SADALP: Signed add and accumulate long pairwise.
SADDL, SADDL2: Signed add long (vector).
SADDLP: Signed add long pairwise.
SADDLV: Signed add long across vector.
SADDW, SADDW2: Signed add wide.
SCVTF (scalar, fixed point): Signed fixed point convert to floating point (scalar).
SCVTF (scalar, integer): Signed integer convert to floating point (scalar).
SCVTF (vector, fixed point): Signed fixed point convert to floating point (vector).
SCVTF (vector, integer): Signed integer convert to floating point (vector).
SDOT (by element): Dot product signed arithmetic (vector, by element).
SDOT (vector): Dot product signed arithmetic (vector).
SHA1C: SHA1 hash update (choose).
SHA1H: SHA1 fixed rotate.
SHA1M: SHA1 hash update (majority).
SHA1P: SHA1 hash update (parity).
SHA1SU0: SHA1 schedule update 0.
SHA1SU1: SHA1 schedule update 1.
SHA256H: SHA256 hash update (part 1).
SHA256H2: SHA256 hash update (part 2).
SHA256SU0: SHA256 schedule update 0.
SHA256SU1: SHA256 schedule update 1.
SHA512H: SHA512 hash update part 1.
SHA512H2: SHA512 hash update part 2.
SHA512SU0: SHA512 schedule update 0.
SHA512SU1: SHA512 schedule update 1.
SHADD: Signed halving add.
```

SHL: Shift left (immediate).
SHLL, SHLL2: Shift left long (by element size).
SHRN, SHRN2: Shift right narrow (immediate).
SHSUB: Signed halving subtract.
SLI: Shift left and insert (immediate).
SM3PARTW1: SM3PARTW1.
SM3PARTW2: SM3PARTW2.
SM3SS1: SM3SS1.
SM3TT1A: SM3TT1A.
SM3TT1B: SM3TT1B.
SM3TT2A: SM3TT2A.
SM3TT2B: SM3TT2B.
SM4E: SM4 encode.
SM4EKEY: SM4 key.
SMAX: Signed maximum (vector).
SMAXP: Signed maximum pairwise.
SMAXV: Signed maximum across vector.
SMIN: Signed minimum (vector).
SMINP: Signed minimum pairwise.
SMINV: Signed minimum across vector.
SMLAL, SMLAL2 (by element): Signed multiply add long (vector, by element).
SMLAL, SMLAL2 (vector): Signed multiply add long (vector).
SMLSL, SMLSL2 (by element): Signed multiply subtract long (vector, by element).
SMLSL, SMLSL2 (vector): Signed multiply subtract long (vector).
SMMLA (vector): Signed 8 bit integer matrix multiply accumulate (vector).
SMOV: Signed move vector element to general purpose register.
SMULL, SMULL2 (by element): Signed multiply long (vector, by element).
SMULL, SMULL2 (vector): Signed multiply long (vector).
SQABS: Signed saturating absolute value.
SQADD: Signed saturating add.
SQDMLAL, SQDMLAL2 (by element): Signed saturating doubling multiply add long (by elemen
t).
SQDMLAL, SQDMLAL2 (vector): Signed saturating doubling multiply add long.
SQDMLSL, SQDMLSL2 (by element): Signed saturating doubling multiply subtract long (by e
lement).
SQDMLSL, SQDMLSL2 (vector): Signed saturating doubling multiply subtract long.
SQDMULH (by element): Signed saturating doubling multiply returning high half (by eleme
nt).
SQDMULH (vector): Signed saturating doubling multiply returning high half.
SQDMULL, SQDMULL2 (by element): Signed saturating doubling multiply long (by element).
SQDMULL, SQDMULL2 (vector): Signed saturating doubling multiply long.
SQNEG: Signed saturating negate.
SQRDMLAH (by element): Signed saturating rounding doubling multiply accumulate returnin
g high half (by element).
SQRDMLAH (vector): Signed saturating rounding doubling multiply accumulate returning hi
gh half (vector).
SQRDMLSH (by element): Signed saturating rounding doubling multiply subtract returning
high half (by element).
SQRDMLSH (vector): Signed saturating rounding doubling multiply subtract returning high
 half (vector).
SQRDMULH (by element): Signed saturating rounding doubling multiply returning high half
 (by element).
SQRDMULH (vector): Signed saturating rounding doubling multiply returning high half.
SQRSHL: Signed saturating rounding shift left (register).
SQRSHRN, SQRSHRN2: Signed saturating rounded shift right narrow (immediate).
SQRSHRUN, SQRSHRUN2: Signed saturating rounded shift right unsigned narrow (immediate).

```
SQSHL (immediate): Signed saturating shift left (immediate).
SQSHL (register): Signed saturating shift left (register).
SQSHLU: Signed saturating shift left unsigned (immediate).
SQSHRN, SQSHRN2: Signed saturating shift right narrow (immediate).
SQSHRUN, SQSHRUN2: Signed saturating shift right unsigned narrow (immediate).
SQSUB: Signed saturating subtract.
SQXTN, SQXTN2: Signed saturating extract narrow.
SQXTUN, SQXTUN2: Signed saturating extract unsigned narrow.
SRHADD: Signed rounding halving add.
SRI: Shift right and insert (immediate).
SRSHL: Signed rounding shift left (register).
SRSHR: Signed rounding shift right (immediate).
SRSRA: Signed rounding shift right and accumulate (immediate).
SSHL: Signed shift left (register).
SSHLL, SSHLL2: Signed shift left long (immediate).
SSHR: Signed shift right (immediate).
SSRA: Signed shift right and accumulate (immediate).
SSUBL, SSUBL2: Signed subtract long.
SSUBW, SSUBW2: Signed subtract wide.
ST1 (multiple structures): Store multiple single element structures from one, two, thre
e, or four registers.
ST1 (single structure): Store a single element structure from one lane of one register.
ST2 (multiple structures): Store multiple 2 element structures from two registers.
ST2 (single structure): Store single 2 element structure from one lane of two registers
.
ST3 (multiple structures): Store multiple 3 element structures from three registers.
ST3 (single structure): Store single 3 element structure from one lane of three registe
rs.
ST4 (multiple structures): Store multiple 4 element structures from four registers.
ST4 (single structure): Store single 4 element structure from one lane of four register
s.
STL1 (SIMD FP): Store release a single element structure from one lane of one register.
STLUR (SIMD FP): Store release SIMD FP register (unscaled offset).
STNP (SIMD FP): Store pair of SIMD FP registers, with non temporal hint.
STP (SIMD FP): Store pair of SIMD FP registers.
STR (immediate, SIMD FP): Store SIMD FP register (immediate offset).
STR (register, SIMD FP): Store SIMD FP register (register offset).
STUR (SIMD FP): Store SIMD FP register (unscaled offset).
SUB (vector): Subtract (vector).
SUBHN, SUBHN2: Subtract returning high narrow.
SUDOT (by element): Dot product with signed and unsigned integers (vector, by element).
SUQADD: Signed saturating accumulate of unsigned value.
SXTL, SXTL2: Signed extend long: an alias of SSHLL, SSHLL2.
TBL: Table vector lookup.
TBX: Table vector lookup extension.
TRN1: Transpose vectors (primary).
TRN2: Transpose vectors (secondary).
UABA: Unsigned absolute difference and accumulate.
UABAL, UABAL2: Unsigned absolute difference and accumulate long.
UABD: Unsigned absolute difference (vector).
UABDL, UABDL2: Unsigned absolute difference long.
UADALP: Unsigned add and accumulate long pairwise.
UADDL, UADDL2: Unsigned add long (vector).
UADDLP: Unsigned add long pairwise.
UADDLV: Unsigned sum long across vector.
UADDW, UADDW2: Unsigned add wide.
```

```
UCVTF (scalar, fixed point): Unsigned fixed point convert to floating point (scalar).
UCVTF (scalar, integer): Unsigned integer convert to floating point (scalar).
UCVTF (vector, fixed point): Unsigned fixed point convert to floating point (vector).
UCVTF (vector, integer): Unsigned integer convert to floating point (vector).
UDOT (by element): Dot product unsigned arithmetic (vector, by element).
UDOT (vector): Dot product unsigned arithmetic (vector).
UHADD: Unsigned halving add.
UHSUB: Unsigned halving subtract.
UMAX: Unsigned maximum (vector).
UMAXP: Unsigned maximum pairwise.
UMAXV: Unsigned maximum across vector.
UMIN: Unsigned minimum (vector).
UMINP: Unsigned minimum pairwise.
UMINV: Unsigned minimum across vector.
UMLAL, UMLAL2 (by element): Unsigned multiply add long (vector, by element).
UMLAL, UMLAL2 (vector): Unsigned multiply add long (vector).
UMLSL, UMLSL2 (by element): Unsigned multiply subtract long (vector, by element).
UMLSL, UMLSL2 (vector): Unsigned multiply subtract long (vector).
UMMLA (vector): Unsigned 8 bit integer matrix multiply accumulate (vector).
UMOV: Unsigned move vector element to general purpose register.
UMULL, UMULL2 (by element): Unsigned multiply long (vector, by element).
UMULL, UMULL2 (vector): Unsigned multiply long (vector).
UQADD: Unsigned saturating add.
UQRSHL: Unsigned saturating rounding shift left (register).
UQRSHRN, UQRSHRN2: Unsigned saturating rounded shift right narrow (immediate).
UQSHL (immediate): Unsigned saturating shift left (immediate).
UQSHL (register): Unsigned saturating shift left (register).
UQSHRN, UQSHRN2: Unsigned saturating shift right narrow (immediate).
UQSUB: Unsigned saturating subtract.
UQXTN, UQXTN2: Unsigned saturating extract narrow.
URECPE: Unsigned reciprocal estimate.
URHADD: Unsigned rounding halving add.
URSHL: Unsigned rounding shift left (register).
URSHR: Unsigned rounding shift right (immediate).
URSQRTE: Unsigned reciprocal square root estimate.
URSRA: Unsigned rounding shift right and accumulate (immediate).
USDOT (by element): Dot product with unsigned and signed integers (vector, by element).
USDOT (vector): Dot product with unsigned and signed integers (vector).
USHL: Unsigned shift left (register).
USHLL, USHLL2: Unsigned shift left long (immediate).
USHR: Unsigned shift right (immediate).
USMMLA (vector): Unsigned and signed 8 bit integer matrix multiply accumulate (vector).
USQADD: Unsigned saturating accumulate of signed value.
USRA: Unsigned shift right and accumulate (immediate).
USUBL, USUBL2: Unsigned subtract long.
USUBW, USUBW2: Unsigned subtract wide.
UXTL, UXTL2: Unsigned extend long: an alias of USHLL, USHLL2.
UZP1: Unzip vectors (primary).
UZP2: Unzip vectors (secondary).
XAR: Exclusive OR and rotate.
XTN, XTN2: Extract narrow.
ZIP1: Zip vectors (primary).
ZIP2: Zip vectors (secondary).
```

带操作码的ARM汇编指令列表

- 来源
  - ARM (groupoid.github.io)

操作码格式介绍

```
INTRO OPCODES

 +--------------------= 32/64 bit
 |
 |
 |      +-+-+-+---------= Opcode
 |      0 0 0 0 --------= UDF
 |      0 0 0 1 --------= N A
 |      0 0 1 x --------= N A
 |      1 0 0 x --------= Data Processing Immediate
 |      1 0 1 x --------= Branching, System, Exceptions
 |      x 1 x 0 --------= Loads and Stores
 |      x 1 0 1 --------= Data Processing Register
 |      x 1 1 1 --------= Data Processing Scalar FPU and SIMD
 |      | | | |
 |      | | | |
 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
 +--------------+ +--------------+ +--------------+ +--------------+
 |      MSB     | |              | |              | |      LSB     |
 +--------------+ +--------------+ +--------------+ +--------------+
```

普通指令

```
0 0 0 0 0 0 0 0 0 0 0 % C6.2.312 UDF
0 x 0 0 1 0 0 0 0 0 0 % C6.2.286 STXRB % C6.2.253 STLXRB % C6.2.254 STLXRH
0 x 0 0 1 0 0 0 0 i 1 % C6.2.039 CASP, CASPA, CASPAL, CASPL
0 x 0 0 1 0 0 0 0 1 0 % C6.2.106 LDAXRH % C6.2.105 LDAXRB % C6.2.163 LDXRB
0 0 0 0 1 0 0 0 1 0 0 % C6.2.246 STLRB
0 0 0 0 1 0 0 0 1 1 0 % C6.2.101 LDARB % C6.2.113 LDLARB
0 0 0 0 1 0 0 0 1 0 0 % C6.2.242 STLLRB % C6.2.243 STLLRH
0 0 0 0 1 0 0 0 1 i 1 % C6.2.037 CASB, CASAB, CASALB, CASLB
0 1 0 0 1 0 0 0 0 1 0 % C6.2.164 LDXRH
0 1 0 0 1 0 0 0 1 0 0 % C6.2.247 STLRH
0 1 0 0 1 0 0 0 1 1 0 % C6.2.102 LDARH % C6.2.114 LDLARH
0 1 0 0 1 0 0 0 1 i 1 % C6.2.038 CASH, CASAH, CASALH, CASLH
0 1 1 0 1 0 0 i i 1 _ % C6.2.118 LDPSW
x 0 1 0 1 0 0 0 0 0 _ % C6.2.255 STNP
x 0 1 0 1 0 0 0 0 1 _ % C6.2.116 LDNP
x 0 1 0 1 0 0 0 1 0 _ % C6.2.256 STP
x 0 1 0 1 0 0 0 1 1 _ % C6.2.117 LDP
1 x 0 0 1 0 0 0 0 0 0 % C6.2.252 STLXR % C6.2.285 STXR
1 x 0 0 1 0 0 0 0 0 1 % C6.2.251 STLXP % C6.2.284 STXP
```

```
 1 x 0 0 1 0 0 0 0 1 0 % C6.2.104 LDAXR % C6.2.162 LDXR
 1 x 0 0 1 0 0 0 0 1 1 % C6.2.103 LDAXP % C6.2.161 LDXP
 1 x 0 0 1 0 0 0 1 0 0 % C6.2.244 STLLR % C6.2.245 STLR
 1 x 0 0 1 0 0 0 1 1 0 % C6.2.100 LDAR % C6.2.115 LDLAR
 1 x 0 0 1 0 0 0 1 i 1 % C6.2.040 CAS, CASA, CASAL, CASL
 x 0 0 0 1 0 1 0 s s 0 % C6.2.012 AND (shifted register)
 x 0 0 0 1 0 1 0 s s 1 % C6.2.029 BIC (shifted register)
 x 0 1 0 1 0 1 0 0 0 0 % C6.2.177 MOV (register)
 x 0 1 0 1 0 1 0 s s 1 % C6.2.186 MVN % C6.2.192 ORN (shifted register)
 x 1 0 0 1 0 1 0 s s 0 % C6.2.078 EOR (shifted register)
 x 1 0 0 1 0 1 0 s s 1 % C6.2.076 EON (shifted register)
 x 1 1 0 1 0 1 0 s s 0 % C6.2.014 ANDS (shifted register) % C6.2.308 TST (shifted regis
ter)
 x 1 1 0 1 0 1 0 s s 1 % C6.2.030 BICS (shifted register)
 x 0 0 0 1 0 1 1 0 0 1 % C6.2.001 ADC % C6.2.003 ADD (extended register)
 x 0 0 0 1 0 1 1 s s 0 % C6.2.005 ADD (shifted register)
 x 0 1 0 1 0 1 1 0 0 1 % C6.2.006 ADDS (extended register) % C6.2.53 CMN (extended regi
ster)
 x 0 1 0 1 0 1 1 s s 0 % C6.2.008 ADDS (shifted register) % C6.2.55 CMN (shifted regist
er)
 x 1 0 0 1 0 1 1 0 0 1 % C6.2.288 SUB (extended register)
 x 1 0 0 1 0 1 1 s s 0 % C6.2.290 SUB (shifted register) % C6.2.187 NEG (shifted regist
er)
 x 1 1 0 1 0 1 1 0 0 1 % C6.2.056 CMP (extended register) % C6.2.291 SUBS (extended reg
ister)
 x 1 1 0 1 0 1 1 s s 0 % C6.2.058 CMP (shifted register) % C6.2.293 SUBS (shifted regis
ter) % C6.2.188 NEGS
 x i i 1 0 0 0 0 _ _ _ % C6.2.009 ADR % C6.2.010 ADRP
 x 0 0 1 0 0 0 1 0 s _ % C6.2.004 ADD (immediate) % C6.2.173 MOV (to from SP)
 x 1 0 1 0 0 0 1 0 s _ % C6.2.289 SUB (immediate)
 x 0 1 1 0 0 0 1 0 s _ % C6.2.007 ADDS (immediate) % C6.2.54 CMN (immediate)
 x 1 1 1 0 0 0 1 0 s _ % C6.2.057 CMP (immediate) % C6.2.292 SUBS (immediate)
 x 0 0 1 0 0 1 0 0 n _ % C6.2.011 AND (immediate)
 x 0 0 1 0 0 1 0 1 h w % C6.2.174 MOV (inverted wide immediate) % C6.2.179 MOVN
                         % C6.2.221 SBFX % C6.2.298 SXTB % C6.2.299 SXTH % C6.2.300 SXTW
 x 0 0 1 0 0 1 1 0 n _ % C6.2.016 ASR (immediate) % C6.2.219 SBFIZ % C6.2.220 SBFM
                         % C6.2.221 SBFX % C6.2.298 SXTB % C6.2.299 SXTH % C6.2.300 SXTW
 x 0 0 1 0 0 1 1 1 n 0 % C6.2.082 EXTR % C6.2.214 ROR (immediate)
 x 0 1 1 0 0 1 0 0 n _ % C6.2.176 MOV (bitmask immediate) % C6.2.193 ORR (immediate)
 x 0 1 1 0 0 1 1 0 n _ % C6.2.025 BFC % C6.2.26 BFI % C6.2.27 BFM % C6.2.28 BFXIL
 x 1 0 1 0 0 1 0 0 n _ % C6.2.077 EOR (immediate)
 x 1 0 1 0 0 1 0 1 h w % C6.2.175 MOV (wide immediate) % C6.2.180 MOVZ
 x 1 0 1 0 0 1 1 0 n _ % C6.2.166 LSL (immediate) % C6.2.169 LSR (immediate) % C6.2.309
 UBFIZ % C6.2.310 UBFM
                         % C6.2.311 UBFX % C6.2.319 UXTB % C6.2.320 UXTH
 x 1 1 1 0 0 1 0 0 n _ % C6.2.013 ANDS (immediate) % C6.2.307 TST (immediate)
 x 1 1 1 0 0 1 0 1 h w % C6.2.178 MOVK
 0 0 0 1 0 1 _ _ _ _ _ % C6.2.024 B
 0 1 0 1 0 1 0 0 _ _ _ % C6.2.023 B.cond
 x 0 1 1 0 1 0 0 _ _ _ % C6.2.042 CBZ
 x 0 1 1 0 1 0 1 _ _ _ % C6.2.041 CBNZ
 1 0 0 1 0 1 _ _ _ _ _ % C6.2.031 BL
 1 1 0 1 0 1 0 0 0 0 0 % C6.2.085 HVC % C6.2.227 SMC % C6.2.294 SVC
 1 1 0 1 0 1 0 0 0 0 1 % C6.2.036 BRK
 1 1 0 1 0 1 0 0 0 1 0 % C6.2.084 HLT
 1 1 0 1 0 1 0 0 1 0 1 % C6.2.070 DCPS1 % C6.2.071 DCPS2 % C6.2.072 DCPS3 % C6.2.073 DM
```

```
B
 1 1 0 1 0 1 0 1 0 0 0 % C6.2.018 AT % C6.2.047 CFINV % C6.2.050 CLREX
                       % C6.2.021 AUTIA, AUTIA1716, AUTIASP, AUTIAZ, AUTIZA
                       % C6.2.022 AUTIB, AUTIB1716, AUTIBSP, AUTIBZ, AUTIZB
                       % C6.2.062 CSDB % C6.2.069 DC % C6.2.075 DSB % C6.2.081 ESB
                       % C6.2.083 HINT % C6.2.086 IC % C6.2.087 ISB % C6.2.182 MSR (imm
ediate)
                       % C6.2.183 MSR (register) % C6.2.198 PACIA, PACIA1716, PACIASP,
PACIAZ, PACIZA
                       % C6.2.204 PSB CSYNC % C6.2.205 PSSBB % C6.2.224 SEV % C6.2.225
SEVL
                       % C6.2.232 SSBB % C6.2.301 SYS % C6.2.305 TLBI % C6.2.306 TSB CS
YNC
                       % C6.2.321 WFE % C6.2.322 WFI % C6.2.324 YIELD % C6.2.191 NOP
 1 1 0 1 0 1 0 1 0 0 1 % C6.2.181 MRS % C6.2.302 SYSL
 x 0 1 1 0 1 1 0 _ _ _ % C6.2.304 TBZ
 x 0 1 1 0 1 1 1 _ _ _ % C6.2.303 TBNZ
 1 1 0 1 0 1 1 0 0 0 0 % C6.2.034 BR
 1 1 0 1 0 1 1 0 0 0 1 % C6.2.032 BLR
 1 1 0 1 0 1 1 0 0 1 0 % C6.2.207 RET % C6.2.208 RETAA, RETAB
 1 1 0 1 0 1 1 0 1 0 0 % C6.2.079 ERET % C6.2.080 ERETAA, ERETAB
 1 1 0 1 0 1 1 0 1 0 1 % C6.2.074 DRPS
 1 1 0 1 0 1 1 z 0 0 0 % C6.2.035 BRAA, BRAAZ, BRAB, BRABZ
 1 1 0 1 0 1 1 z 0 0 1 % C6.2.033 BLRAA, BLRAAZ, BLRAB, BLRABZ
 0 x 0 1 1 0 0 0 _ _ _ % C6.2.120 LDR (literal)
 0 0 1 1 1 0 0 0 0 0 0 % C6.2.273 STTRB % C6.2.282 STURB
 0 0 1 1 1 0 0 0 0 0 1 % C6.2.260 STRB (register)
 0 0 1 1 1 0 0 0 0 1 0 % C6.2.144 LDTRB % C6.2.156 LDURB
 0 0 1 1 1 0 0 0 0 1 1 % C6.2.124 LDRB (register)
 0 0 1 1 1 0 0 0 1 0 1 % C6.2.092 LDAPRB
 0 0 1 1 1 0 0 0 1 i 1 % C6.2.128 LDRSB (register)
 0 0 1 1 1 0 0 0 1 x 0 % C6.2.146 LDTRSB % C6.2.158 LDURSB
 0 0 1 1 1 0 0 0 a r 1 % C6.2.088 LDADDB, LDADDAB, LDADDALB, LDADDLB % C6.2.233 STADDB,
 STADDLB
                       % C6.2.107 LDCLRB, LDCLRAB, LDCLRALB, LDCLRLB % C6.2.236 STCLRB,
 STCLRLB
                       % C6.2.110 LDEORB, LDEORAB, LDEORALB, LDEORLB % C6.2.239 STEORB,
 STEORLB
                       % C6.2.134 LDSETB, LDSETAB, LDSETALB, LDSETLB % C6.2.263 STSETB,
 STSETLB
                       % C6.2.137 LDSMAXB, LDSMAXAB, LDSMAXALB, LDSMAXLB % C6.2.266 STS
MAXB, STSMAXLB
                       % C6.2.140 LDSMINB, LDSMINAB, LDSMINALB, LDSMINLB % C6.2.269 STS
MINB, STSMINLB
                       % C6.2.149 LDUMAXB, LDUMAXAB, LDUMAXALB, LDUMAXLB % C6.2.275 STU
MAXB, STUMAXLB
                       % C6.2.152 LDUMINB, LDUMINAB, LDUMINALB, LDUMINLB % C6.2.278 STU
MINB, STUMINLB
                       % C6.2.295 SWPB, SWPAB, SWPALB, SWPLB
 0 0 1 1 1 0 0 i 0 0 _ % C6.2.259 STRB (immediate)
 0 0 1 1 1 0 0 i 0 1 i % C6.2.123 LDRB (immediate)
 0 0 1 1 1 0 0 i 1 i i % C6.2.127 LDRSB (immediate)
 0 1 1 1 1 0 0 0 0 0 0 % C6.2.274 STTRH % C6.2.283 STURH
 0 1 1 1 1 0 0 0 0 0 1 % C6.2.262 STRH (register)
 0 1 1 1 1 0 0 0 0 1 0 % C6.2.145 LDTRH % C6.2.157 LDURH
 0 1 1 1 1 0 0 0 0 1 1 % C6.2.126 LDRH (register)
```

```
 0 1 1 1 1 0 0 0 1 0 1 % C6.2.093 LDAPRH
 0 1 1 1 1 0 0 0 1 x 0 % C6.2.147 LDTRSH % C6.2.159 LDURSH
 0 1 1 1 1 0 0 0 1 x 1 % C6.2.130 LDRSH (register)
 0 1 1 1 1 0 0 0 a r 1 % C6.2.089 LDADDH, LDADDAH, LDADDALH, LDADDLH % C6.2.234 STADDH,
 STADDLH
                       % C6.2.108 LDCLRH, LDCLRAH, LDCLRALH, LDCLRLH % C6.2.237 STCLRH,
 STCLRLH
                       % C6.2.111 LDEORH, LDEORAH, LDEORALH, LDEORLH % C6.2.240 STEORH,
 STEORLH
                       % C6.2.135 LDSETH, LDSETAH, LDSETALH, LDSETLH % C6.2.264 STSETH,
 STSETLH
                       % C6.2.138 LDSMAXH, LDSMAXAH, LDSMAXALH, LDSMAXLH % C6.2.267 STS
MAXH, STSMAXLH
                       % C6.2.141 LDSMINH, LDSMINAH, LDSMINALH, LDSMINLH % C6.2.270 STS
MINH, STSMINLH
                       % C6.2.150 LDUMAXH, LDUMAXAH, LDUMAXALH, LDUMAXLH % C6.2.276 STU
MAXH, STUMAXLH
                       % C6.2.153 LDUMINH, LDUMINAH, LDUMINALH, LDUMINLH % C6.2.279 STU
MINH, STUMINLH
                       % C6.2.296 SWPH, SWPAH, SWPALH, SWPLH
 0 1 1 1 1 0 0 i 0 0 _ % C6.2.261 STRH (immediate)
 0 1 1 1 1 0 0 i 0 1 i % C6.2.125 LDRH (immediate)
 0 1 1 1 1 0 0 i 1 x i % C6.2.129 LDRSH (immediate)
 0 1 1 1 1 0 0 i 0 0 _ % C6.2.261 STRH (immediate)
 0 1 1 1 1 0 0 i 0 1 i % C6.2.125 LDRH (immediate)
 0 1 1 1 1 0 0 i 1 x i % C6.2.129 LDRSH (immediate)
 0 1 0 1 1 0 0 1 0 0 0 % C6.2.250 STLURH
 0 1 0 1 1 0 0 1 0 1 0 % C6.2.096 LDAPURH
 0 1 0 1 1 0 0 1 1 i 0 % C6.2.098 LDAPURSH
 0 0 0 1 1 0 0 1 0 0 0 % C6.2.249 STLURB
 0 0 0 1 1 0 0 1 0 1 0 % C6.2.095 LDAPURB
 0 0 0 1 1 0 0 1 1 i 0 % C6.2.097 LDAPURSB
 1 0 1 1 1 0 0 0 1 0 0 % C6.2.148 LDTRSW % C6.2.160 LDURSW
 1 0 1 1 1 0 0 i 1 0 i % C6.2.131 LDRSW (immediate) % C6.2.133 LDRSW (register)
 1 0 0 1 1 0 0 0 _ _ _ % C6.2.132 LDRSW (literal)
 1 0 0 1 1 0 0 1 1 0 0 % C6.2.099 LDAPURSW
 1 x 0 1 1 0 0 1 0 0 0 % C6.2.248 STLUR
 1 x 0 1 1 0 0 1 0 1 0 % C6.2.094 LDAPUR
 1 x 1 1 1 0 0 0 0 0 0 % C6.2.272 STTR % C6.2.281 STUR
 1 x 1 1 1 0 0 0 0 0 1 % C6.2.258 STR (register)
 1 x 1 1 1 0 0 0 0 1 0 % C6.2.143 LDTR % C6.2.155 LDUR
 1 x 1 1 1 0 0 0 0 1 1 % C6.2.121 LDR (register)
 1 x 1 1 1 0 0 0 1 0 1 % C6.2.091 LDAPR
 1 x 1 1 1 0 0 0 a r 1 % C6.2.090 LDADD, LDADDA, LDADDAL, LDADDL % C6.2.235 STADD, STAD
DL
                       % C6.2.109 LDCLR, LDCLRA, LDCLRAL, LDCLRL % C6.2.238 STCLR, STCL
RL
                       % C6.2.112 LDEOR, LDEORA, LDEORAL, LDEORL % C6.2.241 STEOR, STEO
RL
                       % C6.2.136 LDSET, LDSETA, LDSETAL, LDSETL % C6.2.265 STSET, STSE
TL
                       % C6.2.139 LDSMAX, LDSMAXA, LDSMAXAL, LDSMAXL % C6.2.268 STSMAX,
 STSMAXL
                       % C6.2.142 LDSMIN, LDSMINA, LDSMINAL, LDSMINL % C6.2.271 STSMIN,
 STSMINL
                       % C6.2.151 LDUMAX, LDUMAXA, LDUMAXAL, LDUMAXL % C6.2.277 STUMAX,
```

```
 STUMAXL
                          % C6.2.154 LDUMIN, LDUMINA, LDUMINAL, LDUMINL % C6.2.280 STUMIN,
 STUMINL
                          % C6.2.297 SWP, SWPA, SWPAL, SWPL
 1 x 1 1 1 0 0 i 0 0 _ % C6.2.257 STR (immediate)
 1 x 1 1 1 0 0 i 0 1 i % C6.2.119 LDR (immediate)
 1 1 0 1 1 0 0 0 _ _ _ % C6.2.201 PRFM (literal)
 1 1 1 1 1 0 0 0 1 0 0 % C6.2.203 PRFUM
 1 1 1 1 1 0 0 0 1 0 1 % C6.2.202 PRFM (register)
 1 1 1 1 1 0 0 0 m s 1 % C6.2.122 LDRAA, LDRAB
 1 1 1 1 1 0 0 0 m s 1 % C6.2.122 LDRAA, LDRAB
 1 1 1 1 1 0 0 1 1 0 _ % C6.2.200 PRFM (immediate)
 0 0 1 1 1 0 1 0 0 0 0 % C6.2.223 SETF8, SETF16
 x 0 0 1 1 0 1 0 1 0 0 % C6.2.048 CINC % C6.2.66 CSINC % C6.2.64 CSET % C6.2.063 CSEL
 x 0 0 1 1 0 1 0 1 1 0 % C6.2.015 ASR (register) % C6.2.17 ASRV % C6.2.060 CRC32B, CRC3
2H, CRC32W, CRC32X
                          % C6.2.061 CRC32CB, CRC32CH, CRC32CW, CRC32CX
                          % C6.2.165 LSL (register) % C6.2.167 LSLV % C6.2.168 LSR (regist
er) % C6.2.170 LSRV
                          % C6.2.215 ROR (register) % C6.2.216 RORV % C6.2.222 SDIV % C6.2.
313 UDIV
 x 0 0 1 1 0 1 1 0 0 0 % C6.2.171 MADD % C6.2.185 MUL % C6.2.172 MNEG % C6.2.184 MSUB
 x 0 1 1 1 0 1 0 0 0 0 % C6.2.002 ADCS
 x 0 1 1 1 0 1 0 0 1 0 % C6.2.043 CCMN (immediate) % C6.2.044 CCMN (register)
 x 0 1 1 1 0 1 0 0 0 0 % C6.2.002 ADCS
 x 1 0 1 1 0 1 0 0 0 0 % C6.2.189 NGC % C6.2.217 SBC
 x 1 0 1 1 0 1 0 1 0 0 % C6.2.049 CINV % C6.2.65 CSETM % C6.2.67 CSINV
 x 1 0 1 1 0 1 0 1 0 0 % C6.2.059 CNEG % C6.2.68 CSNEG
 x 1 0 1 1 0 1 0 1 1 0 % C6.2.051 CLS
 x 1 0 1 1 0 1 0 1 1 0 % C6.2.052 CLZ
 x 1 0 1 1 0 1 0 1 1 0 % C6.2.206 RBIT
 x 1 0 1 1 0 1 0 1 1 0 % C6.2.209 REV % C6.2.210 REV16 % C6.2.212 REV64
 x 1 0 1 1 0 1 0 0 0 0 % C6.2.189 NGC % C6.2.217 SBC
 x 1 1 1 1 0 1 0 0 0 0 % C6.2.190 NGCS % C6.2.218 SBCS
 x 1 1 1 1 0 1 0 0 1 0 % C6.2.045 CCMP (immediate) % C6.2.046 CCMP (register)
 1 0 0 1 1 0 1 0 1 1 0 % C6.2.197 PACGA
 1 0 0 1 1 0 1 1 0 0 1 % C6.2.226 SMADDL % C6.2.231 SMULL % C6.2.228 SMNEGL % C6.2.229
SMSUBL
 1 0 0 1 1 0 1 1 0 1 0 % C6.2.230 SMULH
 1 0 0 1 1 0 1 1 1 0 1 % C6.2.314 UMADDL % C6.2.318 UMULL % C6.2.315 UMNEGL % C6.2.316
UMSUBL
 1 0 0 1 1 0 1 1 1 1 0 % C6.2.317 UMULH
 1 0 1 1 1 0 1 0 0 0 0 % C6.2.213 RMIF
 1 1 0 1 1 0 1 0 1 1 0 % C6.2.019 AUTDA, AUTDZA % C6.2.020 AUTDB, AUTDZB
                          % C6.2.021 AUTIA, AUTIA1716, AUTIASP, AUTIAZ, AUTIZA
                          % C6.2.022 AUTIB, AUTIB1716, AUTIBSP, AUTIBZ, AUTIZB
                          % C6.2.195 PACDA, PACDZA % C6.2.196 PACDB, PACDZB
                          % C6.2.211 REV32 % C6.2.323 XPACD, XPACI, XPACLRI
 1 1 0 1 1 0 1 0 1 1 1 % C6.2.199 PACIB, PACIB1716, PACIBSP, PACIBZ, PACIZB
```

SIMD NEON

```
0 x 0 s 1 1 1 0 s s 1 ... 0 | 1 0 1 1 1 0 % C7.2.001 ABS
```

```
0 x 0 s 1 1 1 0 s s 1 ... _ | 1 0 0 0 0 1 % C7.2.002 ADD (vector)
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 1 0 0 0 0 % C7.2.003 ADDHN, ADDHN2
0 1 0 1 1 1 1 1 0 s s 1 ... 1 | 1 0 1 1 1 0 % C7.2.004 ADDP (scalar)
0 x 0 0 1 1 1 1 0 s s 1 ... _ | 1 0 1 1 1 1 % C7.2.005 ADDP (vector)
0 x 0 0 1 1 1 1 0 s s 1 ... 1 | 1 0 1 1 1 0 % C7.2.006 ADDV
0 1 0 0 1 1 1 0 0 0 1 ... 0 | 0 1 0 1 1 0 % C7.2.007 AESD  AES single round decryption
.
0 1 0 0 1 1 1 0 0 0 1 ... 0 | 0 1 0 0 1 0 % C7.2.008 AESE  AES single round encryption
.
0 1 0 0 1 1 1 0 0 0 1 ... 0 | 0 1 1 1 1 0 % C7.2.009 AESIMC  AES inverse mix columns.
0 1 0 0 1 1 1 0 0 0 1 ... 0 | 0 1 1 0 1 0 % C7.2.010 AESMC  AES mix columns.
0 x 0 0 1 1 1 0 0 0 1 ... _ | 0 0 0 1 1 1 % C7.2.011 AND (vector)
1 1 0 0 1 1 1 0 0 0 1 ... _ | 0 _ _ _ _ _ % C7.2.012 BCAX
0 x 1 0 1 1 1 1 0 0 0 ... c | x x x 1 0 1 % C7.2.013 BIC (vector, immediate)
0 x 0 0 1 1 1 0 0 1 1 ... _ | 0 0 0 1 1 1 % C7.2.014 BIC (vector, register)

C7.2.119 FMLS (vector)
0 x s 0 1 1 1 1 1 0 i ... _ | 1 1 0 0 h 0 % C7.2.120 FMLSL, FMLSL2 (by element)
0 x s 0 1 1 1 0 1 0 1 ... _ | 1 1 0 0 1 1 % C7.2.121 FMLSL, FMLSL2 (vector)
0 x x 0 1 1 1 1 0 0 0 ... _ | 1 1 1 1 0 1 % C7.2.122 FMOV (vector, immediate)
0 0 0 1 1 1 1 0 x x 1 ... 0 | 0 1 0 0 0 0 % C7.2.123 FMOV (register)
x 0 0 1 1 1 1 0 x x 1 ... _ | 0 0 0 0 0 0 % C7.2.124 FMOV (general)
0 0 0 1 1 1 1 0 x x 1 ... _ | _ _ _ 1 0 0 % C7.2.125 FMOV (scalar, immediate)
0 0 0 1 1 1 1 1 x x 1 ... _ | 1 _ _ _ _ _ % C7.2.126 FMSUB
0 x 1 0 1 1 1 0 1 x 1 ... 0 | 1 1 1 1 1 0 % C7.2.132 FNEG (vector)
0 0 0 1 1 1 1 0 s s 1 ... 1 | 0 1 0 0 0 0 % C7.2.133 FNEG (scalar)
0 0 0 1 1 1 1 1 s s 1 ... _ | 0 _ _ _ _ _ % C7.2.134 FNMADD
0 0 0 1 1 1 1 1 s s 1 ... _ | 1 _ _ _ _ _ % C7.2.135 FNMSUB
0 0 0 1 1 1 1 0 s s 1 ... _ | 1 0 0 0 1 0 % C7.2.136 FNMUL (scalar)
0 1 0 1 1 1 1 1 0 1 x 1 ... 1 | 1 1 0 1 1 0 % C7.2.137 FRECPE
0 1 0 1 1 1 1 1 0 0 x 1 ... _ | p p 1 1 1 1 % C7.2.138 FRECPS
0 1 0 1 1 1 1 1 0 1 x 1 ... 1 | 1 1 1 1 1 0 % C7.2.139 FRECPX
0 x 1 0 1 1 1 0 0 x 1 ... 1 | 1 0 0 0 1 0 % C7.2.140 FRINTA (vector)
0 0 0 1 1 1 1 1 0 s s 1 ... 0 | 0 1 0 0 0 0 % C7.2.141 FRINTA (scalar)
0 x 1 0 1 1 1 0 1 x 1 ... 1 | 1 0 0 1 1 0 % C7.2.142 FRINTI (vector)
0 0 0 1 1 1 1 1 0 s s 1 ... 1 | 1 1 0 0 0 0 % C7.2.143 FRINTI (scalar)
0 x 0 0 1 1 1 0 0 x 1 ... 1 | 1 0 0 1 1 0 % C7.2.144 FRINTM (vector)
0 0 0 1 1 1 1 1 0 s s 1 ... 1 | 0 1 0 0 0 0 % C7.2.145 FRINTM (scalar)
0 x 0 0 1 1 1 0 0 x 1 ... 1 | 1 0 0 0 1 0 % C7.2.146 FRINTN (vector)
0 0 0 1 1 1 1 1 0 s s 1 ... 0 | 0 1 0 0 0 0 % C7.2.147 FRINTN (scalar)
0 x 0 0 1 1 1 0 1 x 1 ... 1 | 1 0 0 0 1 0 % C7.2.148 FRINTP (vector)
0 0 0 1 1 1 1 1 0 s s 1 ... 0 | 1 1 0 0 0 0 % C7.2.149 FRINTP (scalar)
0 x 1 0 1 1 1 0 0 s 1 ... 1 | 1 0 0 1 1 0 % C7.2.150 FRINTX (vector)
0 0 0 1 1 1 1 1 0 s s 1 ... 1 | 0 1 0 0 0 0 % C7.2.151 FRINTX (scalar)
0 x 0 0 1 1 1 0 0 1 x ... 1 | 1 0 0 1 1 0 % C7.2.152 FRINTZ (vector)
0 0 0 1 1 1 1 1 0 s s 1 ... 1 | 1 1 0 0 0 0 % C7.2.153 FRINTZ (scalar)
0 1 1 1 1 1 1 1 0 1 x x ... 1 | 1 1 0 1 1 0 % C7.2.154 FRSQRTE
0 1 0 1 1 1 1 1 0 1 x x ... _ | p p 1 1 1 1 % C7.2.155 FRSQRTS
0 x 1 0 1 1 1 0 1 1 1 ... 1 | 1 1 1 1 1 0 % C7.2.156 FSQRT (vector)
0 0 0 1 1 1 1 1 0 s s 1 ... 1 | 1 1 0 0 0 0 % C7.2.157 FSQRT (scalar)
0 x 0 0 1 1 1 0 1 x 1 ... _ | p p 0 1 0 1 % C7.2.158 FSUB (vector)
0 0 0 1 1 1 1 1 0 s s 1 ... _ | 0 0 1 1 1 0 % C7.2.159 FSUB (scalar)
0 x 0 0 1 1 0 0 i 1 0 ... _ | x x 1 x s s % C7.2.162 LD1 (multiple structures)
0 x 0 0 1 1 0 1 i 1 0 ... _ | 1 1 0 0 s s % C7.2.164 LD1R
0 x 0 0 1 1 0 1 i 1 0 ... _ | x x 0 s s s % C7.2.163 LD1 (single structure)
0 x 0 0 1 1 0 0 i 1 0 ... _ | 1 0 0 0 s s % C7.2.165 LD2 (multiple structures)
```

```
0 x 0 0 1 1 0 1 i 1 1 ... _ | 1 1 0 0 s s % C7.2.167 LD2R
0 x 0 0 1 1 0 1 i 1 1 ... _ | x x 0 s s s % C7.2.166 LD2 (single structure)
0 x 0 0 1 1 0 0 i 1 0 ... _ | 0 1 0 0 s s % C7.2.168 LD3 (multiple structures)
0 x 0 0 1 1 0 1 0 1 0 ... 0 | 1 1 1 0 s s % C7.2.170 LD3R
0 x 0 0 1 1 0 1 i 1 0 ... _ | x x 1 s s s % C7.2.169 LD3 (single structure)
0 x 0 0 1 1 0 0 0 1 0 ... 0 | 0 0 0 0 s s % C7.2.171 LD4 (multiple structures)
0 x 0 0 1 1 0 1 i 1 1 ... _ | 1 1 1 0 s s % C7.2.173 LD4R
0 x 0 0 1 1 0 1 i 1 1 ... _ | x x 1 s s s % C7.2.172 LD4 (single structure)
x x 1 0 1 1 0 0 0 1 _ ... _ | _ _ _ _ _ _ % C7.2.174 LDNP (SIMD and FP)
x x 1 0 1 1 0 i i 1 _ ... _ | _ _ _ _ _ _ % C7.2.175 LDP (SIMD and FP)
x x 1 1 1 1 1 0 0 x 1 0 ... _ | _ _ _ _ 0 1 % C7.2.176 LDR (immediate, SIMD and FP)
x x 0 1 1 1 1 0 0 _ _ _ ... _ | _ _ _ _ _ _ % C7.2.177 LDR (literal, SIMD and FP)
x x 1 1 1 1 1 0 0 x 1 1 ... _ | _ _ _ _ 1 0 % C7.2.178 LDR (register, SIMD and FP)
x x 1 1 1 1 1 0 0 x 1 0 ... _ | _ _ _ _ 0 0 % C7.2.179 LDUR (SIMD and FP)
0 x 1 0 1 1 1 1 s s i ... _ | 0 0 0 0 h 0 % C7.2.180 MLA (by element)
0 x 0 0 1 1 1 0 s s 1 ... _ | 1 0 0 1 0 1 % C7.2.181 MLA (vector)
0 x 1 0 1 1 1 1 s s i ... _ | 0 1 0 0 h 0 % C7.2.182 MLS (by element)
0 x 1 0 1 1 1 0 s s 1 ... _ | 1 0 0 1 0 1 % C7.2.183 MLS (vector)
0 1 0 1 1 1 1 0 0 0 0 ... _ | 0 0 0 0 0 1 % C7.2.184 MOV (scalar) % C7.2.32 DUP (elemen
t)
0 1 1 0 1 1 1 0 0 0 0 ... _ | 0 _ _ _ _ 1 % C7.2.185 MOV (element) % C7.2.160 INS (elem
ent)
0 1 0 0 1 1 1 0 0 0 0 ... _ | 0 0 0 1 1 1 % C7.2.186 MOV (from general) % C7.2.161 INS
(general)
0 x 0 0 1 1 1 0 1 0 1 ... _ | 0 0 0 1 1 1 % C7.2.187 MOV (vector) % C7.2.198 ORR (vecto
r, register)
0 x 0 0 1 1 1 0 0 0 0 ... 0 | 0 0 1 1 1 1 % C7.2.188 MOV (to general)
0 x s 0 1 1 1 1 0 0 0 ... _ | _ _ _ _ 0 1 % C7.2.189 MOVI
0 x 0 0 1 1 1 1 s s i ... _ | 1 0 0 0 h 0 % C7.2.190 MUL (by element)
0 x 0 0 1 1 1 0 s s 1 ... _ | 1 0 0 1 1 1 % C7.2.191 MUL (vector)
0 x 1 0 1 1 1 0 0 0 1 ... 0 | 0 1 0 0 1 1 % C7.2.192 MVN % C7.2.195 NOT
0 x 1 0 1 1 1 1 0 0 0 ... _ | _ _ _ _ 0 1 % C7.2.193 MVNI
0 x 1 0 1 1 1 0 s s 1 ... 0 | 1 0 1 1 1 1 % C7.2.194 NEG (vector)
0 x 0 0 1 1 1 0 1 1 1 ... _ | 0 0 0 1 1 1 % C7.2.196 ORN (vector)
0 x 0 0 1 1 1 1 0 0 0 ... _ | _ _ _ 1 0 1 % C7.2.197 ORR (vector, immediate)
0 x 1 0 1 1 1 0 s s 1 ... _ | 1 0 0 1 1 1 % C7.2.199 PMUL
0 x 0 0 1 1 1 0 s s 1 ... _ | 1 1 1 0 0 0 % C7.2.200 PMULL, PMULL2
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 1 0 0 0 0 % C7.2.201 RADDHN, RADDHN2
1 1 0 0 1 1 1 0 0 1 1 ... _ | 1 0 0 0 1 1 % C7.2.202 RAX1
0 x 1 0 1 1 1 0 0 1 1 ... 0 | 0 1 0 1 1 0 % C7.2.203 RBIT (vector)
0 x 0 0 1 1 1 0 s s 1 ... 0 | 0 0 0 1 1 0 % C7.2.204 REV16 (vector)
0 x 1 0 1 1 1 0 s s 1 ... 0 | 0 0 0 0 1 0 % C7.2.205 REV32 (vector)
0 x 0 0 1 1 1 0 s s 1 ... 0 | 0 0 0 0 1 0 % C7.2.206 REV64
0 x 0 0 1 1 1 1 0 _ _ ... _ | 1 0 0 0 1 1 % C7.2.207 RSHRN, RSHRN2
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 1 1 0 0 0 % C7.2.208 RSUBHN, RSUBHN2
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 1 1 1 1 1 % C7.2.209 SABA
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 1 0 1 0 0 % C7.2.210 SABAL, SABAL2
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 1 1 1 0 1 % C7.2.211 SABD
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 1 1 1 0 0 % C7.2.212 SABDL, SABDL2
0 x 0 0 1 1 1 0 s s 1 ... 0 | 0 1 1 0 1 0 % C7.2.213 SADALP
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 0 0 0 0 0 % C7.2.214 SADDL, SADDL2
0 x 0 0 1 1 1 0 s s 1 ... 0 | 0 0 1 0 1 0 % C7.2.215 SADDLP
0 x 0 0 1 1 1 0 s s 1 ... 0 | 0 0 1 1 1 0 % C7.2.216 SADDLV
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 0 0 1 0 0 % C7.2.217 SADDW, SADDW2
0 x 0 i 1 1 1 1 0 _ _ ... _ | 1 1 1 0 0 1 % C7.2.218 SCVTF (vector, fixed point)
0 x 0 i 1 1 1 0 0 i 1 ... 1 | 1 1 0 1 1 0 % C7.2.219 SCVTF (vector, integer)
```

```
x 0 0 1 1 1 1 0 s s 0 ... 0 | _ _ _ _ _ _ % C7.2.220 SCVTF (scalar, fixed point)
x 0 0 1 1 1 1 0 s s 1 ... 0 | 0 0 0 0 0 0 % C7.2.221 SCVTF (scalar, integer)
0 x 0 0 1 1 1 1 s s i ... _ | 1 1 1 0 h 0 % C7.2.222 SDOT (by element)
0 x 0 0 1 1 1 0 s s 0 ... _ | 1 0 0 1 0 1 % C7.2.223 SDOT (vector)
0 1 0 1 1 1 1 0 0 0 0 ... _ | 0 0 0 0 0 0 % C7.2.224 SHA1C
0 1 0 1 1 1 1 0 0 0 1 ... 0 | 0 0 0 0 1 0 % C7.2.225 SHA1H
0 1 0 1 1 1 1 0 0 0 0 ... _ | 0 0 1 0 0 0 % C7.2.226 SHA1M
0 1 0 1 1 1 1 0 0 0 0 ... _ | 0 0 0 1 0 0 % C7.2.227 SHA1P
0 1 0 1 1 1 1 0 0 0 0 ... _ | 0 0 1 1 0 0 % C7.2.228 SHA1SU0
0 1 0 1 1 1 1 0 0 0 1 ... 0 | 0 0 0 1 1 0 % C7.2.229 SHA1SU1
0 1 0 1 1 1 1 0 0 0 0 ... _ | 0 1 0 1 0 0 % C7.2.230 SHA256H2
0 1 0 1 1 1 1 0 0 0 0 ... _ | 0 1 0 0 0 0 % C7.2.231 SHA256H
0 1 0 1 1 1 1 0 0 0 1 ... 0 | 0 0 1 0 1 0 % C7.2.232 SHA256SU0
0 1 0 1 1 1 1 0 0 0 0 ... _ | 0 1 1 0 0 0 % C7.2.233 SHA256SU1
1 1 0 0 1 1 1 0 0 1 1 ... _ | 1 0 0 0 0 0 % C7.2.234 SHA512H
1 1 0 0 1 1 1 0 0 1 1 ... _ | 1 0 0 0 0 1 % C7.2.235 SHA512H2
1 1 0 0 1 1 1 0 1 1 0 ... 0 | 1 0 0 0 0 0 % C7.2.236 SHA512SU0
1 1 0 0 1 1 1 0 0 1 1 ... _ | 1 0 0 0 1 0 % C7.2.237 SHA512SU1
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 0 0 0 0 1 % C7.2.238 SHADD
0 x 0 i 1 1 1 1 0 _ _ ... _ | 0 1 0 1 0 1 % C7.2.239 SHL
0 x 1 0 1 1 1 0 s s 1 ... 1 | 0 0 1 1 1 0 % C7.2.240 SHLL, SHLL2
0 x 0 0 1 1 1 1 0 _ _ ... _ | 1 0 0 0 0 1 % C7.2.241 SHRN, SHRN2
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 0 1 0 0 1 % C7.2.242 SHSUB
0 x 1 i 1 1 1 1 0 _ _ ... _ | 0 1 0 1 0 1 % C7.2.243 SLI
1 1 0 0 1 1 1 0 0 1 1 ... _ | 1 1 0 0 0 0 % C7.2.244 SM3PARTW1
1 1 0 0 1 1 1 0 0 1 1 ... _ | 1 1 0 0 0 1 % C7.2.245 SM3PARTW2
1 1 0 0 1 1 1 0 0 1 0 ... _ | 0 _ _ _ _ _ % C7.2.246 SM3SS1
1 1 0 0 1 1 1 0 0 1 0 ... _ | 1 0 i i 0 0 % C7.2.247 SM3TT1A
1 1 0 0 1 1 1 0 0 1 0 ... _ | 1 0 i i 0 1 % C7.2.248 SM3TT1B
1 1 0 0 1 1 1 0 0 1 0 ... _ | 1 0 i i 1 0 % C7.2.249 SM3TT2A
1 1 0 0 1 1 1 0 0 1 0 ... _ | 1 0 i i 1 1 % C7.2.250 SM3TT2B
1 1 0 0 1 1 1 0 1 1 0 ... 0 | 1 0 0 0 0 1 % C7.2.251 SM4E
1 1 0 0 1 1 1 0 0 1 1 ... _ | 1 1 0 0 1 0 % C7.2.252 SM4EKEY
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 1 1 0 0 1 % C7.2.253 SMAX
0 x 0 0 1 1 1 0 s s 1 ... _ | 1 0 1 0 0 1 % C7.2.254 SMAXP
0 x 0 0 1 1 1 0 s s 1 ... 0 | 1 0 1 0 1 0 % C7.2.255 SMAXV
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 1 1 0 1 1 % C7.2.256 SMIN
0 x 0 0 1 1 1 0 s s 1 ... _ | 1 0 1 0 1 1 % C7.2.257 SMINP
0 x 0 0 1 1 1 0 s s 1 ... 1 | 1 0 1 0 1 0 % C7.2.258 SMINV
0 x 0 0 1 1 1 1 s s i ... _ | 0 0 1 0 h 0 % C7.2.259 SMLAL, SMLAL2 (by element)
0 x 0 0 1 1 1 0 s s 1 ... _ | 1 0 0 0 0 0 % C7.2.260 SMLAL, SMLAL2 (vector)
0 x 0 0 1 1 1 1 s s i ... _ | 0 1 1 0 h 0 % C7.2.261 SMLSL, SMLSL2 (by element)
0 x 0 0 1 1 1 0 s s 1 ... _ | 1 0 1 0 0 0 % C7.2.262 SMLSL, SMLSL2 (vector)
0 x 0 0 1 1 1 0 0 0 0 ... _ | 0 0 1 0 1 1 % C7.2.263 SMOV
0 x 0 0 1 1 1 1 s s i ... _ | 1 0 1 0 h 0 % C7.2.264 SMULL, SMULL2 (by element)
0 x 0 0 1 1 1 0 s s 1 ... _ | 1 1 0 0 0 0 % C7.2.265 SMULL, SMULL2 (vector)
0 x 0 i 1 1 1 0 s s 1 ... 0 | 0 1 1 1 1 0 % C7.2.266 SQABS
0 x 0 i 1 1 1 0 s s 1 ... _ | 0 0 0 0 1 1 % C7.2.267 SQADD
0 x 0 i 1 1 1 1 s s i ... _ | 0 0 1 1 h 0 % C7.2.268 SQDMLAL, SQDMLAL2 (by element)
0 x 0 i 1 1 1 0 s s 1 ... _ | 1 0 0 1 0 0 % C7.2.269 SQDMLAL, SQDMLAL2 (vector)
0 x 0 i 1 1 1 1 s s i ... _ | 0 1 1 1 h 0 % C7.2.270 SQDMLSL, SQDMLSL2 (by element)
0 x 0 i 1 1 1 0 s s 1 ... _ | 1 0 1 1 0 0 % C7.2.271 SQDMLSL, SQDMLSL2 (vector)
0 x 0 i 1 1 1 1 s s i ... _ | 1 1 0 0 h 0 % C7.2.272 SQDMULH (by element)
0 x 0 i 1 1 1 0 s s 1 ... _ | 1 0 1 1 0 1 % C7.2.273 SQDMULH (vector)
0 x 0 0 1 1 1 1 s s i ... _ | 1 0 1 1 h 0 % C7.2.274 SQDMULL, SQDMULL2 (by element)
0 x 0 i 1 1 1 0 s s 1 ... _ | 1 1 0 1 0 0 % C7.2.275 SQDMULL, SQDMULL2 (vector)
```

```
0 x 1 i 1 1 1 0 s s 1 ... _ | 0 1 1 1 1 0 % C7.2.276 SQNEG
0 x 1 i 1 1 1 1 s s i ... _ | 1 1 0 1 h 0 % C7.2.277 SQRDMLAH (by element)
0 x 1 i 1 1 1 0 s s 0 ... _ | 1 0 0 0 0 1 % C7.2.278 SQRDMLAH (vector)
0 x 1 i 1 1 1 1 s s i ... _ | 1 1 1 1 h 0 % C7.2.279 SQRDMLSH (by element)
0 x 1 i 1 1 1 0 s s 0 ... _ | 1 0 0 0 1 1 % C7.2.280 SQRDMLSH (vector)
0 x 0 i 1 1 1 1 s s i ... _ | 1 1 0 1 h 0 % C7.2.281 SQRDMULH (by element)
0 x 1 i 1 1 1 0 s s 1 ... _ | 1 0 1 1 0 1 % C7.2.282 SQRDMULH (vector)
0 x 0 i 1 1 1 0 s s 1 ... _ | 0 1 0 1 1 1 % C7.2.283 SQRSHL
0 x 0 i 1 1 1 1 0 _ _ ... _ | 1 0 0 1 1 1 % C7.2.284 SQRSHRN, SQRSHRN2
0 x 1 i 1 1 1 1 0 _ _ ... _ | 1 0 0 0 1 1 % C7.2.285 SQRSHRUN, SQRSHRUN2
0 x 0 i 1 1 1 1 0 _ _ ... _ | 0 1 1 1 0 1 % C7.2.286 SQSHL (immediate)
0 x 0 i 1 1 1 0 s s 1 ... _ | 0 1 0 0 1 1 % C7.2.287 SQSHL (register)
0 x 1 0 1 1 1 1 0 _ _ ... _ | 0 1 1 0 0 1 % C7.2.288 SQSHLU
0 x 0 i 1 1 1 1 0 _ _ ... _ | 1 0 0 1 0 1 % C7.2.289 SQSHRN, SQSHRN2
0 x 1 i 1 1 1 1 0 _ _ ... _ | 1 0 0 0 0 1 % C7.2.290 SQSHRUN, SQSHRUN2
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 0 1 0 1 1 % C7.2.291 SQSUB
0 x 0 0 1 1 1 0 s s 1 ... 1 | 0 1 0 0 1 0 % C7.2.292 SQXTN, SQXTN2
0 x 1 0 1 1 1 0 s s 1 ... 1 | 0 0 1 0 1 0 % C7.2.293 SQXTUN, SQXTUN2
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 0 0 1 0 1 % C7.2.294 SRHADD
0 x 1 i 1 1 1 1 0 _ _ ... _ | 0 1 0 0 0 1 % C7.2.295 SRI
0 x 0 i 1 1 1 0 s s 1 ... _ | 0 1 0 1 0 1 % C7.2.296 SRSHL
0 x 0 i 1 1 1 1 0 _ _ ... _ | 0 0 1 0 0 1 % C7.2.297 SRSHR
0 x 0 i 1 1 1 1 0 _ _ ... _ | 0 0 1 1 0 1 % C7.2.298 SRSRA
0 x 0 i 1 1 1 0 s s 1 ... _ | 0 1 0 0 0 1 % C7.2.299 SSHL
0 x 0 i 1 1 1 1 0 _ _ ... _ | 0 0 0 0 0 1 % C7.2.301 SSHR
0 x 0 i 1 1 1 1 0 _ _ ... _ | 0 0 0 1 0 1 % C7.2.302 SSRA
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 0 1 0 0 0 % C7.2.303 SSUBL, SSUBL2
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 0 1 1 0 0 % C7.2.304 SSUBW, SSUBW2
0 x 0 0 1 1 0 0 i 0 0 ... _ | x x 1 x s s % C7.2.305 ST1 (multiple structures)
0 x 0 0 1 1 0 1 i 0 0 ... _ | x x 0 s s s % C7.2.306 ST1 (single structure)
0 x 0 0 1 1 0 0 i 0 0 ... _ | 1 0 0 0 s s % C7.2.307 ST2 (multiple structures)
0 x 0 0 1 1 0 1 i 0 1 ... _ | x x 0 s s s % C7.2.308 ST2 (single structure)
0 x 0 0 1 1 0 0 i 0 0 ... _ | 0 1 0 0 s s % C7.2.309 ST3 (multiple structures)
0 x 0 0 1 1 0 1 i 0 0 ... _ | x x 1 s s s % C7.2.310 ST3 (single structure)
0 x 0 0 1 1 0 0 x 0 0 ... _ | 0 0 0 0 s s % C7.2.311 ST4 (multiple structures)
0 x 0 0 1 1 0 1 0 0 1 ... 0 | x x 1 s s s % C7.2.312 ST4 (single structure)
s s 1 0 1 1 0 0 0 0 _ ... _ | _ _ _ _ _ _ % C7.2.313 STNP (SIMD and FP)
s s 1 0 1 1 0 i i 0 _ ... _ | _ _ _ _ _ _ % C7.2.314 STP (SIMDFP)
s s 1 1 1 1 0 0 x 0 0 ... _ | _ _ _ _ i 1 % C7.2.317 STUR (SIMD and FP)
s s 1 1 1 1 0 0 x 0 1 ... _ | _ _ _ _ 1 0 % C7.2.316 STR (register, SIMD and FP)
s s 1 1 1 1 0 0 x 0 0 ... _ | _ _ _ _ 0 0 % C7.2.317 STUR (SIMD and FP)
0 x 1 0 1 1 1 0 s s 1 ... _ | 1 0 0 0 0 1 % C7.2.318 SUB (vector)
0 x 0 0 1 1 1 0 s s 1 ... _ | 0 1 1 0 0 0 % C7.2.319 SUBHN, SUBHN2
0 x 0 0 1 1 1 0 s s 1 ... 0 | 0 0 1 1 1 0 % C7.2.320 SUQADD
0 x 0 0 1 1 1 1 0 _ _ ... 0 | 1 0 1 0 0 1 % C7.2.321 SXTL, SXTL2 % C7.2.300 SSHLL, SSHL
L2
0 x 0 0 1 1 1 0 0 0 0 ... _ | 0 i i 0 0 0 % C7.2.322 TBL
0 x 0 0 1 1 1 0 0 0 0 ... _ | 0 i i 1 0 0 % C7.2.323 TBX
0 x 0 0 1 1 1 0 s s 0 ... _ | 0 0 1 0 1 0 % C7.2.324 TRN1
0 x 0 0 1 1 1 0 s s 0 ... _ | 0 1 1 0 1 0 % C7.2.325 TRN2
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 1 1 1 1 1 % C7.2.326 UABA
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 1 0 1 0 0 % C7.2.327 UABAL, UABAL2
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 1 1 1 0 1 % C7.2.328 UABD
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 1 1 1 0 0 % C7.2.329 UABDL, UABDL2
0 x 1 0 1 1 1 0 s s 1 ... 0 | 0 1 1 0 1 0 % C7.2.330 UADALP
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 0 0 0 0 0 % C7.2.331 UADDL, UADDL2
```

```
0 x 1 0 1 1 1 0 s s 1 ... 0 | 0 0 1 1 0 1 % C7.2.332 UADDLP
0 x 1 0 1 1 1 0 s s 1 ... 0 | 0 0 1 1 1 0 % C7.2.333 UADDLV
0 x 1 0 1 1 1 1 0 _ _ ... _ | 1 1 1 0 0 1 % C7.2.335 UCVTF (vector, fixed point)
0 x 1 1 1 1 1 0 0 s 1 ... 1 | 1 1 0 1 1 0 % C7.2.336 UCVTF (vector, integer)
x 0 0 1 1 1 1 0 s s 0 ... 1 | _ _ _ _ _ _ % C7.2.337 UCVTF (scalar, fixed point)
x 0 0 1 1 1 1 0 s s 1 ... 1 | 0 0 0 0 0 0 % C7.2.338 UCVTF (scalar, integer)
0 x 1 0 1 1 1 1 s s i ... _ | 1 1 1 0 h 0 % C7.2.339 UDOT (by element)
0 x 1 0 1 1 1 0 s s 0 ... _ | 1 0 0 1 0 1 % C7.2.340 UDOT (vector)
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 0 0 0 0 1 % C7.2.341 UHADD
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 0 1 0 0 1 % C7.2.342 UHSUB
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 1 1 0 0 1 % C7.2.343 UMAX
0 x 1 0 1 1 1 0 s s 1 ... _ | 1 0 1 0 0 1 % C7.2.344 UMAXP
0 x 1 0 1 1 1 0 s s 1 ... 0 | 1 0 1 0 1 0 % C7.2.345 UMAXV
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 1 1 0 1 1 % C7.2.346 UMIN
0 x 1 0 1 1 1 0 s s 1 ... _ | 1 0 1 0 1 1 % C7.2.347 UMINP
0 x 1 0 1 1 1 0 s s 1 ... 1 | 1 0 1 0 1 0 % C7.2.348 UMINV
0 x 1 0 1 1 1 1 s s i ... _ | 0 0 1 0 h 0 % C7.2.349 UMLAL, UMLAL2 (by element)
0 x 1 0 1 1 1 0 s s 1 ... _ | 1 0 0 0 0 0 % C7.2.350 UMLAL, UMLAL2 (vector)
0 x 1 0 1 1 1 1 s s i ... _ | 0 1 1 0 h 0 % C7.2.351 UMLSL, UMLSL2 (by element)
0 x 1 0 1 1 1 0 s s 1 ... _ | 1 0 1 0 0 0 % C7.2.352 UMLSL, UMLSL2 (vector)
0 x 0 0 1 1 1 0 0 0 0 ... _ | 0 0 1 1 1 1 % C7.2.353 UMOV
0 x 1 0 1 1 1 1 s s i ... _ | 1 0 1 0 h 0 % C7.2.354 UMULL, UMULL2 (by element)
0 x 1 0 1 1 1 0 s s 1 ... _ | 1 1 0 0 0 0 % C7.2.355 UMULL, UMULL2 (vector)
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 0 0 0 1 1 % C7.2.356 UQADD
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 1 0 1 1 1 % C7.2.357 UQRSHL
0 x 1 0 1 1 1 1 0 _ _ ... _ | 1 0 0 1 1 1 % C7.2.358 UQRSHRN, UQRSHRN2
0 x 1 i 1 1 1 1 0 _ _ ... _ | 0 1 1 1 0 1 % C7.2.359 UQSHL (immediate)
0 x 1 i 1 1 1 0 s s 1 ... _ | 0 1 0 0 1 1 % C7.2.360 UQSHL (register)
0 x 1 0 1 1 1 1 1 0 _ _ ... _ | 1 0 0 1 0 1 % C7.2.361 UQSHRN, UQSHRN2
0 x 1 i 1 1 1 0 s s 1 ... _ | 0 0 1 0 1 1 % C7.2.362 UQSUB - Unsigned saturating Subtra
ct.
0 x 1 i 1 1 1 0 s s 1 ... 1 | 0 1 0 0 1 0 % C7.2.363 UQXTN, UQXTN2 - Unsigned saturatin
g extract Narrow.
0 x 0 0 1 1 1 0 1 s 1 ... 1 | 1 1 0 0 1 0 % C7.2.364 URECPE - Unsigned Reciprocal Estim
ate.
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 0 0 1 0 1 % C7.2.365 URHADD Unsigned Rounding Halving A
dd.
0 x 1 i 1 1 1 0 s s 1 ... _ | 0 1 0 1 0 1 % C7.2.366 URSHL - Unsigned Rounding Shift Le
ft (register).
0 1 1 1 1 1 1 1 0 _ _ ... _ | 0 0 1 0 0 1 % C7.2.367 URSHR - Unsigned Rounding Shift Ri
ght (immediate).
0 x 1 0 1 1 1 0 1 s 1 ... 1 | 1 1 0 0 1 0 % C7.2.368 URSQRTE - Unsigned Reciprocal Squa
re Root Estimate.
0 1 1 1 1 1 1 1 0 _ _ ... _ | 0 0 1 1 0 1 % C7.2.369 URSRA
0 1 1 1 1 1 1 1 0 s s 1 ... _ | 0 1 0 0 0 1 % C7.2.370 USHL - Unsigned Shift Left (regist
er).
0 x 1 0 1 1 1 1 0 _ _ ... 0 | 1 0 1 0 0 1 % C7.2.371 USHLL, USHLL2 % C7.2.377 UXTL, UXT
L2
0 1 1 1 1 1 1 1 0 _ _ ... _ | 0 0 0 0 0 1 % C7.2.372 USHR - Unsigned Shift Right (immed
iate).
0 x 1 s 1 1 1 0 s s 1 ... 0 | 0 0 0 1 1 1 % C7.2.373 USQADD
0 1 1 1 1 1 1 1 0 _ _ ... _ | 0 0 0 1 0 1 % C7.2.374 USRA
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 0 1 0 0 0 % C7.2.375 USUBL, USUBL2
0 x 1 0 1 1 1 0 s s 1 ... _ | 0 0 1 1 0 0 % C7.2.376 USUBW, USUBW2 - Unsigned Subtract
Wide.
0 x 0 0 1 1 1 0 s s 0 ... _ | 0 0 0 1 1 0 % C7.2.378 UZP1 - Unzip vectors (primary).
```

```
0 x 0 0 1 1 1 0 s s 0 ... _ | 0 1 0 1 1 0 % C7.2.379 UZP2 - Unzip vectors (secondary).
1 1 0 0 1 1 1 0 1 0 0 ... _ | _ _ _ _ _ _ % C7.2.380 XAR - Exclusive OR and Rotate.
0 x 0 0 1 1 1 0 s s 1 ... 1 | 0 0 1 0 1 0 % C7.2.381 XTN, XTN2 - Extract Narrow.
0 x 0 0 1 1 1 0 s s 0 ... _ | 0 0 1 1 1 0 % C7.2.382 ZIP1 - Zip vectors (primary).
0 x 0 0 1 1 1 0 s s 0 ... _ | 0 1 1 1 1 0 % C7.2.383 ZIP2 - Zip vectors (secondary).
```

常用汇编指令

赋值

TODO：

下面多个帖子

MOV系列

- 【已解决】ARM汇编指令：FMOV
- 【已解决】ARM汇编指令：MOVK、MOVZ、MOVN

位操作

- 【已解决】ARM汇编指令：ubfx

涉及状态寄存器

- 【已解决】ARM汇编指令：MRS

内存操作

TODO：

下面多个帖子

- 【已解决】ARM汇编指令：LDR、STR
- 【已解决】ARM汇编指令：LDP、STP

Load

- 【已解决】ARM汇编指令：STUR
- 【基本解决】ARM汇编指令：stlxr

Store

- 【已解决】ARM汇编指令：LDUR

比较

TODO：

- 【已解决】ARM汇编指令：CMP、CMN
- 【已解决】ARM汇编指令：FCMP
- 【已解决】ARM汇编指令：TBZ和TBNZ

分支跳转

- 【已解决】ARM汇编指令：跳转指令 分支指令
  - 举例
    - 【整理】iOS逆向调试心得：Block的invoke函数调用
      - 常用 blr 跳转
- 【已解决】ARM汇编指令：br
- 【已解决】ARM汇编指令：bl
- 【已解决】ARM汇编指令：CBZ、CBNZ

条件选择

- 【已解决】ARM汇编指令：CSEL

寻址

- 【已解决】ARM汇编指令：ADR、ADRP

算数运算

TODO：

- 【已解决】ARM汇编指令：FMUL

逻辑运算

TODO：

- 【已解决】ARM汇编指令：AND
- 【已解决】ARM指令中ASR算术右移和LSR逻辑右移的区别

SVC系统调用

- SVC 0x80
  - 【已解决】32位ARM中SVC指令编码格式定义
  - 【记录】在线网站ARM汇编指令编码转换二进制值
  - 【未解决】IDA搜ARM汇编指令svc 0x80二进制值01 10 00 D4为何会搜出DCB 1
  - 【已解决】ARM汇编代码指令SVC 0x80转换成二进制编码值
  - 【已解决】ARM二进制汇编指令编码值01 10 00 D4和SVC指令编码
  - 【已解决】ARM64中为何SVC 0x80指令集编码二进制值是011000D4
  - 【未解决】IDA中如何查看汇编代码指令对应的二进制编码值
  -
  - 【整理】syscall内核系统调用和svc 0x80相关基础知识

其他

- 【整理】ARM中的DCB指令
- 【已解决】ARM汇编指令：FCVTZS
-

ARM常见用法

函数调用

- 【待整理】ARM64 函数调用 栈操作 栈帧
- 【未解决】ARM汇编指令：STP开辟栈空间即入栈和出栈相关

跳转指令

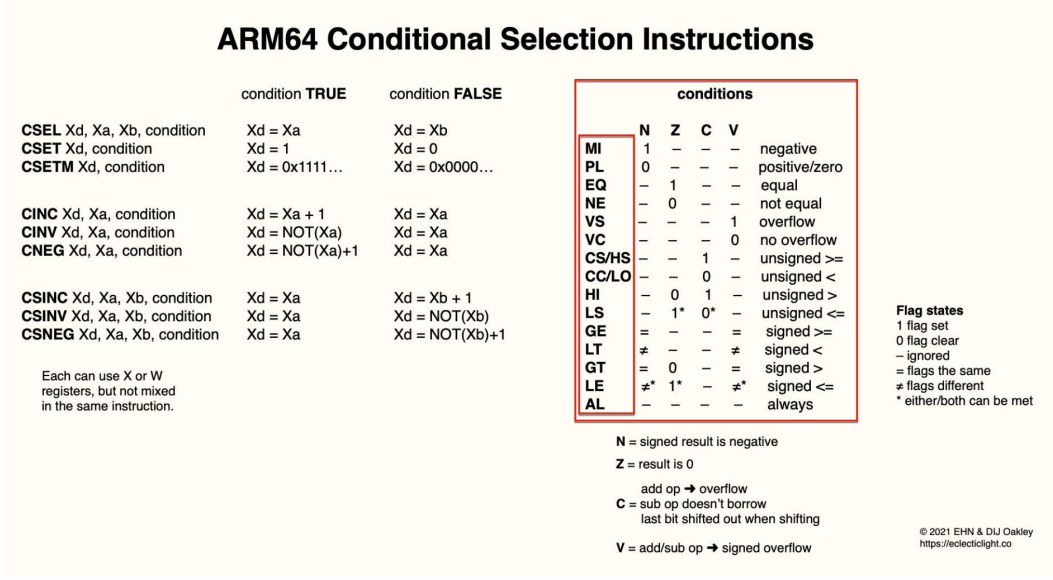
TODO：

【已解决】ARM汇编指令：TBZ和TBNZ

条件执行

- 【已解决】ARM汇编指令通用逻辑：条件执行Conditional execution
- 【基本解决】ARM指令中Carry的C的标志位的含义
- 【已解决】ARM指令AND如何影响Carry的C标志位

cond条件码

- `condition` = 条件码 = `cond` = `condition code field`
  - 来源
    - ARM中，有各种 `conditional selection instruction` 去 `Conditional execution` 中的：
      - `condition` = 条件码 =缩写为： `cond`
    - 指令语法中表示为： `{cond}`
  - 如何计算出 `cond` 的值：
    - `NZCV` 的组合值（确切的说是， `4` 个 `bit` 中某个或某些 `bit` 值）决定了 `cond` 的值
      - `NZCV` 的值
        - ARM32： `CPSR` 中的 `NZCV` 4个bit位（的组合的值）
        - ARM64：特殊寄存器： `NZCV`
  - 具体含义逻辑
    - ARM64 Conditional Selection Instructions
    - 

    - ARM64的条件码表格

Suffix	Flags	Meaning
EQ	Z set	Equal
NE	Z clear	Not equal
CS/HS	C set	Higher or same (unsigned >=)
CC/LO	C clear	Lower (unsigned <)
MI	N set	Negative
PL	N clear	Positive or zero
VS	V set	Overflow
VC	V clear	No overflow
HI	C set and Z clear	Higher (unsigned <=)
LS	C clear or Z set	Lower or same (unsigned <=)
GE	N and V the same	Signed >=
LT	N and V different	Signed <
GT	Z clear, and N and V the same	Signed >
LE	Z set, or N and V different	Signed <=
AL	Any	Always (usually omitted)

用法举例

b.le

此处的指令：

```
libobjc.A.dylib`objc_msgSend:
    0x1921c7b20 <+0> :   cmp    x0, #0x0                  ; =0x0
->  0x1921c7b24 <+4> :   b.le   0x1921c7b98               ; <+120>
```

其中前面的指令：

```
0x1921c7b20 <+0> :   cmp    x0, #0x0                  ; =0x0
```

由于x0中的值是：`0x00000002820463a0`

-》已经设置了`CPSR`中的：`C = Carry = 1`

然后再去运行：

```
b.le   0x1921c7b98
```

- `b.le` == `B指令` + `条件执行的cond是LE`

ARM Developer Suite Assembler Guide

```
LE
```

```
Z set, or N and V different
Signed <=
```

即：

如果之前的CPSR的条件中，满足：

`Z set, or N and V different`

即，逻辑上表示之前的判断是：`<=`，那么就跳转

而此处：

- N=0
- Z=0
- C=1
- V=0

->

- Z是0，不符合
- N和V都是0 -》 一样，不符合

所以，不跳转

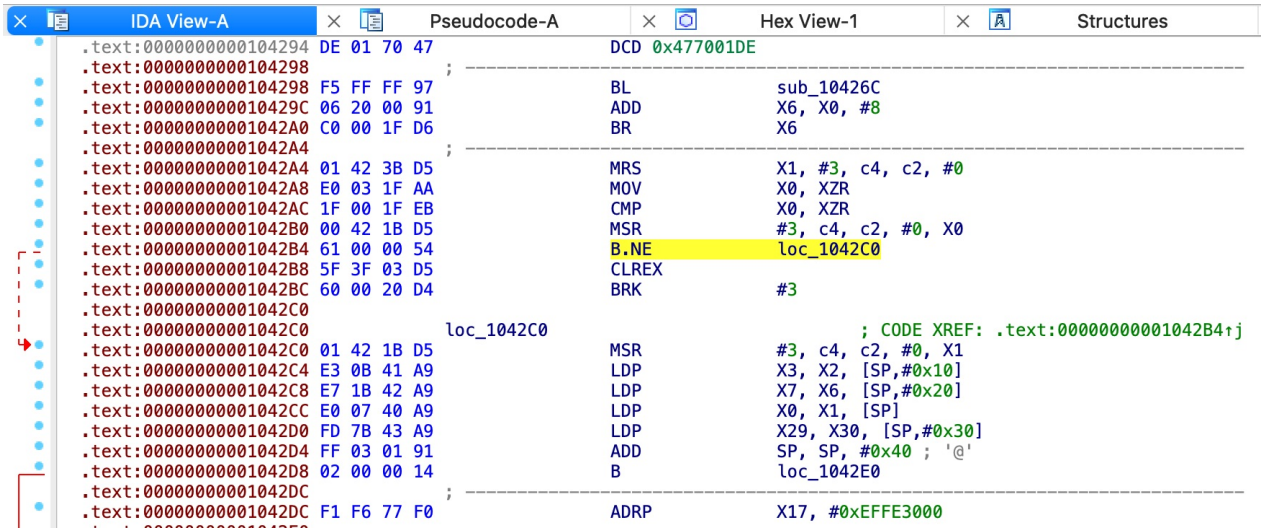
所以单步运行后：

程序没有b去跳转，而是继续执行下一行指令，是符合预期的。

B.NE的固定的间接跳转

IDA中代码：

```
.text:00000000001042A8 E0 03 1F AA                MOV             X0, XZR
.text:00000000001042AC 1F 00 1F EB                CMP             X0, XZR
.text:00000000001042B0 00 42 1B D5                msr nzcv, x0
.text:00000000001042B4 61 00 00 54                B.NE            loc_1042C0
```



中的 `B.NE` 中的 `NE`

这部分代码的总体逻辑是：

- 通过XZR给X0赋值0
- 比较X0和XZR，由于都是0，所以肯定相等
- 把X0=0，写入NZCV条件码状态寄存器
- B.NE的判断逻辑是： `NE = Not Equal = 不相等` ，即： `Z=0` ，此处的确是 `Z=0` ，符合条件，所以肯定会跳转到0x1042C0的代码位置

从而实现了：

固定的代码跳转逻辑： `B.NE` 是100%会跳转到此处0x1042C0的位置的

-》即：一种特殊的间接跳转的实现方式

Pre-index和Post-index

- 【已解决】ARM指令中的LDP指令的Pre-index和Post-index
- 【已解决】ARM汇编指令STP后面的感叹号的含义

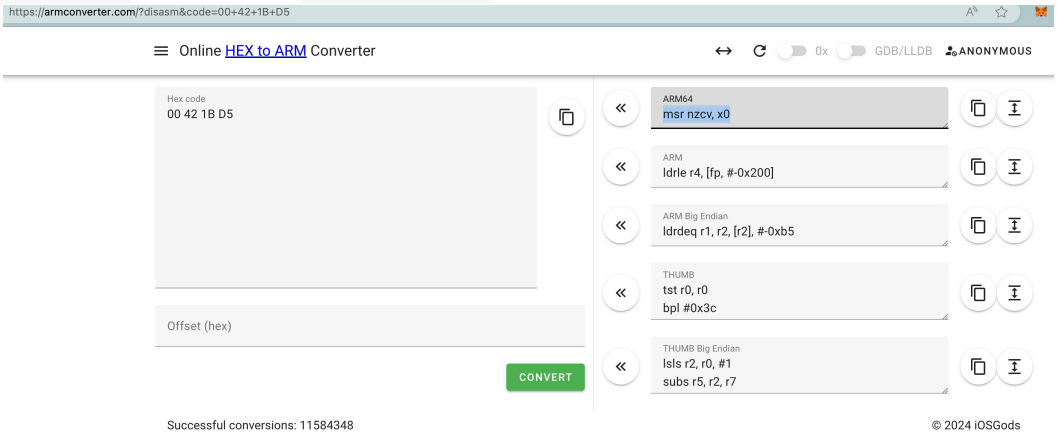
flexible second operand

- 【已解决】ARM汇编指令通用逻辑：Operand2灵活的第二个操作数flexible second operand

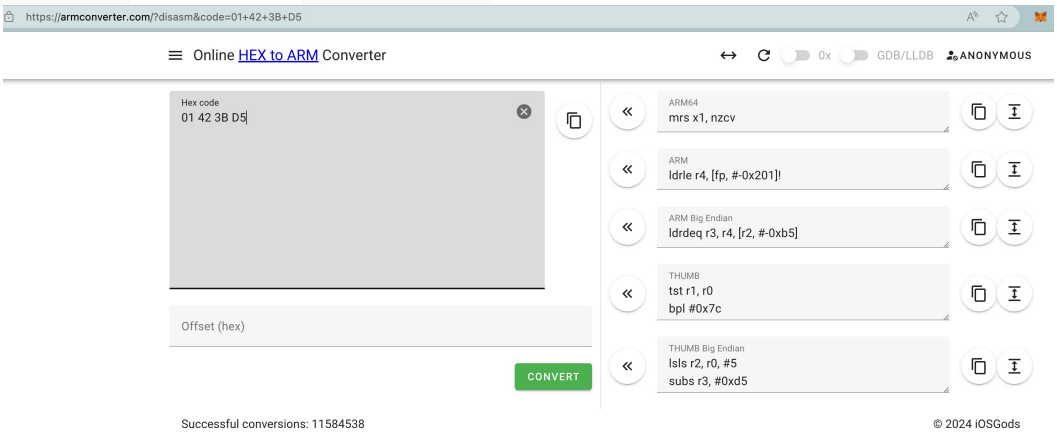
工具

指令和二进制opcode互转

- ARM 指令 在线 生成 转换
  - Online ARM to HEX Converter (armconverter.com)
    - `00 42 1B D5` -> ARM64的: `msr nzcv, x0`



    - `01 42 3B D5` -> ARM64的: `mrs x1, nzcv`



附录

下面列出相关参考资料。

X86汇编

与 `ARM` 相对应的 `X86` 架构

对应的也有各种内容，包括：

X86 调用规范 = 调用约定 = Calling Convention

- x86 Registers=X86寄存器
  -
- Stack during Subroutine Call = 子函数调用时的堆栈

。

参考资料

- 【已解决】32位ARM指令中cond即condition code field定义和语法
- 【已解决】ARM汇编指令：cond条件码的来源
- 【已解决】ARM汇编指令通用逻辑：条件执行Conditional execution条件码cond
- 【记录】ARM寄存器：nzcv
- 【整理】ARM汇编基础知识：寄存器命名叫法 20220531
- 【已解决】ARM中的寄存器 概述 概览 AAPCS
-
- Guide to x86 Assembly (virginia.edu)
- How does the ARM architecture differ from x86? - Stack Overflow
- ARM (groupoid.github.io)
- ARM Developer Suite Assembler Guide
- Online ARM to HEX Converter (armconverter.com)
- 一文搞懂 ARM 64 系列: ADC - chaoguo1234 - 博客园 (cnblogs.com)
- Arm A-profile A64 Instruction Set Architecture
- Arm A-profile A64 Instruction Set Architecture
- prominent features – Capstone – The Ultimate Disassembler (capstone-engine.org)
- Wayback Machine (archive.org)
- Unicorn-Engine-Documentation/Unicorn-Engine Documentation.md at master · kabeor/Unicorn-Engine-Documentation (github.com)
- ARM Procedure Call Standard
- Overview of ARM ABI Conventions | Microsoft Learn
- assembly - What registers to save in the ARM C calling convention? - Stack Overflow
- Registers in AArch64 state - Arm Compiler armasm User Guide
- Arm Compiler armasm User Guide - Program Counter in AArch64 state
- ARM Developer Suite Developer Guide
- Procedure Call Standard for the Arm Architecture - ABI 2020Q2 documentation
-