



Arm Neon Intrinsics

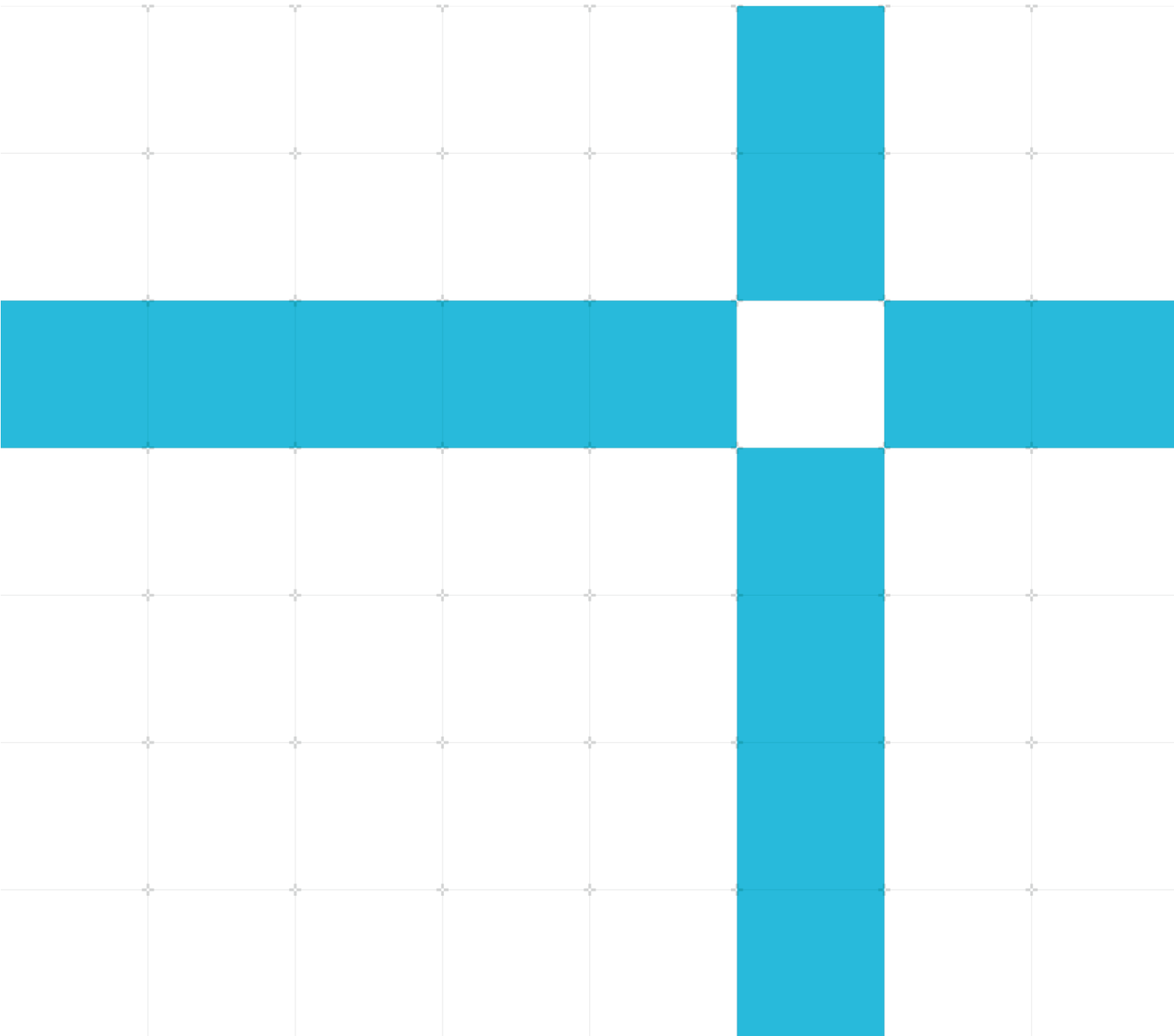
Reference for ACLE Q2 2020

Non-Confidential

Copyright © 2014 - 2020 Arm Limited (or its affiliates).
All rights reserved.

Issue F

IHI 0073F



Arm Neon Intrinsics

Reference

Copyright © 2014 - 2020 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
A	09 May 2014	Non-Confidential	First release
B	24 March 2016	Non-Confidential	Updated for ARMv8.1
C	30 March 2019	Non-Confidential	Version ACLE Q1 2019.
D	30 June 2019	Non-Confidential	Version ACLE Q2 2019.
E	30 September 2019	Non-Confidential	Version ACLE Q3 2019
F	30 May 2020	Non-Confidential	Version ACLE Q2 2020

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © 2014 - 2020 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is final, that is for a developed product.

Web Address

<http://www.arm.com>

About this document

This document is complementary to the main Arm C Language Extensions (ACLE) specification, which can be found on developer.arm.com.

List of Intrinsics

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x8_t vadd_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	ADD Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vaddq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	ADD Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vadd_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	ADD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vaddq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	ADD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vadd_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	ADD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vaddq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	ADD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int64x1_t vadd_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	ADD Dd,Dn,Dm	Dd -> result	v7/A32/A64
int64x2_t vaddq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	ADD Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
uint8x8_t vadd_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	ADD Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vaddq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	ADD Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vadd_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	ADD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vaddq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	ADD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vadd_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	ADD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vaddq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	ADD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vadd_u64(uint64x1_t a, uint64x1_t b)	a -> Dn b -> Dm	ADD Dd,Dn,Dm	Dd -> result	v7/A32/A64
uint64x2_t vaddq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	ADD Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
float32x2_t vadd_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FADD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vaddq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FADD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float64x1_t vadd_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FADD Dd,Dn,Dm	Dd -> result	A64
float64x2_t vaddq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FADD Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
int64_t vadd_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	ADD Dd,Dn,Dm	Dd -> result	A64
uint64_t vadd_u64(uint64_t a, uint64_t b)	a -> Dn b -> Dm	ADD Dd,Dn,Dm	Dd -> result	A64
int16x8_t vaddl_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SADDL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
int32x4_t vaddl_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SADDL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vaddl_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SADDL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
uint16x8_t vaddl_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UADDL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
uint32x4_t vaddl_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UADDL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
uint64x2_t vaddl_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UADDL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
int16x8_t vaddl_high_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SADDL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
int32x4_t vaddl_high_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SADDL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
int64x2_t vaddl_high_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SADDL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
uint16x8_t vaddl_high_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UADDL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
uint32x4_t vaddl_high_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UADDL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
uint64x2_t vaddl_high_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UADDL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t vaddw_s8(int16x8_t a, int8x8_t b)	a -> Vn.8H b -> Vm.8B	SADDW Vd.8H,Vn.8H,Vm.8B	Vd.8H -> result	v7/A32/A64
int32x4_t vaddw_s16(int32x4_t a, int16x4_t b)	a -> Vn.4S b -> Vm.4H	SADDW Vd.4S,Vn.4S,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vaddw_s32(int64x2_t a, int32x2_t b)	a -> Vn.2D b -> Vm.2S	SADDW Vd.2D,Vn.2D,Vm.2S	Vd.2D -> result	v7/A32/A64
uint16x8_t vaddw_u8(uint16x8_t a, uint8x8_t b)	a -> Vn.8H b -> Vm.8B	UADDW Vd.8H,Vn.8H,Vm.8B	Vd.8H -> result	v7/A32/A64
uint32x4_t vaddw_u16(uint32x4_t a, uint16x4_t b)	a -> Vn.4S b -> Vm.4H	UADDW Vd.4S,Vn.4S,Vm.4H	Vd.4S -> result	v7/A32/A64
uint64x2_t vaddw_u32(uint64x2_t a, uint32x2_t b)	a -> Vn.2D b -> Vm.2S	UADDW Vd.2D,Vn.2D,Vm.2S	Vd.2D -> result	v7/A32/A64
int16x8_t vaddw_high_s8(int16x8_t a, int8x16_t b)	a -> Vn.8H b -> Vm.16B	SADDW2 Vd.8H,Vn.8H,Vm.16B	Vd.8H -> result	A64
int32x4_t vaddw_high_s16(int32x4_t a, int16x8_t b)	a -> Vn.4S b -> Vm.8H	SADDW2 Vd.4S,Vn.4S,Vm.8H	Vd.4S -> result	A64
int64x2_t vaddw_high_s32(int64x2_t a, int32x4_t b)	a -> Vn.2D b -> Vm.4S	SADDW2 Vd.2D,Vn.2D,Vm.4S	Vd.2D -> result	A64
uint16x8_t vaddw_high_u8(uint16x8_t a, uint8x16_t b)	a -> Vn.8H b -> Vm.16B	UADDW2 Vd.8H,Vn.8H,Vm.16B	Vd.8H -> result	A64
uint32x4_t vaddw_high_u16(uint32x4_t a, uint16x8_t b)	a -> Vn.4S b -> Vm.8H	UADDW2 Vd.4S,Vn.4S,Vm.8H	Vd.4S -> result	A64
uint64x2_t vaddw_high_u32(uint64x2_t a, uint32x4_t b)	a -> Vn.2D b -> Vm.4S	UADDW2 Vd.2D,Vn.2D,Vm.4S	Vd.2D -> result	A64
int8x8_t vhadd_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SHADD Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vhaddq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SHADD Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vhadd_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SHADD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vhaddq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SHADD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vhadd_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SHADD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vhaddq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SHADD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vhadd_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UHADD Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vhaddq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UHADD Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vhadd_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UHADD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vhaddq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UHADD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vhadd_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UHADD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vhaddq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UHADD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int8x8_t vrhadd_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SRHADD Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vrhaddq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SRHADD Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vrhadd_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SRHADD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vrhaddq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SRHADD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vrhadd_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SRHADD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vrhaddq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SRHADD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vrhadd_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	URHADD Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vrhaddq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	URHADD Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vrhadd_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	URHADD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vrhaddq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	URHADD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vrhadd_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	URHADD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vrhaddq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	URHADD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x8_t vqadd_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SQADD Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vqaddq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SQADD Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vqadd_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SQADD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vqaddq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SQADD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vqadd_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SQADD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vqaddq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SQADD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int64x1_t vqadd_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	SQADD Dd,Dn,Dm	Dd -> result	v7/A32/A64
int64x2_t vqaddq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	SQADD Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
uint8x8_t vqadd_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UQADD Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vqaddq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UQADD Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vqadd_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UQADD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vqaddq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UQADD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vqadd_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UQADD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vqaddq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UQADD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vqadd_u64(uint64x1_t a, uint64x1_t b)	a -> Dn b -> Dm	UQADD Dd,Dn,Dm	Dd -> result	v7/A32/A64
uint64x2_t vqaddq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	UQADD Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
int8_t vqaddb_s8(int8_t a, int8_t b)	a -> Bn b -> Bm	SQADD Bd,Bn,Bm	Bd -> result	A64
int16_t vqaddh_s16(int16_t a, int16_t b)	a -> Hn b -> Hm	SQADD Hd,Hn,Hm	Hd -> result	A64
int32_t vqadds_s32(int32_t a, int32_t b)	a -> Sn b -> Sm	SQADD Sd,Sn,Sm	Sd -> result	A64
int64_t vqaddd_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	SQADD Dd,Dn,Dm	Dd -> result	A64
uint8_t vqaddb_u8(uint8_t a, uint8_t b)	a -> Bn b -> Bm	UQADD Bd,Bn,Bm	Bd -> result	A64
uint16_t vqaddh_u16(uint16_t a, uint16_t b)	a -> Hn b -> Hm	UQADD Hd,Hn,Hm	Hd -> result	A64
uint32_t vqadds_u32(uint32_t a, uint32_t b)	a -> Sn b -> Sm	UQADD Sd,Sn,Sm	Sd -> result	A64
uint64_t vqaddd_u64(uint64_t a, uint64_t b)	a -> Dn b -> Dm	UQADD Dd,Dn,Dm	Dd -> result	A64
int8x8_t vuqadd_s8(int8x8_t a, uint8x8_t b)	a -> Vd.8B b -> Vn.8B	SUQADD Vd.8B,Vn.8B	Vd.8B -> result	A64
int8x16_t vuqaddq_s8(int8x16_t a, uint8x16_t b)	a -> Vd.16B b -> Vn.16B	SUQADD Vd.16B,Vn.16B	Vd.16B -> result	A64
int16x4_t vuqadd_s16(int16x4_t a, uint16x4_t b)	a -> Vd.4H b -> Vn.4H	SUQADD Vd.4H,Vn.4H	Vd.4H -> result	A64
int16x8_t vuqaddq_s16(int16x8_t a, uint16x8_t b)	a -> Vd.8H b -> Vn.8H	SUQADD Vd.8H,Vn.8H	Vd.8H -> result	A64
int32x2_t vuqadd_s32(int32x2_t a, uint32x2_t b)	a -> Vd.2S b -> Vn.2S	SUQADD Vd.2S,Vn.2S	Vd.2S -> result	A64
int32x4_t vuqaddq_s32(int32x4_t a, uint32x4_t b)	a -> Vd.4S b -> Vn.4S	SUQADD Vd.4S,Vn.4S	Vd.4S -> result	A64
int64x1_t vuqadd_s64(int64x1_t a, uint64x1_t b)	a -> Dd b -> Dn	SUQADD Dd,Dn	Dd -> result	A64
int64x2_t vuqaddq_s64(int64x2_t a, uint64x2_t b)	a -> Vd.2D b -> Vn.2D	SUQADD Vd.2D,Vn.2D	Vd.2D -> result	A64
int8_t vuqaddb_s8(int8_t a, uint8_t b)	a -> Bd b -> Bn	SUQADD Bd,Bn	Bd -> result	A64
int16_t vuqaddh_s16(int16_t a, uint16_t b)	a -> Hd b -> Hn	SUQADD Hd,Hn	Hd -> result	A64
int32_t vuqadds_s32(int32_t a, uint32_t b)	a -> Sd b -> Sn	SUQADD Sd,Sn	Sd -> result	A64
int64_t vuqaddd_s64(int64_t a, uint64_t b)	a -> Dd b -> Dn	SUQADD Dd,Dn	Dd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x8_t vsqadd_u8(uint8x8_t a, int8x8_t b)	a -> Vd.8B b -> Vn.8B	USQADD Vd.8B,Vn.8B	Vd.8B -> result	A64
uint8x16_t vsqaddq_u8(uint8x16_t a, int8x16_t b)	a -> Vd.16B b -> Vn.16B	USQADD Vd.16B,Vn.16B	Vd.16B -> result	A64
uint16x4_t vsqadd_u16(uint16x4_t a, int16x4_t b)	a -> Vd.4H b -> Vn.4H	USQADD Vd.4H,Vn.4H	Vd.4H -> result	A64
uint16x8_t vsqaddq_u16(uint16x8_t a, int16x8_t b)	a -> Vd.8H b -> Vn.8H	USQADD Vd.8H,Vn.8H	Vd.8H -> result	A64
uint32x2_t vsqadd_u32(uint32x2_t a, int32x2_t b)	a -> Vd.2S b -> Vn.2S	USQADD Vd.2S,Vn.2S	Vd.2S -> result	A64
uint32x4_t vsqaddq_u32(uint32x4_t a, int32x4_t b)	a -> Vd.4S b -> Vn.4S	USQADD Vd.4S,Vn.4S	Vd.4S -> result	A64
uint64x1_t vsqadd_u64(uint64x1_t a, int64x1_t b)	a -> Dd b -> Dn	USQADD Dd,Dn	Dd -> result	A64
uint64x2_t vsqaddq_u64(uint64x2_t a, int64x2_t b)	a -> Vd.2D b -> Vn.2D	USQADD Vd.2D,Vn.2D	Vd.2D -> result	A64
uint8_t vsqaddb_u8(uint8_t a, int8_t b)	a -> Bd b -> Bn	USQADD Bd,Bn	Bd -> result	A64
uint16_t vsqaddh_u16(uint16_t a, int16_t b)	a -> Hd b -> Hn	USQADD Hd,Hn	Hd -> result	A64
uint32_t vsqadds_u32(uint32_t a, int32_t b)	a -> Sd b -> Sn	USQADD Sd,Sn	Sd -> result	A64
uint64_t vsqadddd_u64(uint64_t a, int64_t b)	a -> Dd b -> Dn	USQADD Dd,Dn	Dd -> result	A64
int8x8_t vaddhn_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	ADDHN Vd.8B,Vn.8H,Vm.8H	Vd.8B -> result	v7/A32/A64
int16x4_t vaddhn_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	ADDHN Vd.4H,Vn.4S,Vm.4S	Vd.4H -> result	v7/A32/A64
int32x2_t vaddhn_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	ADDHN Vd.2S,Vn.2D,Vm.2D	Vd.2S -> result	v7/A32/A64
uint8x8_t vaddhn_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	ADDHN Vd.8B,Vn.8H,Vm.8H	Vd.8B -> result	v7/A32/A64
uint16x4_t vaddhn_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	ADDHN Vd.4H,Vn.4S,Vm.4S	Vd.4H -> result	v7/A32/A64
uint32x2_t vaddhn_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	ADDHN Vd.2S,Vn.2D,Vm.2D	Vd.2S -> result	v7/A32/A64
int8x16_t vaddhn_high_s16(int8x8_t r, int16x8_t a, int16x8_t b)	r -> Vd.8B a -> Vn.8H b -> Vm.8H	ADDHN2 Vd.16B,Vn.8H,Vm.8H	Vd.16B -> result	A64
int16x8_t vaddhn_high_s32(int16x4_t r, int32x4_t a, int32x4_t b)	r -> Vd.4H a -> Vn.4S b -> Vm.4S	ADDHN2 Vd.8H,Vn.4S,Vm.4S	Vd.8H -> result	A64
int32x4_t vaddhn_high_s64(int32x2_t r, int64x2_t a, int64x2_t b)	r -> Vd.2S a -> Vn.2D b -> Vm.2D	ADDHN2 Vd.4S,Vn.2D,Vm.2D	Vd.4S -> result	A64
uint8x16_t vaddhn_high_u16(uint8x8_t r, uint16x8_t a, uint16x8_t b)	r -> Vd.8B a -> Vn.8H b -> Vm.8H	ADDHN2 Vd.16B,Vn.8H,Vm.8H	Vd.16B -> result	A64
uint16x8_t vaddhn_high_u32(uint16x4_t r, uint32x4_t a, uint32x4_t b)	r -> Vd.4H a -> Vn.4S b -> Vm.4S	ADDHN2 Vd.8H,Vn.4S,Vm.4S	Vd.8H -> result	A64
uint32x4_t vaddhn_high_u64(uint32x2_t r, uint64x2_t a, uint64x2_t b)	r -> Vd.2S a -> Vn.2D b -> Vm.2D	ADDHN2 Vd.4S,Vn.2D,Vm.2D	Vd.4S -> result	A64
int8x8_t vraddhn_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	RADDHN Vd.8B,Vn.8H,Vm.8H	Vd.8B -> result	v7/A32/A64
int16x4_t vraddhn_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	RADDHN Vd.4H,Vn.4S,Vm.4S	Vd.4H -> result	v7/A32/A64
int32x2_t vraddhn_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	RADDHN Vd.2S,Vn.2D,Vm.2D	Vd.2S -> result	v7/A32/A64
uint8x8_t vraddhn_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	RADDHN Vd.8B,Vn.8H,Vm.8H	Vd.8B -> result	v7/A32/A64
uint16x4_t vraddhn_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	RADDHN Vd.4H,Vn.4S,Vm.4S	Vd.4H -> result	v7/A32/A64
uint32x2_t vraddhn_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	RADDHN Vd.2S,Vn.2D,Vm.2D	Vd.2S -> result	v7/A32/A64
int8x16_t vraddhn_high_s16(int8x8_t r, int16x8_t a, int16x8_t b)	r -> Vd.8B a -> Vn.8H b -> Vm.8H	RADDHN2 Vd.16B,Vn.8H,Vm.8H	Vd.16B -> result	A64
int16x8_t vraddhn_high_s32(int16x4_t r, int32x4_t a, int32x4_t b)	r -> Vd.4H a -> Vn.4S b -> Vm.4S	RADDHN2 Vd.8H,Vn.4S,Vm.4S	Vd.8H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t vraddhn_high_s64(int32x2_t r, int64x2_t a, int64x2_t b)	r -> Vd.2S a -> Vn.2D b -> Vm.2D	RADDHN2 Vd.4S,Vn.2D,Vm.2D	Vd.4S -> result	A64
uint8x16_t vraddhn_high_u16(uint8x8_t r, uint16x8_t a, uint16x8_t b)	r -> Vd.8B a -> Vn.8H b -> Vm.8H	RADDHN2 Vd.16B,Vn.8H,Vm.8H	Vd.16B -> result	A64
uint16x8_t vraddhn_high_u32(uint16x4_t r, uint32x4_t a, uint32x4_t b)	r -> Vd.4H a -> Vn.4S b -> Vm.4S	RADDHN2 Vd.8H,Vn.4S,Vm.4S	Vd.8H -> result	A64
uint32x4_t vraddhn_high_u64(uint32x2_t r, uint64x2_t a, uint64x2_t b)	r -> Vd.2S a -> Vn.2D b -> Vm.2D	RADDHN2 Vd.4S,Vn.2D,Vm.2D	Vd.4S -> result	A64
int8x8_t vmul_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	MUL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vmulq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	MUL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vmul_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	MUL Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vmulq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	MUL Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vmul_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	MUL Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vmulq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	MUL Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vmul_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	MUL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vmulq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	MUL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vmul_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	MUL Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vmulq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	MUL Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vmul_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	MUL Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vmulq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	MUL Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float32x2_t vmul_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FMUL Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vmulq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FMUL Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
poly8x8_t vmul_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	PMUL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
poly8x16_t vmulq_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	PMUL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
float64x1_t vmul_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FMUL Dd,Dn,Dm	Dd -> result	A64
float64x2_t vmulq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FMUL Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t vmulx_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FMULX Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
float32x4_t vmulxq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FMULX Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x1_t vmulx_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FMULX Dd,Dn,Dm	Dd -> result	A64
float64x2_t vmulxq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FMULX Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32_t vmulxs_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FMULX Sd,Sn,Sm	Sd -> result	A64
float64_t vmulxd_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FMULX Dd,Dn,Dm	Dd -> result	A64
float32x2_t vmulx_lane_f32(float32x2_t a, float32x2_t v, const int lane)	a -> Vn.2S v -> Vm.2S 0 <= lane <= 1	FMULX Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
float32x4_t vmulxq_lane_f32(float32x4_t a, float32x2_t v, const int lane)	a -> Vn.4S v -> Vm.2S 0 <= lane <= 1	FMULX Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
float64x1_t vmulx_lane_f64(float64x1_t a, float64x1_t v, const int lane)	a -> Dn v -> Vm.1D lane == 0	FMULX Dd,Dn,Vm.D[lane]	Dd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64x2_t vmulxq_lane_f64(float64x2_t a, float64x1_t v, const int lane)	a -> Vn.2D v -> Vm.1D lane == 0	FMULX Vd.2D,Vn.2D,Vm.D[lane]	Vd.2D -> result	A64
float32_t vmulxs_lane_f32(float32_t a, float32x2_t v, const int lane)	a -> Sn v -> Vm.2S 0 <= lane <= 1	FMULX Sd,Sn,Vm.S[lane]	Sd -> result	A64
float64_t vmulxd_lane_f64(float64_t a, float64x1_t v, const int lane)	a -> Dn v -> Vm.1D lane == 0	FMULX Dd,Dn,Vm.D[lane]	Dd -> result	A64
float32x2_t vmulx_laneq_f32(float32x2_t a, float32x4_t v, const int lane)	a -> Vn.2S v -> Vm.4S 0 <= lane <= 3	FMULX Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
float32x4_t vmulxq_laneq_f32(float32x4_t a, float32x4_t v, const int lane)	a -> Vn.4S v -> Vm.4S 0 <= lane <= 3	FMULX Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
float64x1_t vmulx_laneq_f64(float64x1_t a, float64x2_t v, const int lane)	a -> Dn v -> Vm.2D 0 <= lane <= 1	FMULX Dd,Dn,Vm.D[lane]	Dd -> result	A64
float64x2_t vmulxq_laneq_f64(float64x2_t a, float64x2_t v, const int lane)	a -> Vn.2D v -> Vm.2D 0 <= lane <= 1	FMULX Vd.2D,Vn.2D,Vm.D[lane]	Vd.2D -> result	A64
float32_t vmulxs_laneq_f32(float32_t a, float32x4_t v, const int lane)	a -> Sn v -> Vm.4S 0 <= lane <= 3	FMULX Sd,Sn,Vm.S[lane]	Sd -> result	A64
float64_t vmulxd_laneq_f64(float64_t a, float64x2_t v, const int lane)	a -> Dn v -> Vm.2D 0 <= lane <= 1	FMULX Dd,Dn,Vm.D[lane]	Dd -> result	A64
float32x2_t vdiv_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FDIV Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
float32x4_t vdivq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FDIV Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x1_t vdiv_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FDIV Dd,Dn,Dm	Dd -> result	A64
float64x2_t vdivq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FDIV Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
int8x8_t vmla_s8(int8x8_t a, int8x8_t b, int8x8_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	MLA Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vmla_s8(int8x16_t a, int8x16_t b, int8x16_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	MLA Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vmla_s16(int16x4_t a, int16x4_t b, int16x4_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.4H	MLA Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vmla_s16(int16x8_t a, int16x8_t b, int16x8_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.8H	MLA Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vmla_s32(int32x2_t a, int32x2_t b, int32x2_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.2S	MLA Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vmla_s32(int32x4_t a, int32x4_t b, int32x4_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.4S	MLA Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vmla_u8(uint8x8_t a, uint8x8_t b, uint8x8_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	MLA Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vmla_u8(uint8x16_t a, uint8x16_t b, uint8x16_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	MLA Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vmla_u16(uint16x4_t a, uint16x4_t b, uint16x4_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.4H	MLA Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vmla_u16(uint16x8_t a, uint16x8_t b, uint16x8_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.8H	MLA Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x2_t vmla_u32(uint32x2_t a, uint32x2_t b, uint32x2_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.2S	MLA Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vmlaq_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.4S	MLA Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float32x2_t vmla_f32(float32x2_t a, float32x2_t b, float32x2_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] + (b[i] * c[i]) for i = 0 to 1	N/A -> result	v7/A32/A64
float32x4_t vmlaq_f32(float32x4_t a, float32x4_t b, float32x4_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] + (b[i] * c[i]) for i = 0 to 3	N/A -> result	v7/A32/A64
float64x1_t vmla_f64(float64x1_t a, float64x1_t b, float64x1_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] + (b[i] * c[i]) for i = 0	N/A -> result	A64
float64x2_t vmlaq_f64(float64x2_t a, float64x2_t b, float64x2_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] + (b[i] * c[i]) for i = 0 to 1	N/A -> result	A64
int16x8_t vmlal_s8(int16x8_t a, int8x8_t b, int8x8_t c)	a -> Vd.8H b -> Vn.8B c -> Vm.8B	SMLAL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
int32x4_t vmlal_s16(int32x4_t a, int16x4_t b, int16x4_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.4H	SMLAL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vmlal_s32(int64x2_t a, int32x2_t b, int32x2_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.2S	SMLAL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
uint16x8_t vmlal_u8(uint16x8_t a, uint8x8_t b, uint8x8_t c)	a -> Vd.8H b -> Vn.8B c -> Vm.8B	UMLAL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
uint32x4_t vmlal_u16(uint32x4_t a, uint16x4_t b, uint16x4_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.4H	UMLAL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
uint64x2_t vmlal_u32(uint64x2_t a, uint32x2_t b, uint32x2_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.2S	UMLAL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
int16x8_t vmlal_high_s8(int16x8_t a, int8x16_t b, int8x16_t c)	a -> Vd.8H b -> Vn.16B c -> Vm.16B	SMLAL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
int32x4_t vmlal_high_s16(int32x4_t a, int16x8_t b, int16x8_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.8H	SMLAL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
int64x2_t vmlal_high_s32(int64x2_t a, int32x4_t b, int32x4_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.4S	SMLAL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
uint16x8_t vmlal_high_u8(uint16x8_t a, uint8x16_t b, uint8x16_t c)	a -> Vd.8H b -> Vn.16B c -> Vm.16B	UMLAL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
uint32x4_t vmlal_high_u16(uint32x4_t a, uint16x8_t b, uint16x8_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.8H	UMLAL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
uint64x2_t vmlal_high_u32(uint64x2_t a, uint32x4_t b, uint32x4_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.4S	UMLAL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
int8x8_t vmls_s8(int8x8_t a, int8x8_t b, int8x8_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	MLS Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vmlsq_s8(int8x16_t a, int8x16_t b, int8x16_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	MLS Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vmls_s16(int16x4_t a, int16x4_t b, int16x4_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.4H	MLS Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vmlsq_s16(int16x8_t a, int16x8_t b, int16x8_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.8H	MLS Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vmls_s32(int32x2_t a, int32x2_t b, int32x2_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.2S	MLS Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vmlsq_s32(int32x4_t a, int32x4_t b, int32x4_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.4S	MLS Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x8_t vmls_u8(uint8x8_t a, uint8x8_t b, uint8x8_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	MLS Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vmlsq_u8(uint8x16_t a, uint8x16_t b, uint8x16_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	MLS Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vmls_u16(uint16x4_t a, uint16x4_t b, uint16x4_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.4H	MLS Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vmlsq_u16(uint16x8_t a, uint16x8_t b, uint16x8_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.8H	MLS Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vmls_u32(uint32x2_t a, uint32x2_t b, uint32x2_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.2S	MLS Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vmlsq_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.4S	MLS Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float32x2_t vmls_f32(float32x2_t a, float32x2_t b, float32x2_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] - (b[i] * c[i]) for i = 0 to 1	N/A -> result	v7/A32/A64
float32x4_t vmlsq_f32(float32x4_t a, float32x4_t b, float32x4_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] - (b[i] * c[i]) for i = 0 to 3	N/A -> result	v7/A32/A64
float64x1_t vmls_f64(float64x1_t a, float64x1_t b, float64x1_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] - (b[i] * c[i]) for i = 0	N/A -> result	A64
float64x2_t vmlsq_f64(float64x2_t a, float64x2_t b, float64x2_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] - (b[i] * c[i]) for i = 0 to 1	N/A -> result	A64
int16x8_t vmlsl_s8(int16x8_t a, int8x8_t b, int8x8_t c)	a -> Vd.8H b -> Vn.8B c -> Vm.8B	SMLSL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
int32x4_t vmlsl_s16(int32x4_t a, int16x4_t b, int16x4_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.4H	SMLSL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vmlsl_s32(int64x2_t a, int32x2_t b, int32x2_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.2S	SMLSL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
uint16x8_t vmlsl_u8(uint16x8_t a, uint8x8_t b, uint8x8_t c)	a -> Vd.8H b -> Vn.8B c -> Vm.8B	UMLSL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
uint32x4_t vmlsl_u16(uint32x4_t a, uint16x4_t b, uint16x4_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.4H	UMLSL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
uint64x2_t vmlsl_u32(uint64x2_t a, uint32x2_t b, uint32x2_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.2S	UMLSL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
int16x8_t vmlsl_high_s8(int16x8_t a, int8x16_t b, int8x16_t c)	a -> Vd.8H b -> Vn.16B c -> Vm.16B	SMLSL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
int32x4_t vmlsl_high_s16(int32x4_t a, int16x8_t b, int16x8_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.8H	SMLSL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
int64x2_t vmlsl_high_s32(int64x2_t a, int32x4_t b, int32x4_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.4S	SMLSL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
uint16x8_t vmlsl_high_u8(uint16x8_t a, uint8x16_t b, uint8x16_t c)	a -> Vd.8H b -> Vn.16B c -> Vm.16B	UMLSL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
uint32x4_t vmlsl_high_u16(uint32x4_t a, uint16x8_t b, uint16x8_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.8H	UMLSL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
uint64x2_t vmlsl_high_u32(uint64x2_t a, uint32x4_t b, uint32x4_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.4S	UMLSL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
float32x2_t vfma_f32(float32x2_t a, float32x2_t b, float32x2_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.2S	FMLA Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vfmaq_f32(float32x4_t a, float32x4_t b, float32x4_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.4S	FMLA Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64x1_t vfma_f64(float64x1_t a, float64x1_t b, float64x1_t c)	a -> Da b -> Dn c -> Dm	FMADD Dd,Dn,Dm,Da	Dd -> result	A64
float64x2_t vfmaq_f64(float64x2_t a, float64x2_t b, float64x2_t c)	a -> Vd.2D b -> Vn.2D c -> Vm.2D	FMLA Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t vfma_lane_f32(float32x2_t a, float32x2_t b, float32x2_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	FMLA Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
float32x4_t vfmaq_lane_f32(float32x4_t a, float32x4_t b, float32x2_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	FMLA Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
float64x1_t vfma_lane_f64(float64x1_t a, float64x1_t b, float64x1_t v, const int lane)	a -> Dd b -> Dn v -> Vm.1D lane == 0	FMLA Dd,Dn,Vm.D[lane]	Dd -> result	A64
float64x2_t vfmaq_lane_f64(float64x2_t a, float64x2_t b, float64x1_t v, const int lane)	a -> Vd.2D b -> Vn.2D v -> Vm.1D lane == 0	FMLA Vd.2D,Vn.2D,Vm.D[lane]	Vd.2D -> result	A64
float32_t vfmas_lane_f32(float32_t a, float32_t b, float32x2_t v, const int lane)	a -> Sd b -> Sn v -> Vm.2S 0 <= lane <= 1	FMLA Sd,Sn,Vm.S[lane]	Sd -> result	A64
float64_t vfmad_lane_f64(float64_t a, float64_t b, float64x1_t v, const int lane)	a -> Dd b -> Dn v -> Vm.1D lane == 0	FMLA Dd,Dn,Vm.D[lane]	Dd -> result	A64
float32x2_t vfma_laneq_f32(float32x2_t a, float32x2_t b, float32x4_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	FMLA Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
float32x4_t vfmaq_laneq_f32(float32x4_t a, float32x4_t b, float32x4_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	FMLA Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
float64x1_t vfma_laneq_f64(float64x1_t a, float64x1_t b, float64x2_t v, const int lane)	a -> Dd b -> Dn v -> Vm.2D 0 <= lane <= 1	FMLA Dd,Dn,Vm.D[lane]	Dd -> result	A64
float64x2_t vfmaq_laneq_f64(float64x2_t a, float64x2_t b, float64x2_t v, const int lane)	a -> Vd.2D b -> Vn.2D v -> Vm.2D 0 <= lane <= 1	FMLA Vd.2D,Vn.2D,Vm.D[lane]	Vd.2D -> result	A64
float32_t vfmas_laneq_f32(float32_t a, float32_t b, float32x4_t v, const int lane)	a -> Sd b -> Sn v -> Vm.4S 0 <= lane <= 3	FMLA Sd,Sn,Vm.S[lane]	Sd -> result	A64
float64_t vfmad_laneq_f64(float64_t a, float64_t b, float64x2_t v, const int lane)	a -> Dd b -> Dn v -> Vm.2D 0 <= lane <= 1	FMLA Dd,Dn,Vm.D[lane]	Dd -> result	A64
float32x2_t vfms_f32(float32x2_t a, float32x2_t b, float32x2_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.2S	FMLS Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vfmsq_f32(float32x4_t a, float32x4_t b, float32x4_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.4S	FMLS Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float64x1_t vfms_f64(float64x1_t a, float64x1_t b, float64x1_t c)	a -> Da b -> Dn c -> Dm	FMSUB Dd,Dn,Dm,Da	Dd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64x2_t vfmsq_f64(float64x2_t a, float64x2_t b, float64x2_t c)	a -> Vd.2D b -> Vn.2D c -> Vm.2D	FMLS Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t vfms_lane_f32(float32x2_t a, float32x2_t b, float32x2_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	FMLS Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
float32x4_t vfmsq_lane_f32(float32x4_t a, float32x4_t b, float32x2_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	FMLS Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
float64x1_t vfms_lane_f64(float64x1_t a, float64x1_t b, float64x1_t v, const int lane)	a -> Dd b -> Dn v -> Vm.1D lane == 0	FMLS Dd,Dn,Vm.D[lane]	Dd -> result	A64
float64x2_t vfmsq_lane_f64(float64x2_t a, float64x2_t b, float64x1_t v, const int lane)	a -> Vd.2D b -> Vn.2D v -> Vm.1D lane == 0	FMLS Vd.2D,Vn.2D,Vm.D[lane]	Vd.2D -> result	A64
float32_t vfms_lane_f32(float32_t a, float32_t b, float32x2_t v, const int lane)	a -> Sd b -> Sn v -> Vm.2S 0 <= lane <= 1	FMLS Sd,Sn,Vm.S[lane]	Sd -> result	A64
float64_t vfmsd_lane_f64(float64_t a, float64_t b, float64x1_t v, const int lane)	a -> Dd b -> Dn v -> Vm.1D lane == 0	FMLS Dd,Dn,Vm.D[lane]	Dd -> result	A64
float32x2_t vfms_laneq_f32(float32x2_t a, float32x2_t b, float32x4_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	FMLS Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
float32x4_t vfmsq_laneq_f32(float32x4_t a, float32x4_t b, float32x4_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	FMLS Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
float64x1_t vfms_laneq_f64(float64x1_t a, float64x1_t b, float64x2_t v, const int lane)	a -> Dd b -> Dn v -> Vm.2D 0 <= lane <= 1	FMLS Dd,Dn,Vm.D[lane]	Dd -> result	A64
float64x2_t vfmsq_laneq_f64(float64x2_t a, float64x2_t b, float64x2_t v, const int lane)	a -> Vd.2D b -> Vn.2D v -> Vm.2D 0 <= lane <= 1	FMLS Vd.2D,Vn.2D,Vm.D[lane]	Vd.2D -> result	A64
float32_t vfms_laneq_f32(float32_t a, float32_t b, float32x4_t v, const int lane)	a -> Sd b -> Sn v -> Vm.4S 0 <= lane <= 3	FMLS Sd,Sn,Vm.S[lane]	Sd -> result	A64
float64_t vfmsd_laneq_f64(float64_t a, float64_t b, float64x2_t v, const int lane)	a -> Dd b -> Dn v -> Vm.2D 0 <= lane <= 1	FMLS Dd,Dn,Vm.D[lane]	Dd -> result	A64
int16x4_t vqdmulh_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SQDMULH Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vqdmulhq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SQDMULH Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vqdmulh_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SQDMULH Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vqdmulhq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SQDMULH Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int16_t vqdmulhh_s16(int16_t a, int16_t b)	a -> Hn b -> Hm	SQDMULH Hd,Hn,Hm	Hd -> result	A64
int32_t vqdmulhs_s32(int32_t a, int32_t b)	a -> Sn b -> Sm	SQDMULH Sd,Sn,Sm	Sd -> result	A64
int16x4_t vqrdmulh_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SQRDMULH Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t vqrdmulhq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SQRDMULH Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vqrdmulh_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SQRDMULH Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vqrdmulhq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SQRDMULH Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int16_t vqrdmulhh_s16(int16_t a, int16_t b)	a -> Hn b -> Hm	SQRDMULH Hd,Hn,Hm	Hd -> result	A64
int32_t vqrdmulhs_s32(int32_t a, int32_t b)	a -> Sn b -> Sm	SQRDMULH Sd,Sn,Sm	Sd -> result	A64
int32x4_t vqrdmlal_s16(int32x4_t a, int16x4_t b, int16x4_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.4H	SQDMLAL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vqrdmlal_s32(int64x2_t a, int32x2_t b, int32x2_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.2S	SQDMLAL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
int32_t vqrdmlalh_s16(int32_t a, int16_t b, int16_t c)	a -> Sd b -> Hn c -> Hm	SQDMLAL Sd,Hn,Hm	Sd -> result	A64
int64_t vqrdmlals_s32(int64_t a, int32_t b, int32_t c)	a -> Dd b -> Sn c -> Sm	SQDMLAL Dd,Sn,Sm	Dd -> result	A64
int32x4_t vqrdmlal_high_s16(int32x4_t a, int16x8_t b, int16x8_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.8H	SQDMLAL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
int64x2_t vqrdmlal_high_s32(int64x2_t a, int32x4_t b, int32x4_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.4S	SQDMLAL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
int32x4_t vqrdmlsl_s16(int32x4_t a, int16x4_t b, int16x4_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.4H	SQDMLSL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vqrdmlsl_s32(int64x2_t a, int32x2_t b, int32x2_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.2S	SQDMLSL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
int32_t vqrdmlslh_s16(int32_t a, int16_t b, int16_t c)	a -> Sd b -> Hn c -> Hm	SQDMLSL Sd,Hn,Hm	Sd -> result	A64
int64_t vqrdmlsls_s32(int64_t a, int32_t b, int32_t c)	a -> Dd b -> Sn c -> Sm	SQDMLSL Dd,Sn,Sm	Dd -> result	A64
int32x4_t vqrdmlsl_high_s16(int32x4_t a, int16x8_t b, int16x8_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.8H	SQDMLSL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
int64x2_t vqrdmlsl_high_s32(int64x2_t a, int32x4_t b, int32x4_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.4S	SQDMLSL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
int16x8_t vmull_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SMULL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
int32x4_t vmull_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SMULL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vmull_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SMULL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
uint16x8_t vmull_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UMULL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
uint32x4_t vmull_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UMULL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
uint64x2_t vmull_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UMULL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
poly16x8_t vmull_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	PMULL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
int16x8_t vmull_high_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SMULL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
int32x4_t vmull_high_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SMULL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
int64x2_t vmull_high_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SMULL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
uint16x8_t vmull_high_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UMULL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
uint32x4_t vmull_high_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UMULL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
uint64x2_t vmull_high_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UMULL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly16x8_t vmull_high_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	PMULL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
int32x4_t vqdmull_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SQDMULL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vqdmull_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SQDMULL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
int32_t vqdmullh_s16(int16_t a, int16_t b)	a -> Hn b -> Hm	SQDMULL Sd,Hn,Hm	Sd -> result	A64
int64_t vqdmulls_s32(int32_t a, int32_t b)	a -> Sn b -> Sm	SQDMULL Dd,Sn,Sm	Dd -> result	A64
int32x4_t vqdmull_high_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SQDMULL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
int64x2_t vqdmull_high_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SQDMULL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
int8x8_t vsub_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SUB Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vsubq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SUB Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vsub_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SUB Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vsubq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SUB Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vsub_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SUB Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vsubq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SUB Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int64x1_t vsub_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	SUB Dd,Dn,Dm	Dd -> result	v7/A32/A64
int64x2_t vsubq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	SUB Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
uint8x8_t vsub_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	SUB Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vsubq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	SUB Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vsub_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	SUB Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vsubq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	SUB Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vsub_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	SUB Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vsubq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	SUB Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vsub_u64(uint64x1_t a, uint64x1_t b)	a -> Dn b -> Dm	SUB Dd,Dn,Dm	Dd -> result	v7/A32/A64
uint64x2_t vsubq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	SUB Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
float32x2_t vsub_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FSUB Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vsubq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FSUB Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float64x1_t vsub_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FSUB Dd,Dn,Dm	Dd -> result	A64
float64x2_t vsubq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FSUB Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
int64_t vsubd_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	SUB Dd,Dn,Dm	Dd -> result	A64
uint64_t vsubd_u64(uint64_t a, uint64_t b)	a -> Dn b -> Dm	SUB Dd,Dn,Dm	Dd -> result	A64
int16x8_t vsubl_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SSUBL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
int32x4_t vsubl_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SSUBL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vsubl_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SSUBL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
uint16x8_t vsubl_u8(int8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	USUBL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
uint32x4_t vsubl_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	USUBL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
uint64x2_t vsubl_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	USUBL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
int16x8_t vsubl_high_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SSUBL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t vsubl_high_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SSUBL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
int64x2_t vsubl_high_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SSUBL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
uint16x8_t vsubl_high_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	USUBL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
uint32x4_t vsubl_high_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	USUBL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
uint64x2_t vsubl_high_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	USUBL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
int16x8_t vsubw_s8(int16x8_t a, int8x8_t b)	a -> Vn.8H b -> Vm.8B	SSUBW Vd.8H,Vn.8H,Vm.8B	Vd.8H -> result	v7/A32/A64
int32x4_t vsubw_s16(int32x4_t a, int16x4_t b)	a -> Vn.4S b -> Vm.4H	SSUBW Vd.4S,Vn.4S,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vsubw_s32(int64x2_t a, int32x2_t b)	a -> Vn.2D b -> Vm.2S	SSUBW Vd.2D,Vn.2D,Vm.2S	Vd.2D -> result	v7/A32/A64
uint16x8_t vsubw_u8(uint16x8_t a, uint8x8_t b)	a -> Vn.8H b -> Vm.8B	USUBW Vd.8H,Vn.8H,Vm.8B	Vd.8H -> result	v7/A32/A64
uint32x4_t vsubw_u16(uint32x4_t a, uint16x4_t b)	a -> Vn.4S b -> Vm.4H	USUBW Vd.4S,Vn.4S,Vm.4H	Vd.4S -> result	v7/A32/A64
uint64x2_t vsubw_u32(uint64x2_t a, uint32x2_t b)	a -> Vn.2D b -> Vm.2S	USUBW Vd.2D,Vn.2D,Vm.2S	Vd.2D -> result	v7/A32/A64
int16x8_t vsubw_high_s8(int16x8_t a, int8x16_t b)	a -> Vn.8H b -> Vm.16B	SSUBW2 Vd.8H,Vn.8H,Vm.16B	Vd.8H -> result	A64
int32x4_t vsubw_high_s16(int32x4_t a, int16x8_t b)	a -> Vn.4S b -> Vm.8H	SSUBW2 Vd.4S,Vn.4S,Vm.8H	Vd.4S -> result	A64
int64x2_t vsubw_high_s32(int64x2_t a, int32x4_t b)	a -> Vn.2D b -> Vm.4S	SSUBW2 Vd.2D,Vn.2D,Vm.4S	Vd.2D -> result	A64
uint16x8_t vsubw_high_u8(uint16x8_t a, uint8x16_t b)	a -> Vn.8H b -> Vm.16B	USUBW2 Vd.8H,Vn.8H,Vm.16B	Vd.8H -> result	A64
uint32x4_t vsubw_high_u16(uint32x4_t a, uint16x8_t b)	a -> Vn.4S b -> Vm.8H	USUBW2 Vd.4S,Vn.4S,Vm.8H	Vd.4S -> result	A64
uint64x2_t vsubw_high_u32(uint64x2_t a, uint32x4_t b)	a -> Vn.2D b -> Vm.4S	USUBW2 Vd.2D,Vn.2D,Vm.4S	Vd.2D -> result	A64
int8x8_t vhsb_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SHSUB Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vhsbq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SHSUB Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vhsb_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SHSUB Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vhsbq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SHSUB Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vhsb_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SHSUB Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vhsbq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SHSUB Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vhsb_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UHSUB Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vhsbq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UHSUB Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vhsb_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UHSUB Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vhsbq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UHSUB Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vhsb_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UHSUB Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vhsbq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UHSUB Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int8x8_t vqsub_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SQSUB Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vqsubq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SQSUB Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vqsub_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SQSUB Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vqsubq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SQSUB Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vqsub_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SQSUB Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vqsubq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SQSUB Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int64x1_t vqsub_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	SQSUB Dd,Dn,Dm	Dd -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64x2_t vqsubq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	SQSUB Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
uint8x8_t vqsub_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UQSUB Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vqsubq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UQSUB Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vqsub_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UQSUB Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vqsubq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UQSUB Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vqsub_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UQSUB Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vqsubq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UQSUB Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vqsub_u64(uint64x1_t a, uint64x1_t b)	a -> Dn b -> Dm	UQSUB Dd,Dn,Dm	Dd -> result	v7/A32/A64
uint64x2_t vqsubq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	UQSUB Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
int8_t vqsubb_s8(int8_t a, int8_t b)	a -> Bn b -> Bm	SQSUB Bd,Bn,Bm	Bd -> result	A64
int16_t vqsubh_s16(int16_t a, int16_t b)	a -> Hn b -> Hm	SQSUB Hd,Hn,Hm	Hd -> result	A64
int32_t vqsubs_s32(int32_t a, int32_t b)	a -> Sn b -> Sm	SQSUB Sd,Sn,Sm	Sd -> result	A64
int64_t vqsubd_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	SQSUB Dd,Dn,Dm	Dd -> result	A64
uint8_t vqsubb_u8(uint8_t a, uint8_t b)	a -> Bn b -> Bm	UQSUB Bd,Bn,Bm	Bd -> result	A64
uint16_t vqsubh_u16(uint16_t a, uint16_t b)	a -> Hn b -> Hm	UQSUB Hd,Hn,Hm	Hd -> result	A64
uint32_t vqsubs_u32(uint32_t a, uint32_t b)	a -> Sn b -> Sm	UQSUB Sd,Sn,Sm	Sd -> result	A64
uint64_t vqsubd_u64(uint64_t a, uint64_t b)	a -> Dn b -> Dm	UQSUB Dd,Dn,Dm	Dd -> result	A64
int8x8_t vsubhn_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SUBHN Vd.8B,Vn.8H,Vm.8H	Vd.8B -> result	v7/A32/A64
int16x4_t vsubhn_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SUBHN Vd.4H,Vn.4S,Vm.4S	Vd.4H -> result	v7/A32/A64
int32x2_t vsubhn_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	SUBHN Vd.2S,Vn.2D,Vm.2D	Vd.2S -> result	v7/A32/A64
uint8x8_t vsubhn_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	SUBHN Vd.8B,Vn.8H,Vm.8H	Vd.8B -> result	v7/A32/A64
uint16x4_t vsubhn_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	SUBHN Vd.4H,Vn.4S,Vm.4S	Vd.4H -> result	v7/A32/A64
uint32x2_t vsubhn_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	SUBHN Vd.2S,Vn.2D,Vm.2D	Vd.2S -> result	v7/A32/A64
int8x16_t vsubhn_high_s16(int8x8_t r, int16x8_t a, int16x8_t b)	r -> Vd.8B a -> Vn.8H b -> Vm.8H	SUBHN2 Vd.16B,Vn.8H,Vm.8H	Vd.16B -> result	A64
int16x8_t vsubhn_high_s32(int16x4_t r, int32x4_t a, int32x4_t b)	r -> Vd.4H a -> Vn.4S b -> Vm.4S	SUBHN2 Vd.8H,Vn.4S,Vm.4S	Vd.8H -> result	A64
int32x4_t vsubhn_high_s64(int32x2_t r, int64x2_t a, int64x2_t b)	r -> Vd.2S a -> Vn.2D b -> Vm.2D	SUBHN2 Vd.4S,Vn.2D,Vm.2D	Vd.4S -> result	A64
uint8x16_t vsubhn_high_u16(uint8x8_t r, uint16x8_t a, uint16x8_t b)	r -> Vd.8B a -> Vn.8H b -> Vm.8H	SUBHN2 Vd.16B,Vn.8H,Vm.8H	Vd.16B -> result	A64
uint16x8_t vsubhn_high_u32(uint16x4_t r, uint32x4_t a, uint32x4_t b)	r -> Vd.4H a -> Vn.4S b -> Vm.4S	SUBHN2 Vd.8H,Vn.4S,Vm.4S	Vd.8H -> result	A64
uint32x4_t vsubhn_high_u64(uint32x2_t r, uint64x2_t a, uint64x2_t b)	r -> Vd.2S a -> Vn.2D b -> Vm.2D	SUBHN2 Vd.4S,Vn.2D,Vm.2D	Vd.4S -> result	A64
int8x8_t vrsubhn_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	RSUBHN Vd.8B,Vn.8H,Vm.8H	Vd.8B -> result	v7/A32/A64
int16x4_t vrsubhn_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	RSUBHN Vd.4H,Vn.4S,Vm.4S	Vd.4H -> result	v7/A32/A64
int32x2_t vrsubhn_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	RSUBHN Vd.2S,Vn.2D,Vm.2D	Vd.2S -> result	v7/A32/A64
uint8x8_t vrsubhn_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	RSUBHN Vd.8B,Vn.8H,Vm.8H	Vd.8B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x4_t vrsubhn_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	RSUBHN Vd.4H,Vn.4S,Vm.4S	Vd.4H -> result	v7/A32/A64
uint32x2_t vrsubhn_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	RSUBHN Vd.2S,Vn.2D,Vm.2D	Vd.2S -> result	v7/A32/A64
int8x16_t vrsubhn_high_s16(int8x8_t r, int16x8_t a, int16x8_t b)	r -> Vd.8B a -> Vn.8H b -> Vm.8H	RSUBHN2 Vd.16B,Vn.8H,Vm.8H	Vd.16B -> result	A64
int16x8_t vrsubhn_high_s32(int16x4_t r, int32x4_t a, int32x4_t b)	r -> Vd.4H a -> Vn.4S b -> Vm.4S	RSUBHN2 Vd.8H,Vn.4S,Vm.4S	Vd.8H -> result	A64
int32x4_t vrsubhn_high_s64(int32x2_t r, int64x2_t a, int64x2_t b)	r -> Vd.2S a -> Vn.2D b -> Vm.2D	RSUBHN2 Vd.4S,Vn.2D,Vm.2D	Vd.4S -> result	A64
uint8x16_t vrsubhn_high_u16(uint8x8_t r, uint16x8_t a, uint16x8_t b)	r -> Vd.8B a -> Vn.8H b -> Vm.8H	RSUBHN2 Vd.16B,Vn.8H,Vm.8H	Vd.16B -> result	A64
uint16x8_t vrsubhn_high_u32(uint16x4_t r, uint32x4_t a, uint32x4_t b)	r -> Vd.4H a -> Vn.4S b -> Vm.4S	RSUBHN2 Vd.8H,Vn.4S,Vm.4S	Vd.8H -> result	A64
uint32x4_t vrsubhn_high_u64(uint32x2_t r, uint64x2_t a, uint64x2_t b)	r -> Vd.2S a -> Vn.2D b -> Vm.2D	RSUBHN2 Vd.4S,Vn.2D,Vm.2D	Vd.4S -> result	A64
uint8x8_t vceq_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	CMEQ Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vceqq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	CMEQ Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vceq_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	CMEQ Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vceqq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	CMEQ Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vceq_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	CMEQ Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vceqq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	CMEQ Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vceq_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	CMEQ Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vceqq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	CMEQ Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vceq_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	CMEQ Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vceqq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	CMEQ Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vceq_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	CMEQ Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vceqq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	CMEQ Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint32x2_t vceq_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FCMEQ Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vceqq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FCMEQ Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vceq_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	CMEQ Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vceqq_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	CMEQ Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint64x1_t vceq_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	CMEQ Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vceqq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	CMEQ Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint64x1_t vceq_u64(uint64x1_t a, uint64x1_t b)	a -> Dn b -> Dm	CMEQ Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vceqq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	CMEQ Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint64x1_t vceq_p64(poly64x1_t a, poly64x1_t b)	a -> Dn b -> Dm	CMEQ Dd,Dn,Dm	Dd -> result	A32/A64
uint64x2_t vceqq_p64(poly64x2_t a, poly64x2_t b)	a -> Vn.2D b -> Vm.2D	CMEQ Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A32/A64
uint64x1_t vceq_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FCMEQ Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vceqq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FCMEQ Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint64_t vceqd_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	CMEQ Dd,Dn,Dm	Dd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64_t vceqzd_u64(uint64_t a, uint64_t b)	a -> Dn b -> Dm	CMEQ Dd,Dn,Dm	Dd -> result	A64
uint32_t vceqz_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FCMEQ Sd,Sn,Sm	Sd -> result	A64
uint64_t vceqzd_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FCMEQ Dd,Dn,Dm	Dd -> result	A64
uint8x8_t vceqz_s8(int8x8_t a)	a -> Vn.8B	CMEQ Vd.8B,Vn.8B,#0	Vd.8B -> result	A64
uint8x16_t vceqzq_s8(int8x16_t a)	a -> Vn.16B	CMEQ Vd.16B,Vn.16B,#0	Vd.16B -> result	A64
uint16x4_t vceqz_s16(int16x4_t a)	a -> Vn.4H	CMEQ Vd.4H,Vn.4H,#0	Vd.4H -> result	A64
uint16x8_t vceqzq_s16(int16x8_t a)	a -> Vn.8H	CMEQ Vd.8H,Vn.8H,#0	Vd.8H -> result	A64
uint32x2_t vceqz_s32(int32x2_t a)	a -> Vn.2S	CMEQ Vd.2S,Vn.2S,#0	Vd.2S -> result	A64
uint32x4_t vceqzq_s32(int32x4_t a)	a -> Vn.4S	CMEQ Vd.4S,Vn.4S,#0	Vd.4S -> result	A64
uint8x8_t vceqz_u8(uint8x8_t a)	a -> Vn.8B	CMEQ Vd.8B,Vn.8B,#0	Vd.8B -> result	A64
uint8x16_t vceqzq_u8(uint8x16_t a)	a -> Vn.16B	CMEQ Vd.16B,Vn.16B,#0	Vd.16B -> result	A64
uint16x4_t vceqz_u16(uint16x4_t a)	a -> Vn.4H	CMEQ Vd.4H,Vn.4H,#0	Vd.4H -> result	A64
uint16x8_t vceqzq_u16(uint16x8_t a)	a -> Vn.8H	CMEQ Vd.8H,Vn.8H,#0	Vd.8H -> result	A64
uint32x2_t vceqz_u32(uint32x2_t a)	a -> Vn.2S	CMEQ Vd.2S,Vn.2S,#0	Vd.2S -> result	A64
uint32x4_t vceqzq_u32(uint32x4_t a)	a -> Vn.4S	CMEQ Vd.4S,Vn.4S,#0	Vd.4S -> result	A64
uint32x2_t vceqz_f32(float32x2_t a)	a -> Vn.2S	FCMEQ Vd.2S,Vn.2S,#0	Vd.2S -> result	A64
uint32x4_t vceqzq_f32(float32x4_t a)	a -> Vn.4S	FCMEQ Vd.4S,Vn.4S,#0	Vd.4S -> result	A64
uint8x8_t vceqz_p8(poly8x8_t a)	a -> Vn.8B	CMEQ Vd.8B,Vn.8B,#0	Vd.8B -> result	A64
uint8x16_t vceqzq_p8(poly8x16_t a)	a -> Vn.16B	CMEQ Vd.16B,Vn.16B,#0	Vd.16B -> result	A64
uint64x1_t vceqz_s64(int64x1_t a)	a -> Dn	CMEQ Dd,Dn,#0	Dd -> result	A64
uint64x2_t vceqzq_s64(int64x2_t a)	a -> Vn.2D	CMEQ Vd.2D,Vn.2D,#0	Vd.2D -> result	A64
uint64x1_t vceqz_u64(uint64x1_t a)	a -> Dn	CMEQ Dd,Dn,#0	Dd -> result	A64
uint64x2_t vceqzq_u64(uint64x2_t a)	a -> Vn.2D	CMEQ Vd.2D,Vn.2D,#0	Vd.2D -> result	A64
uint64x1_t vceqz_p64(poly64x1_t a)	a -> Dn	CMEQ Dd,Dn,#0	Dd -> result	A32/A64
uint64x2_t vceqzq_p64(poly64x2_t a)	a -> Vn.2D	CMEQ Vd.2D,Vn.2D,#0	Vd.2D -> result	A32/A64
uint64x1_t vceqz_f64(float64x1_t a)	a -> Dn	FCMEQ Dd,Dn,#0	Dd -> result	A64
uint64x2_t vceqzq_f64(float64x2_t a)	a -> Vn.2D	FCMEQ Vd.2D,Vn.2D,#0	Vd.2D -> result	A64
uint64_t vceqzd_s64(int64_t a)	a -> Dn	CMEQ Dd,Dn,#0	Dd -> result	A64
uint32_t vceqz_f32(float32_t a)	a -> Sn	FCMEQ Sd,Sn,#0	Sd -> result	A64
uint64_t vceqzd_f64(float64_t a)	a -> Dn	FCMEQ Dd,Dn,#0	Dd -> result	A64
uint8x8_t vcge_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	CMGE Vd.8B,Vm.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vcgeq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	CMGE Vd.16B,Vm.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vcge_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	CMGE Vd.4H,Vm.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vcgeq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	CMGE Vd.8H,Vm.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vcge_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	CMGE Vd.2S,Vm.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcgeq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	CMGE Vd.4S,Vm.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vcge_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	CMHS Vd.8B,Vm.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vcgeq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	CMHS Vd.16B,Vm.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vcge_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	CMHS Vd.4H,Vm.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vcgeq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	CMHS Vd.8H,Vm.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vcge_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	CMHS Vd.2S,Vm.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcgeq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	CMHS Vd.4S,Vm.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint32x2_t vcge_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FCMGE Vd.2S,Vm.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcgeq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FCMGE Vd.4S,Vm.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vcge_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	CMGE Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vcgeq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	CMGE Vd.2D,Vm.2D,Vn.2D	Vd.2D -> result	A64
uint64x1_t vcge_u64(uint64x1_t a, uint64x1_t b)	a -> Dn b -> Dm	CMHS Dd,Dn,Dm	Dd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64x2_t vcgeq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	CMHS Vd.2D,Vm.2D,Vn.2D	Vd.2D -> result	A64
uint64x1_t vcge_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FCMGE Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vcgeq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FCMGE Vd.2D,Vm.2D,Vn.2D	Vd.2D -> result	A64
uint64_t vcged_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	CMGE Dd,Dn,Dm	Dd -> result	A64
uint64_t vcged_u64(uint64_t a, uint64_t b)	a -> Dn b -> Dm	CMHS Dd,Dn,Dm	Dd -> result	A64
uint32_t vcges_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FCMGE Sd,Sn,Sm	Sd -> result	A64
uint64_t vcged_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FCMGE Dd,Dn,Dm	Dd -> result	A64
uint8x8_t vcgez_s8(int8x8_t a)	a -> Vn.8B	CMGE Vd.8B,Vn.8B,#0	Vd.8B -> result	A64
uint8x16_t vcgezq_s8(int8x16_t a)	a -> Vn.16B	CMGE Vd.16B,Vn.16B,#0	Vd.16B -> result	A64
uint16x4_t vcgez_s16(int16x4_t a)	a -> Vn.4H	CMGE Vd.4H,Vn.4H,#0	Vd.4H -> result	A64
uint16x8_t vcgezq_s16(int16x8_t a)	a -> Vn.8H	CMGE Vd.8H,Vn.8H,#0	Vd.8H -> result	A64
uint32x2_t vcgez_s32(int32x2_t a)	a -> Vn.2S	CMGE Vd.2S,Vn.2S,#0	Vd.2S -> result	A64
uint32x4_t vcgezq_s32(int32x4_t a)	a -> Vn.4S	CMGE Vd.4S,Vn.4S,#0	Vd.4S -> result	A64
uint64x1_t vcgez_s64(int64x1_t a)	a -> Dn	CMGE Dd,Dn,#0	Dd -> result	A64
uint64x2_t vcgezq_s64(int64x2_t a)	a -> Vn.2D	CMGE Vd.2D,Vn.2D,#0	Vd.2D -> result	A64
uint32x2_t vcgez_f32(float32x2_t a)	a -> Vn.2S	FCMGE Vd.2S,Vn.2S,#0	Vd.2S -> result	A64
uint32x4_t vcgezq_f32(float32x4_t a)	a -> Vn.4S	FCMGE Vd.4S,Vn.4S,#0	Vd.4S -> result	A64
uint64x1_t vcgez_f64(float64x1_t a)	a -> Dn	FCMGE Dd,Dn,#0	Dd -> result	A64
uint64x2_t vcgezq_f64(float64x2_t a)	a -> Vn.2D	FCMGE Vd.2D,Vn.2D,#0	Vd.2D -> result	A64
uint64_t vcgezd_s64(int64_t a)	a -> Dn	CMGE Dd,Dn,#0	Dd -> result	A64
uint32_t vcgez_s32(float32_t a)	a -> Sn	FCMGE Sd,Sn,#0	Sd -> result	A64
uint64_t vcgezd_f64(float64_t a)	a -> Dn	FCMGE Dd,Dn,#0	Dd -> result	A64
uint8x8_t vcle_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	CMGE Vd.8B,Vm.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vcleq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	CMGE Vd.16B,Vm.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vcle_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	CMGE Vd.4H,Vm.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vcleq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	CMGE Vd.8H,Vm.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vcle_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	CMGE Vd.2S,Vm.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcleq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	CMGE Vd.4S,Vm.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vcle_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	CMHS Vd.8B,Vm.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vcleq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	CMHS Vd.16B,Vm.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vcle_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	CMHS Vd.4H,Vm.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vcleq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	CMHS Vd.8H,Vm.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vcle_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	CMHS Vd.2S,Vm.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcleq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	CMHS Vd.4S,Vm.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint32x2_t vcle_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FCMGE Vd.2S,Vm.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcleq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FCMGE Vd.4S,Vm.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vcle_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	CMGE Dd,Dm,Dn	Dd -> result	A64
uint64x2_t vcleq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	CMGE Vd.2D,Vm.2D,Vn.2D	Vd.2D -> result	A64
uint64x1_t vcle_u64(uint64x1_t a, uint64x1_t b)	a -> Dn b -> Dm	CMHS Dd,Dm,Dn	Dd -> result	A64
uint64x2_t vcleq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	CMHS Vd.2D,Vm.2D,Vn.2D	Vd.2D -> result	A64
uint64x1_t vcle_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FCMGE Dd,Dm,Dn	Dd -> result	A64
uint64x2_t vcleq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FCMGE Vd.2D,Vm.2D,Vn.2D	Vd.2D -> result	A64
uint64_t vcled_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	CMGE Dd,Dm,Dn	Dd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64_t vclded_u64(uint64_t a, uint64_t b)	a -> Dn b -> Dm	CMHS Dd,Dm,Dn	Dd -> result	A64
uint32_t vcles_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FCMGE Sd,Sm,Sn	Sd -> result	A64
uint64_t vclded_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FCMGE Dd,Dm,Dn	Dd -> result	A64
uint8x8_t vclez_s8(int8x8_t a)	a -> Vn.8B	CMLE Vd.8B,Vn.8B,#0	Vd.8B -> result	A64
uint8x16_t vclezq_s8(int8x16_t a)	a -> Vn.16B	CMLE Vd.16B,Vn.16B,#0	Vd.16B -> result	A64
uint16x4_t vclez_s16(int16x4_t a)	a -> Vn.4H	CMLE Vd.4H,Vn.4H,#0	Vd.4H -> result	A64
uint16x8_t vclezq_s16(int16x8_t a)	a -> Vn.8H	CMLE Vd.8H,Vn.8H,#0	Vd.8H -> result	A64
uint32x2_t vclez_s32(int32x2_t a)	a -> Vn.2S	CMLE Vd.2S,Vn.2S,#0	Vd.2S -> result	A64
uint32x4_t vclezq_s32(int32x4_t a)	a -> Vn.4S	CMLE Vd.4S,Vn.4S,#0	Vd.4S -> result	A64
uint64x1_t vclez_s64(int64x1_t a)	a -> Dn	CMLE Dd,Dn,#0	Dd -> result	A64
uint64x2_t vclezq_s64(int64x2_t a)	a -> Vn.2D	CMLE Vd.2D,Vn.2D,#0	Vd.2D -> result	A64
uint32x2_t vclez_f32(float32x2_t a)	a -> Vn.2S	CMLE Vd.2S,Vn.2S,#0	Vd.2S -> result	A64
uint32x4_t vclezq_f32(float32x4_t a)	a -> Vn.4S	FCMLE Vd.4S,Vn.4S,#0	Vd.4S -> result	A64
uint64x1_t vclez_f64(float64x1_t a)	a -> Dn	FCMLE Dd,Dn,#0	Dd -> result	A64
uint64x2_t vclezq_f64(float64x2_t a)	a -> Vn.2D	FCMLE Vd.2D,Vn.2D,#0	Vd.2D -> result	A64
uint64_t vclezd_s64(int64_t a)	a -> Dn	CMLE Dd,Dn,#0	Dd -> result	A64
uint32_t vclezs_f32(float32_t a)	a -> Sn	FCMLE Sd,Sn,#0	Sd -> result	A64
uint64_t vclezd_f64(float64_t a)	a -> Dn	FCMLE Dd,Dn,#0	Dd -> result	A64
uint8x8_t vcgt_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	CMGT Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vcgtq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	CMGT Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vcgt_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	CMGT Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vcgtq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	CMGT Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vcgt_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	CMGT Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcgtq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	CMGT Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vcgt_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	CMHI Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vcgtq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	CMHI Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vcgt_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	CMHI Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vcgtq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	CMHI Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vcgt_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	CMHI Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcgtq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	CMHI Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint32x2_t vcgt_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FCMGT Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcgtq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FCMGT Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vcgt_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	CMGT Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vcgtq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	CMGT Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint64x1_t vcgt_u64(uint64x1_t a, uint64x1_t b)	a -> Dn b -> Dm	CMHI Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vcgtq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	CMHI Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint64x1_t vcgt_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FCMGT Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vcgtq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FCMGT Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint64_t vcgtd_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	CMGT Dd,Dn,Dm	Dd -> result	A64
uint64_t vcgtd_u64(uint64_t a, uint64_t b)	a -> Dn b -> Dm	CMHI Dd,Dn,Dm	Dd -> result	A64
uint32_t vcgts_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FCMGT Sd,Sn,Sm	Sd -> result	A64
uint64_t vcgtd_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FCMGT Dd,Dn,Dm	Dd -> result	A64
uint8x8_t vcgtz_s8(int8x8_t a)	a -> Vn.8B	CMGT Vd.8B,Vn.8B,#0	Vd.8B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t vcgztq_s8(int8x16_t a)	a -> Vn.16B	CMGT Vd.16B,Vn.16B,#0	Vd.16B -> result	A64
uint16x4_t vcgztz_s16(int16x4_t a)	a -> Vn.4H	CMGT Vd.4H,Vn.4H,#0	Vd.4H -> result	A64
uint16x8_t vcgztzq_s16(int16x8_t a)	a -> Vn.8H	CMGT Vd.8H,Vn.8H,#0	Vd.8H -> result	A64
uint32x2_t vcgztz_s32(int32x2_t a)	a -> Vn.2S	CMGT Vd.2S,Vn.2S,#0	Vd.2S -> result	A64
uint32x4_t vcgztzq_s32(int32x4_t a)	a -> Vn.4S	CMGT Vd.4S,Vn.4S,#0	Vd.4S -> result	A64
uint64x1_t vcgztz_s64(int64x1_t a)	a -> Dn	CMGT Dd,Dn,#0	Dd -> result	A64
uint64x2_t vcgztzq_s64(int64x2_t a)	a -> Vn.2D	CMGT Vd.2D,Vn.2D,#0	Vd.2D -> result	A64
uint32x2_t vcgztz_f32(float32x2_t a)	a -> Vn.2S	FCMGT Vd.2S,Vn.2S,#0	Vd.2S -> result	A64
uint32x4_t vcgztzq_f32(float32x4_t a)	a -> Vn.4S	FCMGT Vd.4S,Vn.4S,#0	Vd.4S -> result	A64
uint64x1_t vcgztz_f64(float64x1_t a)	a -> Dn	FCMGT Dd,Dn,#0	Dd -> result	A64
uint64x2_t vcgztzq_f64(float64x2_t a)	a -> Vn.2D	FCMGT Vd.2D,Vn.2D,#0	Vd.2D -> result	A64
uint64_t vcgztzd_s64(int64_t a)	a -> Dn	CMGT Dd,Dn,#0	Dd -> result	A64
uint32_t vcgztzs_f32(float32_t a)	a -> Sn	FCMGT Sd,Sn,#0	Sd -> result	A64
uint64_t vcgztzd_f64(float64_t a)	a -> Dn	FCMGT Dd,Dn,#0	Dd -> result	A64
uint8x8_t vclt_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	CMGT Vd.8B,Vm.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vcltq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	CMGT Vd.16B,Vm.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vclt_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	CMGT Vd.4H,Vm.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vcltq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	CMGT Vd.8H,Vm.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vclt_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	CMGT Vd.2S,Vm.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcltq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	CMGT Vd.4S,Vm.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vclt_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	CMHI Vd.8B,Vm.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vcltq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	CMHI Vd.16B,Vm.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vclt_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	CMHI Vd.4H,Vm.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vcltq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	CMHI Vd.8H,Vm.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vclt_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	CMHI Vd.2S,Vm.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcltq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	CMHI Vd.4S,Vm.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint32x2_t vclt_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FCMGT Vd.2S,Vm.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcltq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FCMGT Vd.4S,Vm.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vclt_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	CMGT Dd,Dm,Dn	Dd -> result	A64
uint64x2_t vcltq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	CMGT Vd.2D,Vm.2D,Vn.2D	Vd.2D -> result	A64
uint64x1_t vclt_u64(uint64x1_t a, uint64x1_t b)	a -> Dn b -> Dm	CMHI Dd,Dm,Dn	Dd -> result	A64
uint64x2_t vcltq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	CMHI Vd.2D,Vm.2D,Vn.2D	Vd.2D -> result	A64
uint64x1_t vclt_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FCMGT Dd,Dm,Dn	Dd -> result	A64
uint64x2_t vcltq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FCMGT Vd.2D,Vm.2D,Vn.2D	Vd.2D -> result	A64
uint64_t vcltd_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	CMGT Dd,Dm,Dn	Dd -> result	A64
uint64_t vcltd_u64(uint64_t a, uint64_t b)	a -> Dn b -> Dm	CMHI Dd,Dm,Dn	Dd -> result	A64
uint32_t vclts_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FCMGT Sd,Sm,Sn	Sd -> result	A64
uint64_t vcltd_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FCMGT Dd,Dm,Dn	Dd -> result	A64
uint8x8_t vcltz_s8(int8x8_t a)	a -> Vn.8B	CMLT Vd.8B,Vn.8B,#0	Vd.8B -> result	A64
uint8x16_t vcltzq_s8(int8x16_t a)	a -> Vn.16B	CMLT Vd.16B,Vn.16B,#0	Vd.16B -> result	A64
uint16x4_t vcltz_s16(int16x4_t a)	a -> Vn.4H	CMLT Vd.4H,Vn.4H,#0	Vd.4H -> result	A64
uint16x8_t vcltzq_s16(int16x8_t a)	a -> Vn.8H	CMLT Vd.8H,Vn.8H,#0	Vd.8H -> result	A64
uint32x2_t vcltz_s32(int32x2_t a)	a -> Vn.2S	CMLT Vd.2S,Vn.2S,#0	Vd.2S -> result	A64
uint32x4_t vcltzq_s32(int32x4_t a)	a -> Vn.4S	CMLT Vd.4S,Vn.4S,#0	Vd.4S -> result	A64
uint64x1_t vcltz_s64(int64x1_t a)	a -> Dn	CMLT Dd,Dn,#0	Dd -> result	A64
uint64x2_t vcltzq_s64(int64x2_t a)	a -> Vn.2D	CMLT Vd.2D,Vn.2D,#0	Vd.2D -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x2_t vcltz_f32(float32x2_t a)	a -> Vn.2S	FCMLT Vd.2S,Vn.2S,#0	Vd.2S -> result	A64
uint32x4_t vcltzq_f32(float32x4_t a)	a -> Vn.4S	FCMLT Vd.4S,Vn.4S,#0	Vd.4S -> result	A64
uint64x1_t vcltz_f64(float64x1_t a)	a -> Dn	FCMLT Dd,Dn,#0	Dd -> result	A64
uint64x2_t vcltzq_f64(float64x2_t a)	a -> Vn.2D	FCMLT Vd.2D,Vn.2D,#0	Vd.2D -> result	A64
uint64_t vcltzd_s64(int64_t a)	a -> Dn	CMLT Dd,Dn,#0	Dd -> result	A64
uint32_t vcltzs_f32(float32_t a)	a -> Sn	FCMLT Sd,Sn,#0	Sd -> result	A64
uint64_t vcltzd_f64(float64_t a)	a -> Dn	FCMLT Dd,Dn,#0	Dd -> result	A64
uint32x2_t vcage_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FACGE Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcageq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FACGE Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vcage_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FACGE Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vcageq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FACGE Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint32_t vcages_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FACGE Sd,Sn,Sm	Sd -> result	A64
uint64_t vcaged_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FACGE Dd,Dn,Dm	Dd -> result	A64
uint32x2_t vcale_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FACGE Vd.2S,Vm.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcaleq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FACGE Vd.4S,Vm.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vcale_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FACGE Dd,Dm,Dn	Dd -> result	A64
uint64x2_t vcaleq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FACGE Vd.2D,Vm.2D,Vn.2D	Vd.2D -> result	A64
uint32_t vcales_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FACGE Sd,Sm,Sn	Sd -> result	A64
uint64_t vcaled_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FACGE Dd,Dm,Dn	Dd -> result	A64
uint32x2_t vcagt_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FACGT Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcagtq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FACGT Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vcagt_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FACGT Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vcagtq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FACGT Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint32_t vcagts_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FACGT Sd,Sn,Sm	Sd -> result	A64
uint64_t vcagtd_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FACGT Dd,Dn,Dm	Dd -> result	A64
uint32x2_t vcalt_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FACGT Vd.2S,Vm.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcaltq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FACGT Vd.4S,Vm.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vcalt_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FACGT Dd,Dm,Dn	Dd -> result	A64
uint64x2_t vcaltq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FACGT Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint32_t vcalts_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FACGT Sd,Sm,Sn	Sd -> result	A64
uint64_t vcaltd_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FACGT Dd,Dm,Dn	Dd -> result	A64
uint8x8_t vtst_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	CMTST Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vtstq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	CMTST Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vtst_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	CMTST Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vtstq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	CMTST Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vtst_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	CMTST Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vtstq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	CMTST Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vtst_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	CMTST Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vtstq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	CMTST Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vtst_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	CMTST Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t vtstq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	CMTST Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vtst_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	CMTST Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vtstq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	CMTST Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vtst_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	CMTST Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vtstq_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	CMTST Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint64x1_t vtst_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	CMTST Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vtstq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	CMTST Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint64x1_t vtst_u64(uint64x1_t a, uint64x1_t b)	a -> Dn b -> Dm	CMTST Dd,Dn,Dm	Dd -> result	A64
uint64x2_t vtstq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	CMTST Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint64x1_t vtst_p64(poly64x1_t a, poly64x1_t b)	a -> Dn b -> Dm	CMTST Dd,Dn,Dm	Dd -> result	A32/A64
uint64x2_t vtstq_p64(poly64x2_t a, poly64x2_t b)	a -> Vn.2D b -> Vm.2D	CMTST Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A32/A64
uint64_t vtstd_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	CMTST Dd,Dn,Dm	Dd -> result	A64
uint64_t vtstd_u64(uint64_t a, uint64_t b)	a -> Dn b -> Dm	CMTST Dd,Dn,Dm	Dd -> result	A64
int8x8_t vabd_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SABD Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vabdq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SABD Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vabd_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SABD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vabdq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SABD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vabd_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SABD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vabdq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SABD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vabd_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UABD Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vabdq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UABD Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vabd_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UABD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vabdq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UABD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vabd_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UABD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vabdq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UABD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float32x2_t vabd_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FABD Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vabdq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FABD Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float64x1_t vabd_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FABD Dd,Dn,Dm	Dd -> result	A64
float64x2_t vabdq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FABD Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32_t vabds_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FABD Sd,Sn,Sm	Sd -> result	A64
float64_t vabdd_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FABD Dd,Dn,Dm	Dd -> result	A64
int16x8_t vabdl_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SABDL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
int32x4_t vabdl_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SABDL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vabdl_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SABDL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
uint16x8_t vabdl_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UABDL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
uint32x4_t vabdl_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UABDL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64x2_t vabdl_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UABDL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
int16x8_t vabdl_high_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SABDL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
int32x4_t vabdl_high_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SABDL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
int64x2_t vabdl_high_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SABDL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
uint16x8_t vabdl_high_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UABDL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
uint32x4_t vabdl_high_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UABDL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
uint64x2_t vabdl_high_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UABDL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
int8x8_t vaba_s8(int8x8_t a, int8x8_t b, int8x8_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	SABA Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vabaq_s8(int8x16_t a, int8x16_t b, int8x16_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	SABA Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vaba_s16(int16x4_t a, int16x4_t b, int16x4_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.4H	SABA Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vabaq_s16(int16x8_t a, int16x8_t b, int16x8_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.8H	SABA Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vaba_s32(int32x2_t a, int32x2_t b, int32x2_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.2S	SABA Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vabaq_s32(int32x4_t a, int32x4_t b, int32x4_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.4S	SABA Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vaba_u8(uint8x8_t a, uint8x8_t b, uint8x8_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	UABA Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vabaq_u8(uint8x16_t a, uint8x16_t b, uint8x16_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	UABA Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vaba_u16(uint16x4_t a, uint16x4_t b, uint16x4_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.4H	UABA Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vabaq_u16(uint16x8_t a, uint16x8_t b, uint16x8_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.8H	UABA Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vaba_u32(uint32x2_t a, uint32x2_t b, uint32x2_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.2S	UABA Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vabaq_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.4S	UABA Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int16x8_t vabal_s8(int16x8_t a, int8x8_t b, int8x8_t c)	a -> Vd.8H b -> Vn.8B c -> Vm.8B	SABAL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
int32x4_t vabal_s16(int32x4_t a, int16x4_t b, int16x4_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.4H	SABAL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
int64x2_t vabal_s32(int64x2_t a, int32x2_t b, int32x2_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.2S	SABAL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
uint16x8_t vabal_u8(uint16x8_t a, uint8x8_t b, uint8x8_t c)	a -> Vd.8H b -> Vn.8B c -> Vm.8B	UABAL Vd.8H,Vn.8B,Vm.8B	Vd.8H -> result	v7/A32/A64
uint32x4_t vabal_u16(uint32x4_t a, uint16x4_t b, uint16x4_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.4H	UABAL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	v7/A32/A64
uint64x2_t vabal_u32(uint64x2_t a, uint32x2_t b, uint32x2_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.2S	UABAL Vd.2D,Vn.2S,Vm.2S	Vd.2D -> result	v7/A32/A64
int16x8_t vabal_high_s8(int16x8_t a, int8x16_t b, int8x16_t c)	a -> Vd.8H b -> Vn.16B c -> Vm.16B	SABAL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t vabal_high_s16(int32x4_t a, int16x8_t b, int16x8_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.8H	SABAL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
int64x2_t vabal_high_s32(int64x2_t a, int32x4_t b, int32x4_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.4S	SABAL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
uint16x8_t vabal_high_u8(uint16x8_t a, uint8x16_t b, uint8x16_t c)	a -> Vd.8H b -> Vn.16B c -> Vm.16B	UABAL2 Vd.8H,Vn.16B,Vm.16B	Vd.8H -> result	A64
uint32x4_t vabal_high_u16(uint32x4_t a, uint16x8_t b, uint16x8_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.8H	UABAL2 Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A64
uint64x2_t vabal_high_u32(uint64x2_t a, uint32x4_t b, uint32x4_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.4S	UABAL2 Vd.2D,Vn.4S,Vm.4S	Vd.2D -> result	A64
int8x8_t vmax_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SMAX Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vmaxq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SMAX Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vmax_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SMAX Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vmaxq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SMAX Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vmax_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SMAX Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vmaxq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SMAX Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vmax_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UMAX Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vmaxq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UMAX Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vmax_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UMAX Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vmaxq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UMAX Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vmax_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UMAX Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vmaxq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UMAX Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float32x2_t vmax_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FMAX Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vmaxq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FMAX Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float64x1_t vmax_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FMAX Dd,Dn,Dm	Dd -> result	A64
float64x2_t vmaxq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FMAX Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
int8x8_t vmin_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SMIN Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vminq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SMIN Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vmin_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SMIN Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vminq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SMIN Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vmin_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SMIN Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vminq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SMIN Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vmin_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UMIN Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vminq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UMIN Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vmin_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UMIN Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vminq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UMIN Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vmin_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UMIN Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vminq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UMIN Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float32x2_t vmin_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FMIN Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t vminq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FMIN Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float64x1_t vmin_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FMIN Dd,Dn,Dm	Dd -> result	A64
float64x2_t vminq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FMIN Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t vmaxnm_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FMAXNM Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A32/A64
float32x4_t vmaxnmq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FMAXNM Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A32/A64
float64x1_t vmaxnm_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FMAXNM Dd,Dn,Dm	Dd -> result	A64
float64x2_t vmaxnmq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FMAXNM Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t vminnm_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FMINNM Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A32/A64
float32x4_t vminnmq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FMINNM Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A32/A64
float64x1_t vminnm_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FMINNM Dd,Dn,Dm	Dd -> result	A64
float64x2_t vminnmq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FMINNM Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
int8x8_t vshl_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SSHL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vshlq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SSHL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vshl_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SSHL Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vshlq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SSHL Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vshl_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SSHL Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vshlq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SSHL Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int64x1_t vshl_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	SSHL Dd,Dn,Dm	Dd -> result	v7/A32/A64
int64x2_t vshlq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	SSHL Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
uint8x8_t vshl_u8(uint8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	USHL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vshlq_u8(uint8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	USHL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vshl_u16(uint16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	USHL Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vshlq_u16(uint16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	USHL Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vshl_u32(uint32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	USHL Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vshlq_u32(uint32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	USHL Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vshl_u64(uint64x1_t a, int64x1_t b)	a -> Dn b -> Dm	USHL Dd,Dn,Dm	Dd -> result	v7/A32/A64
uint64x2_t vshlq_u64(uint64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	USHL Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
int64_t vshld_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	SSHL Dd,Dn,Dm	Dd -> result	A64
uint64_t vshld_u64(uint64_t a, int64_t b)	a -> Dn b -> Dm	USHL Dd,Dn,Dm	Dd -> result	A64
int8x8_t vqshl_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SQSHL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vqshlq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SQSHL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vqshl_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SQSHL Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vqshlq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SQSHL Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vqshl_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SQSHL Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vqshlq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SQSHL Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int64x1_t vqshl_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	SQSHL Dd,Dn,Dm	Dd -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64x2_t vqshlq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	SQSHL Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
uint8x8_t vqshl_u8(uint8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	UQSHL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vqshlq_u8(uint8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	UQSHL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vqshl_u16(uint16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	UQSHL Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vqshlq_u16(uint16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	UQSHL Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vqshl_u32(uint32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	UQSHL Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vqshlq_u32(uint32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	UQSHL Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vqshl_u64(uint64x1_t a, int64x1_t b)	a -> Dn b -> Dm	UQSHL Dd,Dn,Dm	Dd -> result	v7/A32/A64
uint64x2_t vqshlq_u64(uint64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	UQSHL Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
int8_t vqshlb_s8(int8_t a, int8_t b)	a -> Bn b -> Bm	SQSHL Bd,Bn,Bm	Bd -> result	A64
int16_t vqshlh_s16(int16_t a, int16_t b)	a -> Hn b -> Hm	SQSHL Hd,Hn,Hm	Hd -> result	A64
int32_t vqshls_s32(int32_t a, int32_t b)	a -> Sn b -> Sm	SQSHL Sd,Sn,Sm	Sd -> result	A64
int64_t vqshld_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	SQSHL Dd,Dn,Dm	Dd -> result	A64
uint8_t vqshlb_u8(uint8_t a, int8_t b)	a -> Bn b -> Bm	UQSHL Bd,Bn,Bm	Bd -> result	A64
uint16_t vqshlh_u16(uint16_t a, int16_t b)	a -> Hn b -> Hm	UQSHL Hd,Hn,Hm	Hd -> result	A64
uint32_t vqshls_u32(uint32_t a, int32_t b)	a -> Sn b -> Sm	UQSHL Sd,Sn,Sm	Sd -> result	A64
uint64_t vqshld_u64(uint64_t a, int64_t b)	a -> Dn b -> Dm	UQSHL Dd,Dn,Dm	Dd -> result	A64
int8x8_t vrshl_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SRSHL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vrshlq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SRSHL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vrshl_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SRSHL Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vrshlq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SRSHL Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vrshl_u32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SRSHL Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vrshlq_u32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SRSHL Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int64x1_t vrshl_u64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	SRSHL Dd,Dn,Dm	Dd -> result	v7/A32/A64
int64x2_t vrshlq_u64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	SRSHL Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
uint8x8_t vrshl_u8(uint8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	URSHL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vrshlq_u8(uint8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	URSHL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vrshl_u16(uint16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	URSHL Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vrshlq_u16(uint16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	URSHL Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vrshl_u32(uint32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	URSHL Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vrshlq_u32(uint32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	URSHL Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vrshl_u64(uint64x1_t a, int64x1_t b)	a -> Dn b -> Dm	URSHL Dd,Dn,Dm	Dd -> result	v7/A32/A64
uint64x2_t vrshlq_u64(uint64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	URSHL Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
int64_t vrshld_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	SRSHL Dd,Dn,Dm	Dd -> result	A64
uint64_t vrshld_u64(uint64_t a, int64_t b)	a -> Dn b -> Dm	URSHL Dd,Dn,Dm	Dd -> result	A64
int8x8_t vqrshl_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SQRSHL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t vqrrshlq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SQRSHL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vqrrshl_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SQRSHL Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vqrrshlq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SQRSHL Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vqrrshl_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SQRSHL Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vqrrshlq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SQRSHL Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
int64x1_t vqrrshl_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	SQRSHL Dd,Dn,Dm	Dd -> result	v7/A32/A64
int64x2_t vqrrshlq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	SQRSHL Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
uint8x8_t vqrrshl_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UQRSHL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vqrrshlq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UQRSHL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vqrrshl_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UQRSHL Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vqrrshlq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UQRSHL Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vqrrshl_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UQRSHL Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vqrrshlq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UQRSHL Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
uint64x1_t vqrrshl_u64(uint64x1_t a, uint64x1_t b)	a -> Dn b -> Dm	UQRSHL Dd,Dn,Dm	Dd -> result	v7/A32/A64
uint64x2_t vqrrshlq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	UQRSHL Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	v7/A32/A64
int8_t vqrrshlb_s8(int8_t a, int8_t b)	a -> Bn b -> Bm	SQRSHL Bd,Bn,Bm	Bd -> result	A64
int16_t vqrrshlh_s16(int16_t a, int16_t b)	a -> Hn b -> Hm	SQRSHL Hd,Hn,Hm	Hd -> result	A64
int32_t vqrrshls_s32(int32_t a, int32_t b)	a -> Sn b -> Sm	SQRSHL Sd,Sn,Sm	Sd -> result	A64
int64_t vqrrshld_s64(int64_t a, int64_t b)	a -> Dn b -> Dm	SQRSHL Dd,Dn,Dm	Dd -> result	A64
uint8_t vqrrshlb_u8(uint8_t a, uint8_t b)	a -> Bn b -> Bm	UQRSHL Bd,Bn,Bm	Bd -> result	A64
uint16_t vqrrshlh_u16(uint16_t a, uint16_t b)	a -> Hn b -> Hm	UQRSHL Hd,Hn,Hm	Hd -> result	A64
uint32_t vqrrshls_u32(uint32_t a, uint32_t b)	a -> Sn b -> Sm	UQRSHL Sd,Sn,Sm	Sd -> result	A64
uint64_t vqrrshld_u64(uint64_t a, uint64_t b)	a -> Dn b -> Dm	UQRSHL Dd,Dn,Dm	Dd -> result	A64
int8x8_t vshrq_n_s8(int8x8_t a, const int n)	a -> Vn.8B 1 <= n <= 8	SSHR Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
int8x16_t vshrq_n_s8(int8x16_t a, const int n)	a -> Vn.16B 1 <= n <= 8	SSHR Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
int16x4_t vshrq_n_s16(int16x4_t a, const int n)	a -> Vn.4H 1 <= n <= 16	SSHR Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
int16x8_t vshrq_n_s16(int16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 16	SSHR Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
int32x2_t vshrq_n_s32(int32x2_t a, const int n)	a -> Vn.2S 1 <= n <= 32	SSHR Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
int32x4_t vshrq_n_s32(int32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 32	SSHR Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
int64x1_t vshrq_n_s64(int64x1_t a, const int n)	a -> Dn 1 <= n <= 64	SSHR Dd,Dn,#n	Dd -> result	v7/A32/A64
int64x2_t vshrq_n_s64(int64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 64	SSHR Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
uint8x8_t vshrq_n_u8(uint8x8_t a, const int n)	a -> Vn.8B 1 <= n <= 8	USHR Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
uint8x16_t vshrq_n_u8(uint8x16_t a, const int n)	a -> Vn.16B 1 <= n <= 8	USHR Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
uint16x4_t vshrq_n_u16(uint16x4_t a, const int n)	a -> Vn.4H 1 <= n <= 16	USHR Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
uint16x8_t vshrq_n_u16(uint16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 16	USHR Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
uint32x2_t vshrq_n_u32(uint32x2_t a, const int n)	a -> Vn.2S 1 <= n <= 32	USHR Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t vshrq_n_u32(uint32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 32	USHR Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
uint64x1_t vshr_n_u64(uint64x1_t a, const int n)	a -> Dn 1 <= n <= 64	USHR Dd,Dn,#n	Dd -> result	v7/A32/A64
uint64x2_t vshrq_n_u64(uint64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 64	USHR Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
int64_t vshrd_n_s64(int64_t a, const int n)	a -> Dn 1 <= n <= 64	SSHR Dd,Dn,#n	Dd -> result	A64
uint64_t vshrd_n_u64(uint64_t a, const int n)	a -> Dn 1 <= n <= 64	USHR Dd,Dn,#n	Dd -> result	A64
int8x8_t vshl_n_s8(int8x8_t a, const int n)	a -> Vn.8B 0 <= n <= 7	SHL Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
int8x16_t vshlq_n_s8(int8x16_t a, const int n)	a -> Vn.16B 0 <= n <= 7	SHL Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
int16x4_t vshl_n_s16(int16x4_t a, const int n)	a -> Vn.4H 0 <= n <= 15	SHL Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
int16x8_t vshlq_n_s16(int16x8_t a, const int n)	a -> Vn.8H 0 <= n <= 15	SHL Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
int32x2_t vshl_n_s32(int32x2_t a, const int n)	a -> Vn.2S 0 <= n <= 31	SHL Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
int32x4_t vshlq_n_s32(int32x4_t a, const int n)	a -> Vn.4S 0 <= n <= 31	SHL Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
int64x1_t vshl_n_s64(int64x1_t a, const int n)	a -> Dn 0 <= n <= 63	SHL Dd,Dn,#n	Dd -> result	v7/A32/A64
int64x2_t vshlq_n_s64(int64x2_t a, const int n)	a -> Vn.2D 0 <= n <= 63	SHL Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
uint8x8_t vshl_n_u8(uint8x8_t a, const int n)	a -> Vn.8B 0 <= n <= 7	SHL Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
uint8x16_t vshlq_n_u8(uint8x16_t a, const int n)	a -> Vn.16B 0 <= n <= 7	SHL Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
uint16x4_t vshl_n_u16(uint16x4_t a, const int n)	a -> Vn.4H 0 <= n <= 15	SHL Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
uint16x8_t vshlq_n_u16(uint16x8_t a, const int n)	a -> Vn.8H 0 <= n <= 15	SHL Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
uint32x2_t vshl_n_u32(uint32x2_t a, const int n)	a -> Vn.2S 0 <= n <= 31	SHL Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
uint32x4_t vshlq_n_u32(uint32x4_t a, const int n)	a -> Vn.4S 0 <= n <= 31	SHL Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
uint64x1_t vshl_n_u64(uint64x1_t a, const int n)	a -> Dn 0 <= n <= 63	SHL Dd,Dn,#n	Dd -> result	v7/A32/A64
uint64x2_t vshlq_n_u64(uint64x2_t a, const int n)	a -> Vn.2D 0 <= n <= 63	SHL Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
int64_t vshld_n_s64(int64_t a, const int n)	a -> Dn 0 <= n <= 63	SHL Dd,Dn,#n	Dd -> result	A64
uint64_t vshld_n_u64(uint64_t a, const int n)	a -> Dn 0 <= n <= 63	SHL Dd,Dn,#n	Dd -> result	A64
int8x8_t vrshl_n_s8(int8x8_t a, const int n)	a -> Vn.8B 1 <= n <= 8	SRRSHR Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
int8x16_t vrshrq_n_s8(int8x16_t a, const int n)	a -> Vn.16B 1 <= n <= 8	SRRSHR Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
int16x4_t vrshl_n_s16(int16x4_t a, const int n)	a -> Vn.4H 1 <= n <= 16	SRRSHR Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
int16x8_t vrshrq_n_s16(int16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 16	SRRSHR Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
int32x2_t vrshl_n_s32(int32x2_t a, const int n)	a -> Vn.2S 1 <= n <= 32	SRRSHR Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
int32x4_t vrshrq_n_s32(int32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 32	SRRSHR Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
int64x1_t vrshl_n_s64(int64x1_t a, const int n)	a -> Dn 1 <= n <= 64	SRRSHR Dd,Dn,#n	Dd -> result	v7/A32/A64
int64x2_t vrshrq_n_s64(int64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 64	SRRSHR Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
uint8x8_t vrshl_n_u8(uint8x8_t a, const int n)	a -> Vn.8B 1 <= n <= 8	URSHR Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
uint8x16_t vrshrq_n_u8(uint8x16_t a, const int n)	a -> Vn.16B 1 <= n <= 8	URSHR Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
uint16x4_t vrshl_n_u16(uint16x4_t a, const int n)	a -> Vn.4H 1 <= n <= 16	URSHR Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
uint16x8_t vrshrq_n_u16(uint16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 16	URSHR Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
uint32x2_t vrshl_n_u32(uint32x2_t a, const int n)	a -> Vn.2S 1 <= n <= 32	URSHR Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t vrshrq_n_u32(uint32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 32	URSHR Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
uint64x1_t vrshr_n_u64(uint64x1_t a, const int n)	a -> Dn 1 <= n <= 64	URSHR Dd,Dn,#n	Dd -> result	v7/A32/A64
uint64x2_t vrshrq_n_u64(uint64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 64	URSHR Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
int64_t vrshrd_n_s64(int64_t a, const int n)	a -> Dn 1 <= n <= 64	SRSR Dd,Dn,#n	Dd -> result	A64
uint64_t vrshrd_n_u64(uint64_t a, const int n)	a -> Dn 1 <= n <= 64	URSHR Dd,Dn,#n	Dd -> result	A64
int8x8_t vsra_n_s8(int8x8_t a, int8x8_t b, const int n)	a -> Vd.8B b -> Vn.8B 1 <= n <= 8	SSRA Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
int8x16_t vsraq_n_s8(int8x16_t a, int8x16_t b, const int n)	a -> Vd.16B b -> Vn.16B 1 <= n <= 8	SSRA Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
int16x4_t vsra_n_s16(int16x4_t a, int16x4_t b, const int n)	a -> Vd.4H b -> Vn.4H 1 <= n <= 16	SSRA Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
int16x8_t vsraq_n_s16(int16x8_t a, int16x8_t b, const int n)	a -> Vd.8H b -> Vn.8H 1 <= n <= 16	SSRA Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
int32x2_t vsra_n_s32(int32x2_t a, int32x2_t b, const int n)	a -> Vd.2S b -> Vn.2S 1 <= n <= 32	SSRA Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
int32x4_t vsraq_n_s32(int32x4_t a, int32x4_t b, const int n)	a -> Vd.4S b -> Vn.4S 1 <= n <= 32	SSRA Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
int64x1_t vsra_n_s64(int64x1_t a, int64x1_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	SSRA Dd,Dn,#n	Dd -> result	v7/A32/A64
int64x2_t vsraq_n_s64(int64x2_t a, int64x2_t b, const int n)	a -> Vd.2D b -> Vn.2D 1 <= n <= 64	SSRA Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
uint8x8_t vsra_n_u8(uint8x8_t a, uint8x8_t b, const int n)	a -> Vd.8B b -> Vn.8B 1 <= n <= 8	USRA Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
uint8x16_t vsraq_n_u8(uint8x16_t a, uint8x16_t b, const int n)	a -> Vd.16B b -> Vn.16B 1 <= n <= 8	USRA Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
uint16x4_t vsra_n_u16(uint16x4_t a, uint16x4_t b, const int n)	a -> Vd.4H b -> Vn.4H 1 <= n <= 16	USRA Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
uint16x8_t vsraq_n_u16(uint16x8_t a, uint16x8_t b, const int n)	a -> Vd.8H b -> Vn.8H 1 <= n <= 16	USRA Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
uint32x2_t vsra_n_u32(uint32x2_t a, uint32x2_t b, const int n)	a -> Vd.2S b -> Vn.2S 1 <= n <= 32	USRA Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
uint32x4_t vsraq_n_u32(uint32x4_t a, uint32x4_t b, const int n)	a -> Vd.4S b -> Vn.4S 1 <= n <= 32	USRA Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
uint64x1_t vsra_n_u64(uint64x1_t a, uint64x1_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	USRA Dd,Dn,#n	Dd -> result	v7/A32/A64
uint64x2_t vsraq_n_u64(uint64x2_t a, uint64x2_t b, const int n)	a -> Vd.2D b -> Vn.2D 1 <= n <= 64	USRA Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
int64_t vsrad_n_s64(int64_t a, int64_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	SSRA Dd,Dn,#n	Dd -> result	A64
uint64_t vsrad_n_u64(uint64_t a, uint64_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	USRA Dd,Dn,#n	Dd -> result	A64
int8x8_t vrsra_n_s8(int8x8_t a, int8x8_t b, const int n)	a -> Vd.8B b -> Vn.8B 1 <= n <= 8	SRSRA Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
int8x16_t vrsraq_n_s8(int8x16_t a, int8x16_t b, const int n)	a -> Vd.16B b -> Vn.16B 1 <= n <= 8	SRSRA Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
int16x4_t vrsra_n_s16(int16x4_t a, int16x4_t b, const int n)	a -> Vd.4H b -> Vn.4H 1 <= n <= 16	SRSRA Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t vrsraq_n_s16(int16x8_t a, int16x8_t b, const int n)	a -> Vd.8H b -> Vn.8H 1 <= n <= 16	SRSRA Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
int32x2_t vrsra_n_s32(int32x2_t a, int32x2_t b, const int n)	a -> Vd.2S b -> Vn.2S 1 <= n <= 32	SRSRA Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
int32x4_t vrsraq_n_s32(int32x4_t a, int32x4_t b, const int n)	a -> Vd.4S b -> Vn.4S 1 <= n <= 32	SRSRA Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
int64x1_t vrsra_n_s64(int64x1_t a, int64x1_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	SRSRA Dd,Dn,#n	Dd -> result	v7/A32/A64
int64x2_t vrsraq_n_s64(int64x2_t a, int64x2_t b, const int n)	a -> Vd.2D b -> Vn.2D 1 <= n <= 64	SRSRA Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
uint8x8_t vrsra_n_u8(uint8x8_t a, uint8x8_t b, const int n)	a -> Vd.8B b -> Vn.8B 1 <= n <= 8	URSRA Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
uint8x16_t vrsraq_n_u8(uint8x16_t a, uint8x16_t b, const int n)	a -> Vd.16B b -> Vn.16B 1 <= n <= 8	URSRA Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
uint16x4_t vrsra_n_u16(uint16x4_t a, uint16x4_t b, const int n)	a -> Vd.4H b -> Vn.4H 1 <= n <= 16	URSRA Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
uint16x8_t vrsraq_n_u16(uint16x8_t a, uint16x8_t b, const int n)	a -> Vd.8H b -> Vn.8H 1 <= n <= 16	URSRA Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
uint32x2_t vrsra_n_u32(uint32x2_t a, uint32x2_t b, const int n)	a -> Vd.2S b -> Vn.2S 1 <= n <= 32	URSRA Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
uint32x4_t vrsraq_n_u32(uint32x4_t a, uint32x4_t b, const int n)	a -> Vd.4S b -> Vn.4S 1 <= n <= 32	URSRA Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
uint64x1_t vrsra_n_u64(uint64x1_t a, uint64x1_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	URSRA Dd,Dn,#n	Dd -> result	v7/A32/A64
uint64x2_t vrsraq_n_u64(uint64x2_t a, uint64x2_t b, const int n)	a -> Vd.2D b -> Vn.2D 1 <= n <= 64	URSRA Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
int64_t vrsrad_n_s64(int64_t a, int64_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	SRSRA Dd,Dn,#n	Dd -> result	A64
uint64_t vrsrad_n_u64(uint64_t a, uint64_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	URSRA Dd,Dn,#n	Dd -> result	A64
int8x8_t vqshl_n_s8(int8x8_t a, const int n)	a -> Vn.8B 0 <= n <= 7	SQSHL Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
int8x16_t vqshlq_n_s8(int8x16_t a, const int n)	a -> Vn.16B 0 <= n <= 7	SQSHL Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
int16x4_t vqshl_n_s16(int16x4_t a, const int n)	a -> Vn.4H 0 <= n <= 15	SQSHL Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
int16x8_t vqshlq_n_s16(int16x8_t a, const int n)	a -> Vn.8H 0 <= n <= 15	SQSHL Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
int32x2_t vqshl_n_s32(int32x2_t a, const int n)	a -> Vn.2S 0 <= n <= 31	SQSHL Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
int32x4_t vqshlq_n_s32(int32x4_t a, const int n)	a -> Vn.4S 0 <= n <= 31	SQSHL Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
int64x1_t vqshl_n_s64(int64x1_t a, const int n)	a -> Dn 0 <= n <= 63	SQSHL Dd,Dn,#n	Dd -> result	v7/A32/A64
int64x2_t vqshlq_n_s64(int64x2_t a, const int n)	a -> Vn.2D 0 <= n <= 63	SQSHL Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
uint8x8_t vqshl_n_u8(uint8x8_t a, const int n)	a -> Vn.8B 0 <= n <= 7	UQSHL Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
uint8x16_t vqshlq_n_u8(uint8x16_t a, const int n)	a -> Vn.16B 0 <= n <= 7	UQSHL Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
uint16x4_t vqshl_n_u16(uint16x4_t a, const int n)	a -> Vn.4H 0 <= n <= 15	UQSHL Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
uint16x8_t vqshlq_n_u16(uint16x8_t a, const int n)	a -> Vn.8H 0 <= n <= 15	UQSHL Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
uint32x2_t vqshl_n_u32(uint32x2_t a, const int n)	a -> Vn.2S 0 <= n <= 31	UQSHL Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
uint32x4_t vqshlq_n_u32(uint32x4_t a, const int n)	a -> Vn.4S 0 <= n <= 31	UQSHL Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64x1_t vqshl_n_u64(uint64x1_t a, const int n)	a -> Dn 0 <= n <= 63	UQSHL Dd,Dn,#n	Dd -> result	v7/A32/A64
uint64x2_t vqshlq_n_u64(uint64x2_t a, const int n)	a -> Vn.2D 0 <= n <= 63	UQSHL Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
int8_t vqshlb_n_s8(int8_t a, const int n)	a -> Bn 0 <= n <= 7	SQSHL Bd,Bn,#n	Bd -> result	A64
int16_t vqshlh_n_s16(int16_t a, const int n)	a -> Hn 0 <= n <= 15	SQSHL Hd,Hn,#n	Hd -> result	A64
int32_t vqshls_n_s32(int32_t a, const int n)	a -> Sn 0 <= n <= 31	SQSHL Sd,Sn,#n	Sd -> result	A64
int64_t vqshld_n_s64(int64_t a, const int n)	a -> Dn 0 <= n <= 63	SQSHL Dd,Dn,#n	Dd -> result	A64
uint8_t vqshlb_n_u8(uint8_t a, const int n)	a -> Bn 0 <= n <= 7	UQSHL Bd,Bn,#n	Bd -> result	A64
uint16_t vqshlh_n_u16(uint16_t a, const int n)	a -> Hn 0 <= n <= 15	UQSHL Hd,Hn,#n	Hd -> result	A64
uint32_t vqshls_n_u32(uint32_t a, const int n)	a -> Sn 0 <= n <= 31	UQSHL Sd,Sn,#n	Sd -> result	A64
uint64_t vqshld_n_u64(uint64_t a, const int n)	a -> Dn 0 <= n <= 63	UQSHL Dd,Dn,#n	Dd -> result	A64
uint8x8_t vqshlu_n_s8(int8x8_t a, const int n)	a -> Vn.8B 0 <= n <= 7	SQSHLU Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
uint8x16_t vqshluq_n_s8(int8x16_t a, const int n)	a -> Vn.16B 0 <= n <= 7	SQSHLU Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
uint16x4_t vqshlu_n_s16(int16x4_t a, const int n)	a -> Vn.4H 0 <= n <= 15	SQSHLU Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
uint16x8_t vqshluq_n_s16(int16x8_t a, const int n)	a -> Vn.8H 0 <= n <= 15	SQSHLU Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
uint32x2_t vqshlu_n_s32(int32x2_t a, const int n)	a -> Vn.2S 0 <= n <= 31	SQSHLU Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
uint32x4_t vqshluq_n_s32(int32x4_t a, const int n)	a -> Vn.4S 0 <= n <= 31	SQSHLU Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
uint64x1_t vqshlu_n_s64(int64x1_t a, const int n)	a -> Dn 0 <= n <= 63	SQSHLU Dd,Dn,#n	Dd -> result	v7/A32/A64
uint64x2_t vqshluq_n_s64(int64x2_t a, const int n)	a -> Vn.2D 0 <= n <= 63	SQSHLU Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
uint8_t vqshlub_n_s8(int8_t a, const int n)	a -> Bn 0 <= n <= 7	SQSHLU Bd,Bn,#n	Bd -> result	A64
uint16_t vqshluh_n_s16(int16_t a, const int n)	a -> Hn 0 <= n <= 15	SQSHLU Hd,Hn,#n	Hd -> result	A64
uint32_t vqshlus_n_s32(int32_t a, const int n)	a -> Sn 0 <= n <= 31	SQSHLU Sd,Sn,#n	Sd -> result	A64
uint64_t vqshlud_n_s64(int64_t a, const int n)	a -> Dn 0 <= n <= 63	SQSHLU Dd,Dn,#n	Dd -> result	A64
int8x8_t vshrn_n_s16(int16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 8	SHRN Vd.8B,Vn.8H,#n	Vd.8B -> result	v7/A32/A64
int16x4_t vshrn_n_s32(int32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 16	SHRN Vd.4H,Vn.4S,#n	Vd.4H -> result	v7/A32/A64
int32x2_t vshrn_n_s64(int64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 32	SHRN Vd.2S,Vn.2D,#n	Vd.2S -> result	v7/A32/A64
uint8x8_t vshrn_n_u16(uint16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 8	SHRN Vd.8B,Vn.8H,#n	Vd.8B -> result	v7/A32/A64
uint16x4_t vshrn_n_u32(uint32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 16	SHRN Vd.4H,Vn.4S,#n	Vd.4H -> result	v7/A32/A64
uint32x2_t vshrn_n_u64(uint64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 32	SHRN Vd.2S,Vn.2D,#n	Vd.2S -> result	v7/A32/A64
int8x16_t vshrn_high_n_s16(int8x8_t r, int16x8_t a, const int n)	r -> Vd.8B a -> Vn.8H 1 <= n <= 8	SHRN2 Vd.16B,Vn.8H,#n	Vd.16B -> result	A64
int16x8_t vshrn_high_n_s32(int16x4_t r, int32x4_t a, const int n)	r -> Vd.4H a -> Vn.4S 1 <= n <= 16	SHRN2 Vd.8H,Vn.4S,#n	Vd.8H -> result	A64
int32x4_t vshrn_high_n_s64(int32x2_t r, int64x2_t a, const int n)	r -> Vd.2S a -> Vn.2D 1 <= n <= 32	SHRN2 Vd.4S,Vn.2D,#n	Vd.4S -> result	A64
uint8x16_t vshrn_high_n_u16(uint8x8_t r, uint16x8_t a, const int n)	r -> Vd.8B a -> Vn.8H 1 <= n <= 8	SHRN2 Vd.16B,Vn.8H,#n	Vd.16B -> result	A64
uint16x8_t vshrn_high_n_u32(uint16x4_t r, uint32x4_t a, const int n)	r -> Vd.4H a -> Vn.4S 1 <= n <= 16	SHRN2 Vd.8H,Vn.4S,#n	Vd.8H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t vshrn_high_n_u64(uint32x2_t r, uint64x2_t a, const int n)	r -> Vd.2S a -> Vn.2D 1 <= n <= 32	SHRN2 Vd.4S,Vn.2D,#n	Vd.4S -> result	A64
uint8x8_t vqshrun_n_s16(int16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 8	SQSHRUN Vd.8B,Vn.8H,#n	Vd.8B -> result	v7/A32/A64
uint16x4_t vqshrun_n_s32(int32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 16	SQSHRUN Vd.4H,Vn.4S,#n	Vd.4H -> result	v7/A32/A64
uint32x2_t vqshrun_n_s64(int64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 32	SQSHRUN Vd.2S,Vn.2D,#n	Vd.2S -> result	v7/A32/A64
uint8_t vqshrunh_n_s16(int16_t a, const int n)	a -> Hn 1 <= n <= 8	SQSHRUN Bd,Hn,#n	Bd -> result	A64
uint16_t vqshruns_n_s32(int32_t a, const int n)	a -> Sn 1 <= n <= 16	SQSHRUN Hd,Sn,#n	Hd -> result	A64
uint32_t vqshrund_n_s64(int64_t a, const int n)	a -> Dn 1 <= n <= 32	SQSHRUN Sd,Dn,#n	Sd -> result	A64
uint8x16_t vqshrun_high_n_s16(uint8x8_t r, int16x8_t a, const int n)	r -> Vd.8B a -> Vn.8H 1 <= n <= 8	SQSHRUN2 Vd.16B,Vn.8H,#n	Vd.16B -> result	A64
uint16x8_t vqshrun_high_n_s32(uint16x4_t r, int32x4_t a, const int n)	r -> Vd.4H a -> Vn.4S 1 <= n <= 16	SQSHRUN2 Vd.8H,Vn.4S,#n	Vd.8H -> result	A64
uint32x4_t vqshrun_high_n_s64(uint32x2_t r, int64x2_t a, const int n)	r -> Vd.2S a -> Vn.2D 1 <= n <= 32	SQSHRUN2 Vd.4S,Vn.2D,#n	Vd.4S -> result	A64
uint8x8_t vqsrshrun_n_s16(int16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 8	SQRSHRUN Vd.8B,Vn.8H,#n	Vd.8B -> result	v7/A32/A64
uint16x4_t vqsrshrun_n_s32(int32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 16	SQRSHRUN Vd.4H,Vn.4S,#n	Vd.4H -> result	v7/A32/A64
uint32x2_t vqsrshrun_n_s64(int64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 32	SQRSHRUN Vd.2S,Vn.2D,#n	Vd.2S -> result	v7/A32/A64
uint8_t vqsrshrunh_n_s16(int16_t a, const int n)	a -> Hn 1 <= n <= 8	SQRSHRUN Bd,Hn,#n	Bd -> result	A64
uint16_t vqsrshruns_n_s32(int32_t a, const int n)	a -> Sn 1 <= n <= 16	SQRSHRUN Hd,Sn,#n	Hd -> result	A64
uint32_t vqsrshrund_n_s64(int64_t a, const int n)	a -> Dn 1 <= n <= 32	SQRSHRUN Sd,Dn,#n	Sd -> result	A64
uint8x16_t vqsrshrun_high_n_s16(uint8x8_t r, int16x8_t a, const int n)	r -> Vd.8B a -> Vn.8H 1 <= n <= 8	SQRSHRUN2 Vd.16B,Vn.8H,#n	Vd.16B -> result	A64
uint16x8_t vqsrshrun_high_n_s32(uint16x4_t r, int32x4_t a, const int n)	r -> Vd.4H a -> Vn.4S 1 <= n <= 16	SQRSHRUN2 Vd.8H,Vn.4S,#n	Vd.8H -> result	A64
uint32x4_t vqsrshrun_high_n_s64(uint32x2_t r, int64x2_t a, const int n)	r -> Vd.2S a -> Vn.2D 1 <= n <= 32	SQRSHRUN2 Vd.4S,Vn.2D,#n	Vd.4S -> result	A64
int8x8_t vqshrn_n_s16(int16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 8	SQSHRN Vd.8B,Vn.8H,#n	Vd.8B -> result	v7/A32/A64
int16x4_t vqshrn_n_s32(int32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 16	SQSHRN Vd.4H,Vn.4S,#n	Vd.4H -> result	v7/A32/A64
int32x2_t vqshrn_n_s64(int64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 32	SQSHRN Vd.2S,Vn.2D,#n	Vd.2S -> result	v7/A32/A64
uint8x8_t vqshrn_u16(uint16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 8	UQSHRN Vd.8B,Vn.8H,#n	Vd.8B -> result	v7/A32/A64
uint16x4_t vqshrn_u32(uint32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 16	UQSHRN Vd.4H,Vn.4S,#n	Vd.4H -> result	v7/A32/A64
uint32x2_t vqshrn_u64(uint64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 32	UQSHRN Vd.2S,Vn.2D,#n	Vd.2S -> result	v7/A32/A64
int8_t vqshrn_h_n_s16(int16_t a, const int n)	a -> Hn 1 <= n <= 8	SQSHRN Bd,Hn,#n	Bd -> result	A64
int16_t vqshrn_h_n_s32(int32_t a, const int n)	a -> Sn 1 <= n <= 16	SQSHRN Hd,Sn,#n	Hd -> result	A64
int32_t vqshrn_d_n_s64(int64_t a, const int n)	a -> Dn 1 <= n <= 32	SQSHRN Sd,Dn,#n	Sd -> result	A64
uint8_t vqshrn_h_n_u16(uint16_t a, const int n)	a -> Hn 1 <= n <= 8	UQSHRN Bd,Hn,#n	Bd -> result	A64
uint16_t vqshrn_h_n_u32(uint32_t a, const int n)	a -> Sn 1 <= n <= 16	UQSHRN Hd,Sn,#n	Hd -> result	A64
uint32_t vqshrn_d_n_u64(uint64_t a, const int n)	a -> Dn 1 <= n <= 32	UQSHRN Sd,Dn,#n	Sd -> result	A64
int8x16_t vqshrn_high_n_s16(int8x8_t r, int16x8_t a, const int n)	r -> Vd.8B a -> Vn.8H 1 <= n <= 8	SQSHRN2 Vd.16B,Vn.8H,#n	Vd.16B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t vqshrn_high_n_s32(int16x4_t r, int32x4_t a, const int n)	r -> Vd.4H a -> Vn.4S 1 <= n <= 16	SQSHRN2 Vd.8H,Vn.4S,#n	Vd.8H -> result	A64
int32x4_t vqshrn_high_n_s64(int32x2_t r, int64x2_t a, const int n)	r -> Vd.2S a -> Vn.2D 1 <= n <= 32	SQSHRN2 Vd.4S,Vn.2D,#n	Vd.4S -> result	A64
uint8x16_t vqshrn_high_n_u16(uint8x8_t r, uint16x8_t a, const int n)	r -> Vd.8B a -> Vn.8H 1 <= n <= 8	UQSHRN2 Vd.16B,Vn.8H,#n	Vd.16B -> result	A64
uint16x8_t vqshrn_high_n_u32(uint16x4_t r, uint32x4_t a, const int n)	r -> Vd.4H a -> Vn.4S 1 <= n <= 16	UQSHRN2 Vd.8H,Vn.4S,#n	Vd.8H -> result	A64
uint32x4_t vqshrn_high_n_u64(uint32x2_t r, uint64x2_t a, const int n)	r -> Vd.2S a -> Vn.2D 1 <= n <= 32	UQSHRN2 Vd.4S,Vn.2D,#n	Vd.4S -> result	A64
int8x8_t vrshrn_n_s16(int16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 8	RSHRN Vd.8B,Vn.8H,#n	Vd.8B -> result	v7/A32/A64
int16x4_t vrshrn_n_s32(int32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 16	RSHRN Vd.4H,Vn.4S,#n	Vd.4H -> result	v7/A32/A64
int32x2_t vrshrn_n_s64(int64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 32	RSHRN Vd.2S,Vn.2D,#n	Vd.2S -> result	v7/A32/A64
uint8x8_t vrshrn_n_u16(uint16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 8	RSHRN Vd.8B,Vn.8H,#n	Vd.8B -> result	v7/A32/A64
uint16x4_t vrshrn_n_u32(uint32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 16	RSHRN Vd.4H,Vn.4S,#n	Vd.4H -> result	v7/A32/A64
uint32x2_t vrshrn_n_u64(uint64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 32	RSHRN Vd.2S,Vn.2D,#n	Vd.2S -> result	v7/A32/A64
int8x16_t vrshrn_high_n_s16(int8x8_t r, int16x8_t a, const int n)	r -> Vd.8B a -> Vn.8H 1 <= n <= 8	RSHRN2 Vd.16B,Vn.8H,#n	Vd.16B -> result	A64
int16x8_t vrshrn_high_n_s32(int16x4_t r, int32x4_t a, const int n)	r -> Vd.4H a -> Vn.4S 1 <= n <= 16	RSHRN2 Vd.8H,Vn.4S,#n	Vd.8H -> result	A64
int32x4_t vrshrn_high_n_s64(int32x2_t r, int64x2_t a, const int n)	r -> Vd.2S a -> Vn.2D 1 <= n <= 32	RSHRN2 Vd.4S,Vn.2D,#n	Vd.4S -> result	A64
uint8x16_t vrshrn_high_n_u16(uint8x8_t r, uint16x8_t a, const int n)	r -> Vd.8B a -> Vn.8H 1 <= n <= 8	RSHRN2 Vd.16B,Vn.8H,#n	Vd.16B -> result	A64
uint16x8_t vrshrn_high_n_u32(uint16x4_t r, uint32x4_t a, const int n)	r -> Vd.4H a -> Vn.4S 1 <= n <= 16	RSHRN2 Vd.8H,Vn.4S,#n	Vd.8H -> result	A64
uint32x4_t vrshrn_high_n_u64(uint32x2_t r, uint64x2_t a, const int n)	r -> 32(Vd) a -> Vn.2D 1 <= n <= 32	RSHRN2 Vd.4S,Vn.2D,#n	Vd.4S -> result	A64
int8x8_t vqrshrn_n_s16(int16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 8	SQRSHRN Vd.8B,Vn.8H,#n	Vd.8B -> result	v7/A32/A64
int16x4_t vqrshrn_n_s32(int32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 16	SQRSHRN Vd.4H,Vn.4S,#n	Vd.4H -> result	v7/A32/A64
int32x2_t vqrshrn_n_s64(int64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 32	SQRSHRN Vd.2S,Vn.2D,#n	Vd.2S -> result	v7/A32/A64
uint8x8_t vqrshrn_n_u16(uint16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 8	UQRSHRN Vd.8B,Vn.8H,#n	Vd.8B -> result	v7/A32/A64
uint16x4_t vqrshrn_n_u32(uint32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 16	UQRSHRN Vd.4H,Vn.4S,#n	Vd.4H -> result	v7/A32/A64
uint32x2_t vqrshrn_n_u64(uint64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 32	UQRSHRN Vd.2S,Vn.2D,#n	Vd.2S -> result	v7/A32/A64
int8_t vqrshrnh_n_s16(int16_t a, const int n)	a -> Hn 1 <= n <= 8	SQRSHRN Bd,Hn,#n	Bd -> result	A64
int16_t vqrshrns_n_s32(int32_t a, const int n)	a -> Sn 1 <= n <= 16	SQRSHRN Hd,Sn,#n	Hd -> result	A64
int32_t vqrshrnd_n_s64(int64_t a, const int n)	a -> Dn 1 <= n <= 32	SQRSHRN Sd,Dn,#n	Sd -> result	A64
uint8_t vqrshrnh_n_u16(uint16_t a, const int n)	a -> Hn 1 <= n <= 8	UQRSHRN Bd,Hn,#n	Bd -> result	A64
uint16_t vqrshrns_n_u32(uint32_t a, const int n)	a -> Sn 1 <= n <= 16	UQRSHRN Hd,Sn,#n	Hd -> result	A64
uint32_t vqrshrnd_n_u64(uint64_t a, const int n)	a -> Dn 1 <= n <= 32	UQRSHRN Sd,Dn,#n	Sd -> result	A64
int8x16_t vqrshrn_high_n_s16(int8x8_t r, int16x8_t a, const int n)	r -> Vd.8B a -> Vn.8H 1 <= n <= 8	SQRSHRN2 Vd.16B,Vn.8H,#n	Vd.16B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t vqsrshr_n_high_n_s32(int16x4_t r, int32x4_t a, const int n)	r -> Vd.4H a -> Vn.4S 1 <= n <= 16	SQRSHRN2 Vd.8H,Vn.4S,#n	Vd.8H -> result	A64
int32x4_t vqsrshr_n_high_n_s64(int32x2_t r, int64x2_t a, const int n)	r -> Vd.2S a -> Vn.2D 1 <= n <= 32	SQRSHRN2 Vd.4S,Vn.2D,#n	Vd.4S -> result	A64
uint8x16_t vqsrshr_n_high_n_u16(uint8x8_t r, uint16x8_t a, const int n)	r -> Vd.8B a -> Vn.8H 1 <= n <= 8	UQRSHRN2 Vd.16B,Vn.8H,#n	Vd.16B -> result	A64
uint16x8_t vqsrshr_n_high_n_u32(uint16x4_t r, uint32x4_t a, const int n)	r -> Vd.4H a -> Vn.4S 1 <= n <= 16	UQRSHRN2 Vd.8H,Vn.4S,#n	Vd.8H -> result	A64
uint32x4_t vqsrshr_n_high_n_u64(uint32x2_t r, uint64x2_t a, const int n)	r -> Vd.2S a -> Vn.2D 1 <= n <= 32	UQRSHRN2 Vd.4S,Vn.2D,#n	Vd.4S -> result	A64
int16x8_t vshll_n_s8(int8x8_t a, const int n)	a -> Vn.8B 0 <= n <= 7	SSHLL Vd.8H,Vn.8B,#n	Vd.8H -> result	v7/A32/A64
int32x4_t vshll_n_s16(int16x4_t a, const int n)	a -> Vn.4H 0 <= n <= 15	SSHLL Vd.4S,Vn.4H,#n	Vd.4S -> result	v7/A32/A64
int64x2_t vshll_n_s32(int32x2_t a, const int n)	a -> Vn.2S 0 <= n <= 31	SSHLL Vd.2D,Vn.2S,#n	Vd.2D -> result	v7/A32/A64
uint16x8_t vshll_n_u8(uint8x8_t a, const int n)	a -> Vn.8B 0 <= n <= 7	USHLL Vd.8H,Vn.8B,#n	Vd.8H -> result	v7/A32/A64
uint32x4_t vshll_n_u16(uint16x4_t a, const int n)	a -> Vn.4H 0 <= n <= 15	USHLL Vd.4S,Vn.4H,#n	Vd.4S -> result	v7/A32/A64
uint64x2_t vshll_n_u32(uint32x2_t a, const int n)	a -> Vn.2S 0 <= n <= 31	USHLL Vd.2D,Vn.2S,#n	Vd.2D -> result	v7/A32/A64
int16x8_t vshll_high_n_s8(int8x16_t a, const int n)	a -> Vn.16B 0 <= n <= 7	SSHLL2 Vd.8H,Vn.16B,#n	Vd.8H -> result	A64
int32x4_t vshll_high_n_s16(int16x8_t a, const int n)	a -> Vn.8H 0 <= n <= 15	SSHLL2 Vd.4S,Vn.8H,#n	Vd.4S -> result	A64
int64x2_t vshll_high_n_s32(int32x4_t a, const int n)	a -> Vn.4S 0 <= n <= 31	SSHLL2 Vd.2D,Vn.4S,#n	Vd.2D -> result	A64
uint16x8_t vshll_high_n_u8(uint8x16_t a, const int n)	a -> Vn.16B 0 <= n <= 7	USHLL2 Vd.8H,Vn.16B,#n	Vd.8H -> result	A64
uint32x4_t vshll_high_n_u16(uint16x8_t a, const int n)	a -> Vn.8H 0 <= n <= 15	USHLL2 Vd.4S,Vn.8H,#n	Vd.4S -> result	A64
uint64x2_t vshll_high_n_u32(uint32x4_t a, const int n)	a -> Vn.4S 0 <= n <= 31	USHLL2 Vd.2D,Vn.4S,#n	Vd.2D -> result	A64
int16x8_t vshll_n_s8(int8x8_t a, const int n)	a -> Vn.8B n == 8	SHLL Vd.8H,Vn.8B,#n	Vd.8H -> result	v7/A32/A64
int32x4_t vshll_n_s16(int16x4_t a, const int n)	a -> Vn.4H n == 16	SHLL Vd.4S,Vn.4H,#n	Vd.4S -> result	v7/A32/A64
int64x2_t vshll_n_s32(int32x2_t a, const int n)	a -> Vn.2S n == 32	SHLL Vd.2D,Vn.2S,#n	Vd.2D -> result	v7/A32/A64
uint16x8_t vshll_n_u8(uint8x8_t a, const int n)	a -> Vn.8B n == 8	SHLL Vd.8H,Vn.8B,#n	Vd.8H -> result	v7/A32/A64
uint32x4_t vshll_n_u16(uint16x4_t a, const int n)	a -> Vn.4H n == 16	SHLL Vd.4S,Vn.4H,#n	Vd.4S -> result	v7/A32/A64
uint64x2_t vshll_n_u32(uint32x2_t a, const int n)	a -> Vn.2S n == 32	SHLL Vd.2D,Vn.2S,#n	Vd.2D -> result	v7/A32/A64
int16x8_t vshll_high_n_s8(int8x16_t a, const int n)	a -> Vn.16B n == 8	SHLL2 Vd.8H,Vn.16B,#n	Vd.8H -> result	A64
int32x4_t vshll_high_n_s16(int16x8_t a, const int n)	a -> Vn.8H n == 16	SHLL2 Vd.4S,Vn.8H,#n	Vd.4S -> result	A64
int64x2_t vshll_high_n_s32(int32x4_t a, const int n)	a -> Vn.4S n == 32	SHLL2 Vd.2D,Vn.4S,#n	Vd.2D -> result	A64
uint16x8_t vshll_high_n_u8(uint8x16_t a, const int n)	a -> Vn.16B n == 8	SHLL2 Vd.8H,Vn.16B,#n	Vd.8H -> result	A64
uint32x4_t vshll_high_n_u16(uint16x8_t a, const int n)	a -> Vn.8H n == 16	SHLL2 Vd.4S,Vn.8H,#n	Vd.4S -> result	A64
uint64x2_t vshll_high_n_u32(uint32x4_t a, const int n)	a -> Vn.4S n == 32	SHLL2 Vd.2D,Vn.4S,#n	Vd.2D -> result	A64
int8x8_t vsri_n_s8(int8x8_t a, int8x8_t b, const int n)	a -> Vd.8B b -> Vn.8B 1 <= n <= 8	SRI Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
int8x16_t vsriq_n_s8(int8x16_t a, int8x16_t b, const int n)	a -> Vd.16B b -> Vn.16B 1 <= n <= 8	SRI Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
int16x4_t vsri_n_s16(int16x4_t a, int16x4_t b, const int n)	a -> Vd.4H b -> Vn.4H 1 <= n <= 16	SRI Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t vsriq_n_s16(int16x8_t a, int16x8_t b, const int n)	a -> Vd.8H b -> Vn.8H 1 <= n <= 16	SRI Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
int32x2_t vsri_n_s32(int32x2_t a, int32x2_t b, const int n)	a -> Vd.2S b -> Vn.2S 1 <= n <= 32	SRI Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
int32x4_t vsriq_n_s32(int32x4_t a, int32x4_t b, const int n)	a -> Vd.4S b -> Vn.4S 1 <= n <= 32	SRI Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
int64x1_t vsri_n_s64(int64x1_t a, int64x1_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	SRI Dd,Dn,#n	Dd -> result	v7/A32/A64
int64x2_t vsriq_n_s64(int64x2_t a, int64x2_t b, const int n)	a -> Vd.2D b -> Vn.2D 1 <= n <= 64	SRI Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
uint8x8_t vsri_u8(uint8x8_t a, uint8x8_t b, const int n)	a -> Vd.8B b -> Vn.8B 1 <= n <= 8	SRI Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
uint8x16_t vsriq_u8(uint8x16_t a, uint8x16_t b, const int n)	a -> Vd.16B b -> Vn.16B 1 <= n <= 8	SRI Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
uint16x4_t vsri_n_u16(uint16x4_t a, uint16x4_t b, const int n)	a -> Vd.4H b -> Vn.4H 1 <= n <= 16	SRI Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
uint16x8_t vsriq_n_u16(uint16x8_t a, uint16x8_t b, const int n)	a -> Vd.8H b -> Vn.8H 1 <= n <= 16	SRI Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
uint32x2_t vsri_n_u32(uint32x2_t a, uint32x2_t b, const int n)	a -> Vd.2S b -> Vn.2S 1 <= n <= 32	SRI Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
uint32x4_t vsriq_n_u32(uint32x4_t a, uint32x4_t b, const int n)	a -> Vd.4S b -> Vn.4S 1 <= n <= 32	SRI Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
uint64x1_t vsri_n_u64(uint64x1_t a, uint64x1_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	SRI Dd,Dn,#n	Dd -> result	v7/A32/A64
uint64x2_t vsriq_n_u64(uint64x2_t a, uint64x2_t b, const int n)	a -> Vd.2D b -> Vn.2D 1 <= n <= 64	SRI Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
poly64x1_t vsri_n_p64(poly64x1_t a, poly64x1_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	SRI Dd,Dn,#n	Dd -> result	A32/A64
poly64x2_t vsriq_n_p64(poly64x2_t a, poly64x2_t b, const int n)	a -> Vd.2D b -> Vn.2D 1 <= n <= 64	SRI Vd.2D,Vn.2D,#n	Vd.2D -> result	A32/A64
poly8x8_t vsri_n_p8(poly8x8_t a, poly8x8_t b, const int n)	a -> Vd.8B b -> Vn.8B 1 <= n <= 8	SRI Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
poly8x16_t vsriq_n_p8(poly8x16_t a, poly8x16_t b, const int n)	a -> Vd.16B b -> Vn.16B 1 <= n <= 8	SRI Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
poly16x4_t vsri_n_p16(poly16x4_t a, poly16x4_t b, const int n)	a -> Vd.4H b -> Vn.4H 1 <= n <= 16	SRI Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
poly16x8_t vsriq_n_p16(poly16x8_t a, poly16x8_t b, const int n)	a -> Vd.8H b -> Vn.8H 1 <= n <= 16	SRI Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
int64_t vsrid_n_s64(int64_t a, int64_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	SRI Dd,Dn,#n	Dd -> result	A64
uint64_t vsrid_n_u64(uint64_t a, uint64_t b, const int n)	a -> Dd b -> Dn 1 <= n <= 64	SRI Dd,Dn,#n	Dd -> result	A64
int8x8_t vsli_n_s8(int8x8_t a, int8x8_t b, const int n)	a -> Vd.8B b -> Vn.8B 0 <= n <= 7	SLI Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
int8x16_t vsliq_n_s8(int8x16_t a, int8x16_t b, const int n)	a -> Vd.16B b -> Vn.16B 0 <= n <= 7	SLI Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
int16x4_t vsli_n_s16(int16x4_t a, int16x4_t b, const int n)	a -> Vd.4H b -> Vn.4H 0 <= n <= 15	SLI Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t vsliq_n_s16(int16x8_t a, int16x8_t b, const int n)	a -> Vd.8H b -> Vn.8H 0 <= n <= 15	SLI Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
int32x2_t vsli_n_s32(int32x2_t a, int32x2_t b, const int n)	a -> Vd.2S b -> Vn.2S 0 <= n <= 31	SLI Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
int32x4_t vsliq_n_s32(int32x4_t a, int32x4_t b, const int n)	a -> Vd.4S b -> Vn.4S 0 <= n <= 31	SLI Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
int64x1_t vsli_n_s64(int64x1_t a, int64x1_t b, const int n)	a -> Dd b -> Dn 0 <= n <= 63	SLI Dd,Dn,#n	Dd -> result	v7/A32/A64
int64x2_t vsliq_n_s64(int64x2_t a, int64x2_t b, const int n)	a -> Vd.2D b -> Vn.2D 0 <= n <= 63	SLI Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
uint8x8_t vsli_u8(uint8x8_t a, uint8x8_t b, const int n)	a -> Vd.8B b -> Vn.8B 0 <= n <= 7	SLI Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
uint8x16_t vsliq_n_u8(uint8x16_t a, uint8x16_t b, const int n)	a -> Vd.16B b -> Vn.16B 0 <= n <= 7	SLI Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
uint16x4_t vsli_n_u16(uint16x4_t a, uint16x4_t b, const int n)	a -> Vd.4H b -> Vn.4H 0 <= n <= 15	SLI Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
uint16x8_t vsliq_n_u16(uint16x8_t a, uint16x8_t b, const int n)	a -> Vd.8H b -> Vn.8H 0 <= n <= 15	SLI Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
uint32x2_t vsli_n_u32(uint32x2_t a, uint32x2_t b, const int n)	a -> Vd.2S b -> Vn.2S 0 <= n <= 31	SLI Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
uint32x4_t vsliq_n_u32(uint32x4_t a, uint32x4_t b, const int n)	a -> Vd.4S b -> Vn.4S 0 <= n <= 31	SLI Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
uint64x1_t vsli_n_u64(uint64x1_t a, uint64x1_t b, const int n)	a -> Dd b -> Dn 0 <= n <= 63	SLI Dd,Dn,#n	Dd -> result	v7/A32/A64
uint64x2_t vsliq_n_u64(uint64x2_t a, uint64x2_t b, const int n)	a -> Vd.2D b -> Vn.2D 0 <= n <= 63	SLI Vd.2D,Vn.2D,#n	Vd.2D -> result	v7/A32/A64
poly64x1_t vsli_n_p64(poly64x1_t a, poly64x1_t b, const int n)	a -> Dd b -> Dn 0 <= n <= 63	SLI Dd,Dn,#n	Dd -> result	A32/A64
poly64x2_t vsliq_n_p64(poly64x2_t a, poly64x2_t b, const int n)	a -> Vd.2D b -> Vn.2D 0 <= n <= 63	SLI Vd.2D,Vn.2D,#n	Vd.2D -> result	A32/A64
poly8x8_t vsli_n_p8(poly8x8_t a, poly8x8_t b, const int n)	a -> Vd.8B b -> Vn.8B 0 <= n <= 7	SLI Vd.8B,Vn.8B,#n	Vd.8B -> result	v7/A32/A64
poly8x16_t vsliq_n_p8(poly8x16_t a, poly8x16_t b, const int n)	a -> Vd.16B b -> Vn.16B 0 <= n <= 7	SLI Vd.16B,Vn.16B,#n	Vd.16B -> result	v7/A32/A64
poly16x4_t vsli_n_p16(poly16x4_t a, poly16x4_t b, const int n)	a -> Vd.4H b -> Vn.4H 0 <= n <= 15	SLI Vd.4H,Vn.4H,#n	Vd.4H -> result	v7/A32/A64
poly16x8_t vsliq_n_p16(poly16x8_t a, poly16x8_t b, const int n)	a -> Vd.8H b -> Vn.8H 0 <= n <= 15	SLI Vd.8H,Vn.8H,#n	Vd.8H -> result	v7/A32/A64
int64_t vsli_d_n_s64(int64_t a, int64_t b, const int n)	a -> Dd b -> Dn 0 <= n <= 63	SLI Dd,Dn,#n	Dd -> result	A64
uint64_t vsli_d_n_u64(uint64_t a, uint64_t b, const int n)	a -> Dd b -> Dn 0 <= n <= 63	SLI Dd,Dn,#n	Dd -> result	A64
int32x2_t vcvtn_s32_f32(float32x2_t a)	a -> Vn.2S	FCVTZS Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vcvtnq_s32_f32(float32x4_t a)	a -> Vn.4S	FCVTZS Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint32x2_t vcvtn_u32_f32(float32x2_t a)	a -> Vn.2S	FCVTZU Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vcvtnq_u32_f32(float32x4_t a)	a -> Vn.4S	FCVTZU Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
int32x2_t vcvtn_s32_f32(float32x2_t a)	a -> Vn.2S	FCVTNS Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
int32x4_t vcvtnq_s32_f32(float32x4_t a)	a -> Vn.4S	FCVTNS Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
uint32x2_t vcvtn_u32_f32(float32x2_t a)	a -> Vn.2S	FCVTNU Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
uint32x4_t vcvtnq_u32_f32(float32x4_t a)	a -> Vn.4S	FCVTNU Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
int32x2_t vcvtnm_s32_f32(float32x2_t a)	a -> Vn.2S	FCVTMS Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
int32x4_t vcvtnmq_s32_f32(float32x4_t a)	a -> Vn.4S	FCVTMS Vd.4S,Vn.4S	Vd.4S -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x2_t vcvtn_u32_f32(float32x2_t a)	a -> Vn.2S	FCVTMU Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
uint32x4_t vcvtn_u32_f32(float32x4_t a)	a -> Vn.4S	FCVTMU Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
int32x2_t vcvtp_s32_f32(float32x2_t a)	a -> Vn.2S	FCVTPS Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
int32x4_t vcvtpq_s32_f32(float32x4_t a)	a -> Vn.4S	FCVTPS Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
uint32x2_t vcvtp_u32_f32(float32x2_t a)	a -> Vn.2S	FCVTPU Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
uint32x4_t vcvtpq_u32_f32(float32x4_t a)	a -> Vn.4S	FCVTPU Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
int32x2_t vcvta_s32_f32(float32x2_t a)	a -> Vn.2S	FCVTAS Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
int32x4_t vcvtaq_s32_f32(float32x4_t a)	a -> Vn.4S	FCVTAS Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
uint32x2_t vcvta_u32_f32(float32x2_t a)	a -> Vn.2S	FCVTAU Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
uint32x4_t vcvtaq_u32_f32(float32x4_t a)	a -> Vn.4S	FCVTAU Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
int32_t vcvts_s32_f32(float32_t a)	a -> Sn	FCVTZS Sd,Sn	Sd -> result	A64
uint32_t vcvts_u32_f32(float32_t a)	a -> Sn	FCVTZU Sd,Sn	Sd -> result	A64
int32_t vcvtns_s32_f32(float32_t a)	a -> Sn	FCVTNS Sd,Sn	Sd -> result	A64
uint32_t vcvtns_u32_f32(float32_t a)	a -> Sn	FCVTNU Sd,Sn	Sd -> result	A64
int32_t vcvtns_s32_f32(float32_t a)	a -> Sn	FCVTMS Sd,Sn	Sd -> result	A64
uint32_t vcvtns_u32_f32(float32_t a)	a -> Sn	FCVTMU Sd,Sn	Sd -> result	A64
int32_t vcvtps_s32_f32(float32_t a)	a -> Sn	FCVTPS Sd,Sn	Sd -> result	A64
uint32_t vcvtps_u32_f32(float32_t a)	a -> Sn	FCVTPU Sd,Sn	Sd -> result	A64
int32_t vcvtas_s32_f32(float32_t a)	a -> Sn	FCVTAS Sd,Sn	Sd -> result	A64
uint32_t vcvtas_u32_f32(float32_t a)	a -> Sn	FCVTAU Sd,Sn	Sd -> result	A64
int64x1_t vcvtd_s64_f64(float64x1_t a)	a -> Dn	FCVTZS Dd,Dn	Dd -> result	A64
int64x2_t vcvtdq_s64_f64(float64x2_t a)	a -> Vn.2D	FCVTZS Vd.2D,Vn.2D	Vd.2D -> result	A64
uint64x1_t vcvtd_u64_f64(float64x1_t a)	a -> Dn	FCVTZU Dd,Dn	Dd -> result	A64
uint64x2_t vcvtdq_u64_f64(float64x2_t a)	a -> Vn.2D	FCVTZU Vd.2D,Vn.2D	Vd.2D -> result	A64
int64x1_t vcvtn_s64_f64(float64x1_t a)	a -> Dn	FCVTNS Dd,Dn	Dd -> result	A64
int64x2_t vcvtnq_s64_f64(float64x2_t a)	a -> Vn.2D	FCVTNS Vd.2D,Vn.2D	Vd.2D -> result	A64
uint64x1_t vcvtn_u64_f64(float64x1_t a)	a -> Dn	FCVTNU Dd,Dn	Dd -> result	A64
uint64x2_t vcvtnq_u64_f64(float64x2_t a)	a -> Vn.2D	FCVTNU Vd.2D,Vn.2D	Vd.2D -> result	A64
int64x1_t vcvtnm_s64_f64(float64x1_t a)	a -> Dn	FCVTMS Dd,Dn	Dd -> result	A64
int64x2_t vcvtnmq_s64_f64(float64x2_t a)	a -> Vn.2D	FCVTMS Vd.2D,Vn.2D	Vd.2D -> result	A64
uint64x1_t vcvtnm_u64_f64(float64x1_t a)	a -> Dn	FCVTMU Dd,Dn	Dd -> result	A64
uint64x2_t vcvtnmq_u64_f64(float64x2_t a)	a -> Vn.2D	FCVTMU Vd.2D,Vn.2D	Vd.2D -> result	A64
int64x1_t vcvtp_s64_f64(float64x1_t a)	a -> Dn	FCVTPS Dd,Dn	Dd -> result	A64
int64x2_t vcvtpq_s64_f64(float64x2_t a)	a -> Vn.2D	FCVTPS Vd.2D,Vn.2D	Vd.2D -> result	A64
uint64x1_t vcvtp_u64_f64(float64x1_t a)	a -> Dn	FCVTPU Dd,Dn	Dd -> result	A64
uint64x2_t vcvtpq_u64_f64(float64x2_t a)	a -> Vn.2D	FCVTPU Vd.2D,Vn.2D	Vd.2D -> result	A64
int64x1_t vcvta_s64_f64(float64x1_t a)	a -> Dn	FCVTAS Dd,Dn	Dd -> result	A64
int64x2_t vcvtaq_s64_f64(float64x2_t a)	a -> Vn.2D	FCVTAS Vd.2D,Vn.2D	Vd.2D -> result	A64
uint64x1_t vcvta_u64_f64(float64x1_t a)	a -> Dn	FCVTAU Dd,Dn	Dd -> result	A64
uint64x2_t vcvtaq_u64_f64(float64x2_t a)	a -> Vn.2D	FCVTAU Vd.2D,Vn.2D	Vd.2D -> result	A64
int64_t vcvtd_s64_f64(float64_t a)	a -> Dn	FCVTZS Dd,Dn	Dd -> result	A64
uint64_t vcvtd_u64_f64(float64_t a)	a -> Dn	FCVTZU Dd,Dn	Dd -> result	A64
int64_t vcvtns_s64_f64(float64_t a)	a -> Dn	FCVTNS Dd,Dn	Dd -> result	A64
uint64_t vcvtns_u64_f64(float64_t a)	a -> Dn	FCVTNU Dd,Dn	Dd -> result	A64
int64_t vcvtnm_s64_f64(float64_t a)	a -> Dn	FCVTMS Dd,Dn	Dd -> result	A64
uint64_t vcvtnm_u64_f64(float64_t a)	a -> Dn	FCVTMU Dd,Dn	Dd -> result	A64
int64_t vcvtpd_s64_f64(float64_t a)	a -> Dn	FCVTPS Dd,Dn	Dd -> result	A64
uint64_t vcvtpd_u64_f64(float64_t a)	a -> Dn	FCVTPU Dd,Dn	Dd -> result	A64
int64_t vcvtdad_s64_f64(float64_t a)	a -> Dn	FCVTAS Dd,Dn	Dd -> result	A64
uint64_t vcvtdad_u64_f64(float64_t a)	a -> Dn	FCVTAU Dd,Dn	Dd -> result	A64
int32x2_t vcvtn_u32_f32(float32x2_t a, const int n)	a -> Vn.2S 1 <= n <= 32	FCVTZS Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
int32x4_t vcvtn_u32_f32(float32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 32	FCVTZS Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
uint32x2_t vcvtn_u32_f32(float32x2_t a, const int n)	a -> Vn.2S 1 <= n <= 32	FCVTZU Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
uint32x4_t vcvtn_u32_f32(float32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 32	FCVTZU Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
int32_t vcvts_n_s32_f32(float32_t a, const int n)	a -> Sn 1 <= n <= 32	FCVTZS Sd,Sn,#n	Sd -> result	A64
uint32_t vcvts_n_u32_f32(float32_t a, const int n)	a -> Sn 1 <= n <= 32	FCVTZU Sd,Sn,#n	Sd -> result	A64
int64x1_t vcvtd_n_s64_f64(float64x1_t a, const int n)	a -> Dn 1 <= n <= 64	FCVTZS Dd,Dn,#n	Dd -> result	A64
int64x2_t vcvtdq_n_s64_f64(float64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 64	FCVTZS Vd.2D,Vn.2D,#n	Vd.2D -> result	A64
uint64x1_t vcvtd_n_u64_f64(float64x1_t a, const int n)	a -> Dn 1 <= n <= 64	FCVTZU Dd,Dn,#n	Dd -> result	A64
uint64x2_t vcvtdq_n_u64_f64(float64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 64	FCVTZU Vd.2D,Vn.2D,#n	Vd.2D -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64_t vcvt_d_n_s64_f64(float64_t a, const int n)	a -> Dn 1 <= n <= 64	FCVTZS Dd,Dn,#n	Dd -> result	A64
uint64_t vcvt_d_n_u64_f64(float64_t a, const int n)	a -> Dn 1 <= n <= 64	FCVTZU Dd,Dn,#n	Dd -> result	A64
float32x2_t vcvt_f32_s32(int32x2_t a)	a -> Vn.2S	SCVTF Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vcvtq_f32_s32(int32x4_t a)	a -> Vn.4S	SCVTF Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
float32x2_t vcvt_f32_u32(uint32x2_t a)	a -> Vn.2S	UCVTF Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vcvtq_f32_u32(uint32x4_t a)	a -> Vn.4S	UCVTF Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
float32_t vcvt_s32_f32(int32_t a)	a -> Sn	SCVTF Sd,Sn	Sd -> result	A64
float32_t vcvt_s32_u32(uint32_t a)	a -> Sn	UCVTF Sd,Sn	Sd -> result	A64
float64x1_t vcvt_s64_f64(int64x1_t a)	a -> Dn	SCVTF Dd,Dn	Dd -> result	A64
float64x2_t vcvtq_f64_s64(int64x2_t a)	a -> Vn.2D	SCVTF Vd.2D,Vn.2D	Vd.2D -> result	A64
float64x1_t vcvt_f64_u64(uint64x1_t a)	a -> Dn	UCVTF Dd,Dn	Dd -> result	A64
float64x2_t vcvtq_f64_u64(uint64x2_t a)	a -> Vn.2D	UCVTF Vd.2D,Vn.2D	Vd.2D -> result	A64
float64_t vcvt_d_f64_s64(int64_t a)	a -> Dn	SCVTF Dd,Dn	Dd -> result	A64
float64_t vcvt_d_f64_u64(uint64_t a)	a -> Dn	UCVTF Dd,Dn	Dd -> result	A64
float32x2_t vcvt_n_f32_s32(int32x2_t a, const int n)	a -> Vn.2S 1 <= n <= 32	SCVTF Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
float32x4_t vcvtq_n_f32_s32(int32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 32	SCVTF Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
float32x2_t vcvt_n_f32_u32(uint32x2_t a, const int n)	a -> Vn.2S 1 <= n <= 32	UCVTF Vd.2S,Vn.2S,#n	Vd.2S -> result	v7/A32/A64
float32x4_t vcvtq_n_f32_u32(uint32x4_t a, const int n)	a -> Vn.4S 1 <= n <= 32	UCVTF Vd.4S,Vn.4S,#n	Vd.4S -> result	v7/A32/A64
float32_t vcvt_s_n_f32_s32(int32_t a, const int n)	a -> Sn 1 <= n <= 32	SCVTF Sd,Sn,#n	Sd -> result	A64
float32_t vcvt_s_n_f32_u32(uint32_t a, const int n)	a -> Sn 1 <= n <= 32	UCVTF Sd,Sn,#n	Sd -> result	A64
float64x1_t vcvt_n_f64_s64(int64x1_t a, const int n)	a -> Dn 1 <= n <= 64	SCVTF Dd,Dn,#n	Dd -> result	A64
float64x2_t vcvtq_n_f64_s64(int64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 64	SCVTF Vd.2D,Vn.2D,#n	Vd.2D -> result	A64
float64x1_t vcvt_n_f64_u64(uint64x1_t a, const int n)	a -> Dn 1 <= n <= 64	UCVTF Dd,Dn,#n	Dd -> result	A64
float64x2_t vcvtq_n_f64_u64(uint64x2_t a, const int n)	a -> Vn.2D 1 <= n <= 64	UCVTF Vd.2D,Vn.2D,#n	Vd.2D -> result	A64
float64_t vcvt_d_n_f64_s64(int64_t a, const int n)	a -> Dn 1 <= n <= 64	SCVTF Dd,Dn,#n	Dd -> result	A64
float64_t vcvt_d_n_f64_u64(uint64_t a, const int n)	a -> Dn 1 <= n <= 64	UCVTF Dd,Dn,#n	Dd -> result	A64
float16x4_t vcvt_f16_f32(float32x4_t a)	a -> Vn.4S	FCVTN Vd.4H,Vn.4S	Vd.4H -> result	v7/A32/A64
float16x8_t vcvt_high_f16_f32(float16x4_t r, float32x4_t a)	r -> Vd.4H a -> Vn.4S	FCVTN2 Vd.8H,Vn.4S	Vd.8H -> result	A64
float32x2_t vcvt_f32_f64(float64x2_t a)	a -> Vn.2D	FCVTN Vd.2S,Vn.2D	Vd.2S -> result	A64
float32x4_t vcvt_high_f32_f64(float32x2_t r, float64x2_t a)	r -> Vd.2S a -> Vn.2D	FCVTN2 Vd.4S,Vn.2D	Vd.4S -> result	A64
float32x4_t vcvt_f32_f16(float16x4_t a)	a -> Vn.4H	FCVTL Vd.4S,Vn.4H	Vd.4S -> result	v7/A32/A64
float32x4_t vcvt_high_f32_f16(float16x8_t a)	a -> Vn.8H	FCVTL2 Vd.4S,Vn.8H	Vd.4S -> result	A64
float64x2_t vcvt_f64_f32(float32x2_t a)	a -> Vn.2S	FCVTL Vd.2D,Vn.2S	Vd.2D -> result	A64
float64x2_t vcvt_high_f64_f32(float32x4_t a)	a -> Vn.4S	FCVTL2 Vd.2D,Vn.4S	Vd.2D -> result	A64
float32x2_t vcvt_x_f32_f64(float64x2_t a)	a -> Vn.2D	FCVTXN Vd.2S,Vn.2D	Vd.2S -> result	A64
float32_t vcvt_xd_f32_f64(float64_t a)	a -> Dn	FCVTXN Sd,Dn	Sd -> result	A64
float32x4_t vcvt_high_f32_f64(float32x2_t r, float64x2_t a)	r -> Vd.2S a -> Vn.2D	FCVTXN2 Vd.4S,Vn.2D	Vd.4S -> result	A64
float32x2_t vrnd_f32(float32x2_t a)	a -> Vn.2S	FRINTZ Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
float32x4_t vrndq_f32(float32x4_t a)	a -> Vn.4S	FRINTZ Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
float64x1_t vrnd_f64(float64x1_t a)	a -> Dn	FRINTZ Dd,Dn	Dd -> result	A64
float64x2_t vrndq_f64(float64x2_t a)	a -> Vn.2D	FRINTZ Vd.2D,Vn.2D	Vd.2D -> result	A64
float32x2_t vrndn_f32(float32x2_t a)	a -> Vn.2S	FRINTN Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
float32x4_t vrndnq_f32(float32x4_t a)	a -> Vn.4S	FRINTN Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
float64x1_t vrndn_f64(float64x1_t a)	a -> Dn	FRINTN Dd,Dn	Dd -> result	A32/A64
float64x2_t vrndnq_f64(float64x2_t a)	a -> Vn.2D	FRINTN Vd.2D,Vn.2D	Vd.2D -> result	A32/A64
float32_t vrndns_f32(float32_t a)	a -> Sn	FRINTN Sd,Sn	Sd -> result	A32/A64
float32x2_t vrndm_f32(float32x2_t a)	a -> Vn.2S	FRINTM Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
float32x4_t vrndmq_f32(float32x4_t a)	a -> Vn.4S	FRINTM Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
float64x1_t vrndm_f64(float64x1_t a)	a -> Dn	FRINTM Dd,Dn	Dd -> result	A64
float64x2_t vrndmq_f64(float64x2_t a)	a -> Vn.2D	FRINTM Vd.2D,Vn.2D	Vd.2D -> result	A64
float32x2_t vrndp_f32(float32x2_t a)	a -> Vn.2S	FRINTP Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
float32x4_t vrndpq_f32(float32x4_t a)	a -> Vn.4S	FRINTP Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
float64x1_t vrndp_f64(float64x1_t a)	a -> Dn	FRINTP Dd,Dn	Dd -> result	A64
float64x2_t vrndpq_f64(float64x2_t a)	a -> Vn.2D	FRINTP Vd.2D,Vn.2D	Vd.2D -> result	A64
float32x2_t vrnda_f32(float32x2_t a)	a -> Vn.2S	FRINTA Vd.2S,Vn.2S	Vd.2S -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4 t_vrndaq_f32(float32x4_t a)	a -> Vn.4S	FRINTA Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
float64x1 t_vrndad_f64(float64x1_t a)	a -> Dn	FRINTA Dd,Dn	Dd -> result	A64
float64x2 t_vrndaq_f64(float64x2_t a)	a -> Vn.2D	FRINTA Vd.2D,Vn.2D	Vd.2D -> result	A64
float32x2 t_vrndx_f32(float32x2_t a)	a -> Vn.2S	FRINTI Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
float32x4 t_vrndiq_f32(float32x4_t a)	a -> Vn.4S	FRINTI Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
float64x1 t_vrndi_f64(float64x1_t a)	a -> Dn	FRINTI Dd,Dn	Dd -> result	A64
float64x2 t_vrndiq_f64(float64x2_t a)	a -> Vn.2D	FRINTI Vd.2D,Vn.2D	Vd.2D -> result	A64
float32x2 t_vrndx_f32(float32x2_t a)	a -> Vn.2S	FRINTX Vd.2S,Vn.2S	Vd.2S -> result	A32/A64
float32x4 t_vrndxq_f32(float32x4_t a)	a -> Vn.4S	FRINTX Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
float64x1 t_vrndx_f64(float64x1_t a)	a -> Dn	FRINTX Dd,Dn	Dd -> result	A64
float64x2 t_vrndxq_f64(float64x2_t a)	a -> Vn.2D	FRINTX Vd.2D,Vn.2D	Vd.2D -> result	A64
int8x8 t_vmovn_s16(int16x8_t a)	a -> Vn.8H	XTN Vd.8B,Vn.8H	Vd.8B -> result	v7/A32/A64
int16x4 t_vmovn_s32(int32x4_t a)	a -> Vn.4S	XTN Vd.4H,Vn.4S	Vd.4H -> result	v7/A32/A64
int32x2 t_vmovn_s64(int64x2_t a)	a -> Vn.2D	XTN Vd.2S,Vn.2D	Vd.2S -> result	v7/A32/A64
uint8x8 t_vmovn_u16(uint16x8_t a)	a -> Vn.8H	XTN Vd.8B,Vn.8H	Vd.8B -> result	v7/A32/A64
uint16x4 t_vmovn_u32(uint32x4_t a)	a -> Vn.4S	XTN Vd.4H,Vn.4S	Vd.4H -> result	v7/A32/A64
uint32x2 t_vmovn_u64(uint64x2_t a)	a -> Vn.2D	XTN Vd.2S,Vn.2D	Vd.2S -> result	v7/A32/A64
int8x16_t_vmovn_high_s16(int8x8_t r, int16x8_t a)	r -> Vd.8B a -> Vn.8H	XTN2 Vd.16B,Vn.8H	Vd.16B -> result	v7/A32/A64
int16x8_t_vmovn_high_s32(int16x4_t r, int32x4_t a)	r -> Vd.4H a -> Vn.4S	XTN2 Vd.8H,Vn.4S	Vd.8H -> result	v7/A32/A64
int32x4_t_vmovn_high_s64(int32x2_t r, int64x2_t a)	r -> Vd.2S a -> Vn.2D	XTN2 Vd.4S,Vn.2D	Vd.4S -> result	v7/A32/A64
uint8x16_t_vmovn_high_u16(uint8x8_t r, uint16x8_t a)	r -> Vd.8B a -> Vn.8H	XTN2 Vd.16B,Vn.8H	Vd.16B -> result	v7/A32/A64
uint16x8_t_vmovn_high_u32(uint16x4_t r, uint32x4_t a)	r -> Vd.4H a -> Vn.4S	XTN2 Vd.8H,Vn.4S	Vd.8H -> result	v7/A32/A64
uint32x4_t_vmovn_high_u64(uint32x2_t r, uint64x2_t a)	r -> Vd.2S a -> Vn.2D	XTN2 Vd.4S,Vn.2D	Vd.4S -> result	v7/A32/A64
int16x8 t_vmovl_s8(int8x8_t a)	a -> Vn.8B	SSHLL Vd.8H,Vn.8B,#0	Vd.8H -> result	v7/A32/A64
int32x4 t_vmovl_s16(int16x4_t a)	a -> Vn.4H	SSHLL Vd.4S,Vn.4H,#0	Vd.4S -> result	v7/A32/A64
int64x2 t_vmovl_s32(int32x2_t a)	a -> Vn.2S	SSHLL Vd.2D,Vn.2S,#0	Vd.2D -> result	v7/A32/A64
uint16x8 t_vmovl_u8(uint8x8_t a)	a -> Vn.8B	USHLL Vd.8H,Vn.8B,#0	Vd.8H -> result	v7/A32/A64
uint32x4 t_vmovl_u16(uint16x4_t a)	a -> Vn.4H	USHLL Vd.4S,Vn.4H,#0	Vd.4S -> result	v7/A32/A64
uint64x2 t_vmovl_u32(uint32x2_t a)	a -> Vn.2S	USHLL Vd.2D,Vn.2S,#0	Vd.2D -> result	v7/A32/A64
int16x8 t_vmovl_high_s8(int8x16_t a)	a -> Vn.16B	SSHLL2 Vd.8H,Vn.16B,#0	Vd.8H -> result	A64
int32x4 t_vmovl_high_s16(int16x8_t a)	a -> Vn.8H	SSHLL2 Vd.4S,Vn.8H,#0	Vd.4S -> result	A64
int64x2 t_vmovl_high_s32(int32x4_t a)	a -> Vn.4S	SSHLL2 Vd.2D,Vn.4S,#0	Vd.2D -> result	A64
uint16x8 t_vmovl_high_u8(uint8x16_t a)	a -> Vn.16B	USHLL2 Vd.8H,Vn.16B,#0	Vd.8H -> result	A64
uint32x4 t_vmovl_high_u16(uint16x8_t a)	a -> Vn.8H	USHLL2 Vd.4S,Vn.8H,#0	Vd.4S -> result	A64
uint64x2 t_vmovl_high_u32(uint32x4_t a)	a -> Vn.4S	USHLL2 Vd.2D,Vn.4S,#0	Vd.2D -> result	A64
int8x8 t_vqmovn_s16(int16x8_t a)	a -> Vn.8H	SQXTN Vd.8B,Vn.8H	Vd.8B -> result	v7/A32/A64
int16x4 t_vqmovn_s32(int32x4_t a)	a -> Vn.4S	SQXTN Vd.4H,Vn.4S	Vd.4H -> result	v7/A32/A64
int32x2 t_vqmovn_s64(int64x2_t a)	a -> Vn.2D	SQXTN Vd.2S,Vn.2D	Vd.2S -> result	v7/A32/A64
uint8x8 t_vqmovn_u16(uint16x8_t a)	a -> Vn.8H	UQXTN Vd.8B,Vn.8H	Vd.8B -> result	v7/A32/A64
uint16x4 t_vqmovn_u32(uint32x4_t a)	a -> Vn.4S	UQXTN Vd.4H,Vn.4S	Vd.4H -> result	v7/A32/A64
uint32x2 t_vqmovn_u64(uint64x2_t a)	a -> Vn.2D	UQXTN Vd.2S,Vn.2D	Vd.2S -> result	v7/A32/A64
int8 t_vqmovnh_s16(int16_t a)	a -> Hn	SQXTN Bd,Hn	Bd -> result	A64
int16 t_vqmovns_s32(int32_t a)	a -> Sn	SQXTN Hd,Sn	Hd -> result	A64
int32 t_vqmovnd_s64(int64_t a)	a -> Dn	SQXTN Sd,Dn	Sd -> result	A64
uint8 t_vqmovnh_u16(uint16_t a)	a -> Hn	UQXTN Bd,Hn	Bd -> result	A64
uint16 t_vqmovns_u32(uint32_t a)	a -> Sn	UQXTN Hd,Sn	Hd -> result	A64
uint32 t_vqmovnd_u64(uint64_t a)	a -> Dn	UQXTN Sd,Dn	Sd -> result	A64
int8x16_t_vqmovn_high_s16(int8x8_t r, int16x8_t a)	r -> Vd.8B a -> Vn.8H	SQXTN2 Vd.16B,Vn.8H	Vd.16B -> result	A64
int16x8_t_vqmovn_high_s32(int16x4_t r, int32x4_t a)	r -> Vd.4H a -> Vn.4S	SQXTN2 Vd.8H,Vn.4S	Vd.8H -> result	A64
int32x4_t_vqmovn_high_s64(int32x2_t r, int64x2_t a)	r -> Vd.2S a -> Vn.2D	SQXTN2 Vd.4S,Vn.2D	Vd.4S -> result	A64
uint8x16_t_vqmovn_high_u16(uint8x8_t r, uint16x8_t a)	r -> Vd.8B a -> Vn.8H	UQXTN2 Vd.16B,Vn.8H	Vd.16B -> result	A64
uint16x8_t_vqmovn_high_u32(uint16x4_t r, uint32x4_t a)	r -> Vd.4H a -> Vn.4S	UQXTN2 Vd.8H,Vn.4S	Vd.8H -> result	A64
uint32x4_t_vqmovn_high_u64(uint32x2_t r, uint64x2_t a)	r -> Vd.2S a -> Vn.2D	UQXTN2 Vd.4S,Vn.2D	Vd.4S -> result	A64
uint8x8 t_vqmovun_s16(int16x8_t a)	a -> Vn.8H	SQXTUN Vd.8B,Vn.8H	Vd.8B -> result	v7/A32/A64
uint16x4 t_vqmovun_s32(int32x4_t a)	a -> Vn.4S	SQXTUN Vd.4H,Vn.4S	Vd.4H -> result	v7/A32/A64
uint32x2 t_vqmovun_s64(int64x2_t a)	a -> Vn.2D	SQXTUN Vd.2S,Vn.2D	Vd.2S -> result	v7/A32/A64
uint8 t_vqmovunh_s16(int16_t a)	a -> Hn	SQXTUN Bd,Hn	Bd -> result	A64
uint16 t_vqmovuns_s32(int32_t a)	a -> Sn	SQXTUN Hd,Sn	Hd -> result	A64
uint32 t_vqmovund_s64(int64_t a)	a -> Dn	SQXTUN Sd,Dn	Sd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t vqmovun_high_s16(uint8x8_t r, int16x8_t a)	r -> Vd.8B a -> Vn.8H	SQXTUN2 Vd.16B,Vn.8H	Vd.16B -> result	A64
uint16x8_t vqmovun_high_s32(uint16x4_t r, int32x4_t a)	r -> Vd.4H a -> Vn.4S	SQXTUN2 Vd.8H,Vn.4S	Vd.8H -> result	A64
uint32x4_t vqmovun_high_s64(uint32x2_t r, int64x2_t a)	r -> Vd.2S a -> Vn.2D	SQXTUN2 Vd.4S,Vn.2D	Vd.4S -> result	A64
int16x4_t vmla_lane_s16(int16x4_t a, int16x4_t b, int16x4_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	MLA Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	v7/A32/A64
int16x8_t vmlaq_lane_s16(int16x8_t a, int16x8_t b, int16x4_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	MLA Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	v7/A32/A64
int32x2_t vmla_lane_s32(int32x2_t a, int32x2_t b, int32x2_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	MLA Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	v7/A32/A64
int32x4_t vmlaq_lane_s32(int32x4_t a, int32x4_t b, int32x2_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	MLA Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	v7/A32/A64
uint16x4_t vmla_lane_u16(uint16x4_t a, uint16x4_t b, uint16x4_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	MLA Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	v7/A32/A64
uint16x8_t vmlaq_lane_u16(uint16x8_t a, uint16x8_t b, uint16x4_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	MLA Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	v7/A32/A64
uint32x2_t vmla_lane_u32(uint32x2_t a, uint32x2_t b, uint32x2_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	MLA Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	v7/A32/A64
uint32x4_t vmlaq_lane_u32(uint32x4_t a, uint32x4_t b, uint32x2_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	MLA Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	v7/A32/A64
float32x2_t vmla_lane_f32(float32x2_t a, float32x2_t b, float32x2_t v, const int lane)	0 <= lane <= 1	RESULT[i] = a[i] + (b[i] * v[lane]) for i = 0 to 1	N/A -> result	v7/A32/A64
float32x4_t vmlaq_lane_f32(float32x4_t a, float32x4_t b, float32x2_t v, const int lane)	0 <= lane <= 1	RESULT[i] = a[i] + (b[i] * v[lane]) for i = 0 to 3	N/A -> result	v7/A32/A64
int16x4_t vmla_laneq_s16(int16x4_t a, int16x4_t b, int16x8_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	MLA Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
int16x8_t vmlaq_laneq_s16(int16x8_t a, int16x8_t b, int16x8_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	MLA Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
int32x2_t vmla_laneq_s32(int32x2_t a, int32x2_t b, int32x4_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	MLA Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
int32x4_t vmlaq_laneq_s32(int32x4_t a, int32x4_t b, int32x4_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	MLA Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x4_t vmla_laneq_u16(uint16x4_t a, uint16x4_t b, uint16x8_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	MLA Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
uint16x8_t vmlaq_laneq_u16(uint16x8_t a, uint16x8_t b, uint16x8_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	MLA Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
uint32x2_t vmla_laneq_u32(uint32x2_t a, uint32x2_t b, uint32x4_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	MLA Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
uint32x4_t vmlaq_laneq_u32(uint32x4_t a, uint32x4_t b, uint32x4_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	MLA Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
float32x2_t vmla_laneq_f32(float32x2_t a, float32x2_t b, float32x4_t v, const int lane)	0 <= lane <= 3	RESULT[i] = a[i] + (b[i] * v[lane]) for i = 0 to 1	N/A -> result	A64
float32x4_t vmlaq_laneq_f32(float32x4_t a, float32x4_t b, float32x4_t v, const int lane)	0 <= lane <= 3	RESULT[i] = a[i] + (b[i] * v[lane]) for i = 0 to 3	N/A -> result	A64
int32x4_t vmlal_lane_s16(int32x4_t a, int16x4_t b, int16x4_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	SMLAL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	v7/A32/A64
int64x2_t vmlal_lane_s32(int64x2_t a, int32x2_t b, int32x2_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	SMLAL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	v7/A32/A64
uint32x4_t vmlal_lane_u16(uint32x4_t a, uint16x4_t b, uint16x4_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	UMLAL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	v7/A32/A64
uint64x2_t vmlal_lane_u32(uint64x2_t a, uint32x2_t b, uint32x2_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	UMLAL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	v7/A32/A64
int32x4_t vmlal_high_lane_s16(int32x4_t a, int16x8_t b, int16x4_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	SMLAL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vmlal_high_lane_s32(int64x2_t a, int32x4_t b, int32x2_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	SMLAL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
uint32x4_t vmlal_high_lane_u16(uint32x4_t a, uint16x8_t b, uint16x4_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	UMLAL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
uint64x2_t vmlal_high_lane_u32(uint64x2_t a, uint32x4_t b, uint32x2_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	UMLAL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
int32x4_t vmlal_laneq_s16(int32x4_t a, int16x4_t b, int16x8_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SMLAL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vmlal_laneq_s32(int64x2_t a, int32x2_t b, int32x4_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SMLAL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t vmlal_laneq_u16(uint32x4_t a, uint16x4_t b, uint16x8_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	UMLAL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A64
uint64x2_t vmlal_laneq_u32(uint64x2_t a, uint32x2_t b, uint32x4_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	UMLAL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	A64
int32x4_t vmlal_high_laneq_s16(int32x4_t a, int16x8_t b, int16x8_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	SMLAL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vmlal_high_laneq_s32(int64x2_t a, int32x4_t b, int32x4_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	SMLAL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
uint32x4_t vmlal_high_laneq_u16(uint32x4_t a, uint16x8_t b, uint16x8_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	UMLAL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
uint64x2_t vmlal_high_laneq_u32(uint64x2_t a, uint32x4_t b, uint32x4_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	UMLAL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
int32x4_t vqdmmlal_lane_s16(int32x4_t a, int16x4_t b, int16x4_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	SQDMLAL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	v7/A32/A64
int64x2_t vqdmmlal_lane_s32(int64x2_t a, int32x2_t b, int32x2_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	SQDMLAL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	v7/A32/A64
int32_t vqdmmlalh_lane_s16(int32_t a, int16_t b, int16x4_t v, const int lane)	a -> Sd b -> Hn v -> Vm.4H 0 <= lane <= 3	SQDMLAL Sd,Hn,Vm.H[lane]	Sd -> result	A64
int64_t vqdmmlals_lane_s32(int64_t a, int32_t b, int32x2_t v, const int lane)	a -> Dd b -> Sn v -> Vm.2S 0 <= lane <= 1	SQDMLAL Dd,Sn,Vm.S[lane]	Dd -> result	A64
int32x4_t vqdmmlal_high_lane_s16(int32x4_t a, int16x8_t b, int16x4_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	SQDMLAL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vqdmmlal_high_lane_s32(int64x2_t a, int32x4_t b, int32x2_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	SQDMLAL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
int32x4_t vqdmmlal_laneq_s16(int32x4_t a, int16x4_t b, int16x8_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SQDMLAL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vqdmmlal_laneq_s32(int64x2_t a, int32x2_t b, int32x4_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SQDMLAL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32_t vqdmmlalh_laneq_s16(int32_t a, int16_t b, int16x8_t v, const int lane)	a -> Sd b -> Hn v -> Vm.8H 0 <= lane <= 7	SQDMLAL Sd,Hn,Vm.H[lane]	Sd -> result	A64
int64_t vqdmmlals_laneq_s32(int64_t a, int32_t b, int32x4_t v, const int lane)	a -> Dd b -> Sn v -> Vm.4S 0 <= lane <= 3	SQDMLAL Dd,Sn,Vm.S[lane]	Dd -> result	A64
int32x4_t vqdmmlal_high_laneq_s16(int32x4_t a, int16x8_t b, int16x8_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	SQDMLAL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vqdmmlal_high_laneq_s32(int64x2_t a, int32x4_t b, int32x4_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	SQDMLAL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
int16x4_t vmls_lane_s16(int16x4_t a, int16x4_t b, int16x4_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	MLS Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	v7/A32/A64
int16x8_t vmlsq_lane_s16(int16x8_t a, int16x8_t b, int16x4_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	MLS Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	v7/A32/A64
int32x2_t vmls_lane_s32(int32x2_t a, int32x2_t b, int32x2_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	MLS Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	v7/A32/A64
int32x4_t vmlsq_lane_s32(int32x4_t a, int32x4_t b, int32x2_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	MLS Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	v7/A32/A64
uint16x4_t vmls_lane_u16(uint16x4_t a, uint16x4_t b, uint16x4_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	MLS Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	v7/A32/A64
uint16x8_t vmlsq_lane_u16(uint16x8_t a, uint16x8_t b, uint16x4_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	MLS Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	v7/A32/A64
uint32x2_t vmls_lane_u32(uint32x2_t a, uint32x2_t b, uint32x2_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	MLS Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	v7/A32/A64
uint32x4_t vmlsq_lane_u32(uint32x4_t a, uint32x4_t b, uint32x2_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	MLS Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	v7/A32/A64
float32x2_t vmls_lane_f32(float32x2_t a, float32x2_t b, float32x2_t v, const int lane)	0 <= lane <= 1	RESULT[I] = a[i] - (b[i] * v[lane]) for i = 0 to 1	N/A -> result	v7/A32/A64
float32x4_t vmlsq_lane_f32(float32x4_t a, float32x4_t b, float32x2_t v, const int lane)	0 <= lane <= 1	RESULT[I] = a[i] - (b[i] * v[lane]) for i = 0 to 3	N/A -> result	v7/A32/A64
int16x4_t vmls_laneq_s16(int16x4_t a, int16x4_t b, int16x8_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	MLS Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
int16x8_t vmlsq_laneq_s16(int16x8_t a, int16x8_t b, int16x8_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	MLS Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x2_t vmls_laneq_s32(int32x2_t a, int32x2_t b, int32x4_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	MLS Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
int32x4_t vmlsq_laneq_s32(int32x4_t a, int32x4_t b, int32x4_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	MLS Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
uint16x4_t vmls_laneq_u16(uint16x4_t a, uint16x4_t b, uint16x8_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	MLS Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
uint16x8_t vmlsq_laneq_u16(uint16x8_t a, uint16x8_t b, uint16x8_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	MLS Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
uint32x2_t vmls_laneq_u32(uint32x2_t a, uint32x2_t b, uint32x4_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	MLS Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
uint32x4_t vmlsq_laneq_u32(uint32x4_t a, uint32x4_t b, uint32x4_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	MLS Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
float32x2_t vmls_laneq_f32(float32x2_t a, float32x2_t b, float32x4_t v, const int lane)	0 <= lane <= 3	RESULT[I] = a[i] - (b[i] * v[lane]) for i = 0 to 1	N/A -> result	A64
float32x4_t vmlsq_laneq_f32(float32x4_t a, float32x4_t b, float32x4_t v, const int lane)	0 <= lane <= 3	RESULT[I] = a[i] - (b[i] * v[lane]) for i = 0 to 3	N/A -> result	A64
int32x4_t vmlsl_lane_s16(int32x4_t a, int16x4_t b, int16x4_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	SMLSL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	v7/A32/A64
int64x2_t vmlsl_lane_s32(int64x2_t a, int32x2_t b, int32x2_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	SMLSL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	v7/A32/A64
uint32x4_t vmlsl_lane_u16(uint32x4_t a, uint16x4_t b, uint16x4_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	UMLSL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	v7/A32/A64
uint64x2_t vmlsl_lane_u32(uint64x2_t a, uint32x2_t b, uint32x2_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	UMLSL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	v7/A32/A64
int32x4_t vmlsl_high_lane_s16(int32x4_t a, int16x8_t b, int16x4_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	SMLSL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vmlsl_high_lane_s32(int64x2_t a, int32x4_t b, int32x2_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	SMLSL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
uint32x4_t vmlsl_high_lane_u16(uint32x4_t a, uint16x8_t b, uint16x4_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	UMLSL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
uint64x2_t vmlsl_high_lane_u32(uint64x2_t a, uint32x4_t b, uint32x2_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	UMLSL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t vmlsl_laneq_s16(int32x4_t a, int16x4_t b, int16x8_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SMLSL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vmlsl_laneq_s32(int64x2_t a, int32x2_t b, int32x4_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SMLSL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	A64
uint32x4_t vmlsl_laneq_u16(uint32x4_t a, uint16x4_t b, uint16x8_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	UMLSL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A64
uint64x2_t vmlsl_laneq_u32(uint64x2_t a, uint32x2_t b, uint32x4_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	UMLSL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	A64
int32x4_t vmlsl_high_laneq_s16(int32x4_t a, int16x8_t b, int16x8_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	SMLSL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vmlsl_high_laneq_s32(int64x2_t a, int32x4_t b, int32x4_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	SMLSL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
uint32x4_t vmlsl_high_laneq_u16(uint32x4_t a, uint16x8_t b, uint16x8_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	UMLSL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
uint64x2_t vmlsl_high_laneq_u32(uint64x2_t a, uint32x4_t b, uint32x4_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	UMLSL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
int32x4_t vqdmmlsl_lane_s16(int32x4_t a, int16x4_t b, int16x4_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	SQDMLSL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	v7/A32/A64
int64x2_t vqdmmlsl_lane_s32(int64x2_t a, int32x2_t b, int32x2_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	SQDMLSL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	v7/A32/A64
int32_t vqdmmlslh_lane_s16(int32_t a, int16_t b, int16x4_t v, const int lane)	a -> Sd b -> Hn v -> Vm.4H 0 <= lane <= 3	SQDMLSL Sd,Hn,Vm.H[lane]	Sd -> result	A64
int64_t vqdmmlsls_lane_s32(int64_t a, int32_t b, int32x2_t v, const int lane)	a -> Dd b -> Sn v -> Vm.2S 0 <= lane <= 1	SQDMLSL Dd,Sn,Vm.S[lane]	Dd -> result	A64
int32x4_t vqdmmlsl_high_lane_s16(int32x4_t a, int16x8_t b, int16x4_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	SQDMLSL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vqdmmlsl_high_lane_s32(int64x2_t a, int32x4_t b, int32x2_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	SQDMLSL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t vqdmmlsl_laneq_s16(int32x4_t a, int16x4_t b, int16x8_t v, const int lane)	a -> Vd.4S b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SQDMLSL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vqdmmlsl_laneq_s32(int64x2_t a, int32x2_t b, int32x4_t v, const int lane)	a -> Vd.2D b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SQDMLSL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	A64
int32_t vqdmmlslh_laneq_s16(int32_t a, int16_t b, int16x8_t v, const int lane)	a -> Sd b -> Hn v -> Vm.8H 0 <= lane <= 7	SQDMLSL Sd,Hn,Vm.H[lane]	Sd -> result	A64
int64_t vqdmmlsls_laneq_s32(int64_t a, int32_t b, int32x4_t v, const int lane)	a -> Dd b -> Sn v -> Vm.4S 0 <= lane <= 3	SQDMLSL Dd,Sn,Vm.S[lane]	Dd -> result	A64
int32x4_t vqdmmlsl_high_laneq_s16(int32x4_t a, int16x8_t b, int16x8_t v, const int lane)	a -> Vd.4S b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	SQDMLSL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vqdmmlsl_high_laneq_s32(int64x2_t a, int32x4_t b, int32x4_t v, const int lane)	a -> Vd.2D b -> Vn.4S v -> Vm.4S 0 <= lane <= 3	SQDMLSL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
int16x4_t vmul_n_s16(int16x4_t a, int16_t b)	a -> Vn.4H b -> Vm.H[0]	MUL Vd.4H,Vn.4H,Vm.H[0]	Vd.4H -> result	v7/A32/A64
int16x8_t vmulq_n_s16(int16x8_t a, int16_t b)	a -> Vn.8H b -> Vm.H[0]	MUL Vd.8H,Vn.8H,Vm.H[0]	Vd.8H -> result	v7/A32/A64
int32x2_t vmul_n_s32(int32x2_t a, int32_t b)	a -> Vn.2S b -> Vm.S[0]	MUL Vd.2S,Vn.2S,Vm.S[0]	Vd.2S -> result	v7/A32/A64
int32x4_t vmulq_n_s32(int32x4_t a, int32_t b)	a -> Vn.4S b -> Vm.S[0]	MUL Vd.4S,Vn.4S,Vm.S[0]	Vd.4S -> result	v7/A32/A64
uint16x4_t vmul_n_u16(uint16x4_t a, uint16_t b)	a -> Vn.4H b -> Vm.H[0]	MUL Vd.4H,Vn.4H,Vm.H[0]	Vd.4H -> result	v7/A32/A64
uint16x8_t vmulq_n_u16(uint16x8_t a, uint16_t b)	a -> Vn.8H b -> Vm.H[0]	MUL Vd.8H,Vn.8H,Vm.H[0]	Vd.8H -> result	v7/A32/A64
uint32x2_t vmul_n_u32(uint32x2_t a, uint32_t b)	a -> Vn.2S b -> Vm.S[0]	MUL Vd.2S,Vn.2S,Vm.S[0]	Vd.2S -> result	v7/A32/A64
uint32x4_t vmulq_n_u32(uint32x4_t a, uint32_t b)	a -> Vn.4S b -> Vm.S[0]	MUL Vd.4S,Vn.4S,Vm.S[0]	Vd.4S -> result	v7/A32/A64
float32x2_t vmul_n_f32(float32x2_t a, float32_t b)	a -> Vn.2S b -> Vm.S[0]	FMUL Vd.2S,Vn.2S,Vm.S[0]	Vd.2S -> result	v7/A32/A64
float32x4_t vmulq_n_f32(float32x4_t a, float32_t b)	a -> Vn.4S b -> Vm.S[0]	FMUL Vd.4S,Vn.4S,Vm.S[0]	Vd.4S -> result	v7/A32/A64
float64x1_t vmul_n_f64(float64x1_t a, float64_t b)	a -> Dn b -> Vm.D[0]	FMUL Dd,Dn,Vm.D[0]	Dd -> result	A64
float64x2_t vmulq_n_f64(float64x2_t a, float64_t b)	a -> Vn.2D b -> Vm.D[0]	FMUL Vd.2D,Vn.2D,Vm.D[0]	Vd.2D -> result	A64
int16x4_t vmul_lane_s16(int16x4_t a, int16x4_t v, const int lane)	a -> Vn.4H v -> Vm.4H 0 <= lane <= 3	MUL Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	v7/A32/A64
int16x8_t vmulq_lane_s16(int16x8_t a, int16x4_t v, const int lane)	a -> Vn.8H v -> Vm.4H 0 <= lane <= 3	MUL Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	v7/A32/A64
int32x2_t vmul_lane_s32(int32x2_t a, int32x2_t v, const int lane)	a -> Vn.2S v -> Vm.2S 0 <= lane <= 1	MUL Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	v7/A32/A64
int32x4_t vmulq_lane_s32(int32x4_t a, int32x2_t v, const int lane)	a -> Vn.4S v -> Vm.2S 0 <= lane <= 1	MUL Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	v7/A32/A64
uint16x4_t vmul_lane_u16(uint16x4_t a, uint16x4_t v, const int lane)	a -> Vn.4H v -> Vm.4H 0 <= lane <= 3	MUL Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t vmulq_lane_u16(uint16x8_t a, uint16x4_t v, const int lane)	a -> Vn.8H v -> Vm.4H 0 <= lane <= 3	MUL Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	v7/A32/A64
uint32x2_t vmul_lane_u32(uint32x2_t a, uint32x2_t v, const int lane)	a -> Vn.2S v -> Vm.2S 0 <= lane <= 1	MUL Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	v7/A32/A64
uint32x4_t vmulq_lane_u32(uint32x4_t a, uint32x2_t v, const int lane)	a -> Vn.4S v -> Vm.2S 0 <= lane <= 1	MUL Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	v7/A32/A64
float32x2_t vmul_lane_f32(float32x2_t a, float32x2_t v, const int lane)	a -> Vn.2S v -> Vm.2S 0 <= lane <= 1	FMUL Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	v7/A32/A64
float32x4_t vmulq_lane_f32(float32x4_t a, float32x2_t v, const int lane)	a -> Vn.4S v -> Vm.2S 0 <= lane <= 1	FMUL Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	v7/A32/A64
float64x1_t vmul_lane_f64(float64x1_t a, float64x1_t v, const int lane)	a -> Dn v -> Vm.1D lane == 0	FMUL Dd,Dn,Vm.D[lane]	Dd -> result	A64
float64x2_t vmulq_lane_f64(float64x2_t a, float64x1_t v, const int lane)	a -> Vn.2D v -> Vm.1D lane == 0	FMUL Vd.2D,Vn.2D,Vm.D[lane]	Vd.2D -> result	A64
float32_t vmuls_lane_f32(float32_t a, float32x2_t v, const int lane)	a -> Sn v -> Vm.2S 0 <= lane <= 1	FMUL Sd,Sn,Vm.S[lane]	Sd -> result	A64
float64_t vmuld_lane_f64(float64_t a, float64x1_t v, const int lane)	a -> Dn v -> Vm.1D lane == 0	FMUL Dd,Dn,Vm.S[lane]	Dd -> result	A64
int16x4_t vmul_laneq_s16(int16x4_t a, int16x8_t v, const int lane)	a -> Vn.4H v -> Vm.8H 0 <= lane <= 7	MUL Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
int16x8_t vmulq_laneq_s16(int16x8_t a, int16x8_t v, const int lane)	a -> Vn.8H v -> Vm.8H 0 <= lane <= 7	MUL Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
int32x2_t vmul_laneq_s32(int32x2_t a, int32x4_t v, const int lane)	a -> Vn.2S v -> Vm.4S 0 <= lane <= 3	MUL Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
int32x4_t vmulq_laneq_s32(int32x4_t a, int32x4_t v, const int lane)	a -> Vn.4S v -> Vm.4S 0 <= lane <= 3	MUL Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
uint16x4_t vmul_laneq_u16(uint16x4_t a, uint16x8_t v, const int lane)	a -> Vn.4H v -> Vm.8H 0 <= lane <= 7	MUL Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
uint16x8_t vmulq_laneq_u16(uint16x8_t a, uint16x8_t v, const int lane)	a -> Vn.8H v -> Vm.8H 0 <= lane <= 7	MUL Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
uint32x2_t vmul_laneq_u32(uint32x2_t a, uint32x4_t v, const int lane)	a -> Vn.2S v -> Vm.4S 0 <= lane <= 3	MUL Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
uint32x4_t vmulq_laneq_u32(uint32x4_t a, uint32x4_t v, const int lane)	a -> Vn.4S v -> Vm.4S 0 <= lane <= 3	MUL Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
float32x2_t vmulq_laneq_f32(float32x2_t a, float32x4_t v, const int lane)	a -> Vn.2S v -> Vm.4S 0 <= lane <= 3	FMUL Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
float32x4_t vmulq_laneq_f32(float32x4_t a, float32x4_t v, const int lane)	a -> Vn.4S v -> Vm.4S 0 <= lane <= 3	FMUL Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64x1_t vmul_laneq_f64(float64x1_t a, float64x2_t v, const int lane)	a -> Dn v -> Vm.2D 0 <= lane <= 1	FMUL Dd,Dn,Vm.D[lane]	Dd -> result	A64
float64x2_t vmulq_laneq_f64(float64x2_t a, float64x2_t v, const int lane)	a -> Vn.2D v -> Vm.2D 0 <= lane <= 1	FMUL Vd.2D,Vn.2D,Vm.D[lane]	Vd.2D -> result	A64
float32_t vmuls_laneq_f32(float32_t a, float32x4_t v, const int lane)	a -> Sn v -> Vm.4S 0 <= lane <= 3	FMUL Sd,Sn,Vm.S[lane]	Sd -> result	A64
float64_t vmuld_laneq_f64(float64_t a, float64x2_t v, const int lane)	a -> Dn v -> Vm.2D 0 <= lane <= 1	FMUL Dd,Dn,Vm.D[lane]	Dd -> result	A64
int32x4_t vmull_n_s16(int16x4_t a, int16_t b)	a -> Vn.4H b -> Vm.H[0]	SMULL Vd.4S,Vn.4H,Vm.H[0]	Vd.4S -> result	v7/A32/A64
int64x2_t vmull_n_s32(int32x2_t a, int32_t b)	a -> Vn.2S b -> Vm.S[0]	SMULL Vd.2D,Vn.2S,Vm.S[0]	Vd.2D -> result	v7/A32/A64
uint32x4_t vmull_n_u16(uint16x4_t a, uint16_t b)	a -> Vn.4H b -> Vm.H[0]	UMULL Vd.4S,Vn.4H,Vm.H[0]	Vd.4S -> result	v7/A32/A64
uint64x2_t vmull_n_u32(uint32x2_t a, uint32_t b)	a -> Vn.2S b -> Vm.S[0]	UMULL Vd.2D,Vn.2S,Vm.S[0]	Vd.2D -> result	v7/A32/A64
int32x4_t vmull_high_n_s16(int16x8_t a, int16_t b)	a -> Vn.8H b -> Vm.H[0]	SMULL2 Vd.4S,Vn.8H,Vm.H[0]	Vd.4S -> result	A64
int64x2_t vmull_high_n_s32(int32x4_t a, int32_t b)	a -> Vn.4S b -> Vm.S[0]	SMULL2 Vd.2D,Vn.4S,Vm.S[0]	Vd.2D -> result	A64
uint32x4_t vmull_high_n_u16(uint16x8_t a, uint16_t b)	a -> Vn.8H b -> Vm.H[0]	UMULL2 Vd.4S,Vn.8H,Vm.H[0]	Vd.4S -> result	A64
uint64x2_t vmull_high_n_u32(uint32x4_t a, uint32_t b)	a -> Vn.4S b -> Vm.S[0]	UMULL2 Vd.2D,Vn.4S,Vm.S[0]	Vd.2D -> result	A64
int32x4_t vmull_lane_s16(int16x4_t a, int16x4_t v, const int lane)	a -> Vn.4H v -> Vm.4H 0 <= lane <= 3	SMULL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	v7/A32/A64
int64x2_t vmull_lane_s32(int32x2_t a, int32x2_t v, const int lane)	a -> Vn.2S v -> Vm.2S 0 <= lane <= 1	SMULL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	v7/A32/A64
uint32x4_t vmull_lane_u16(uint16x4_t a, uint16x4_t v, const int lane)	a -> Vn.4H v -> Vm.4H 0 <= lane <= 3	UMULL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	v7/A32/A64
uint64x2_t vmull_lane_u32(uint32x2_t a, uint32x2_t v, const int lane)	a -> Vn.2S v -> Vm.2S 0 <= lane <= 1	UMULL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	v7/A32/A64
int32x4_t vmull_high_lane_s16(int16x8_t a, int16x4_t v, const int lane)	a -> Vn.8H v -> Vm.4H 0 <= lane <= 3	SMULL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vmull_high_lane_s32(int32x4_t a, int32x2_t v, const int lane)	a -> Vn.4S v -> Vm.2S 0 <= lane <= 1	SMULL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
uint32x4_t vmull_high_lane_u16(uint16x8_t a, uint16x4_t v, const int lane)	a -> Vn.8H v -> Vm.4H 0 <= lane <= 3	UMULL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
uint64x2_t vmull_high_lane_u32(uint32x4_t a, uint32x2_t v, const int lane)	a -> Vn.4S v -> Vm.2S 0 <= lane <= 1	UMULL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
int32x4_t vmull_laneq_s16(int16x4_t a, int16x8_t v, const int lane)	a -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SMULL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vmull_laneq_s32(int32x2_t a, int32x4_t v, const int lane)	a -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SMULL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t vmull_laneq_u16(uint16x4_t a, uint16x8_t v, const int lane)	a -> Vn.4H v -> Vm.8H 0 <= lane <= 7	UMULL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A64
uint64x2_t vmull_laneq_u32(uint32x2_t a, uint32x4_t v, const int lane)	a -> Vn.2S v -> Vm.4S 0 <= lane <= 3	UMULL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	A64
int32x4_t vmull_high_laneq_s16(int16x8_t a, int16x8_t v, const int lane)	a -> Vn.8H v -> Vm.8H 0 <= lane <= 7	SMULL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vmull_high_laneq_s32(int32x4_t a, int32x4_t v, const int lane)	a -> Vn.4S v -> Vm.4S 0 <= lane <= 3	SMULL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
uint32x4_t vmull_high_laneq_u16(uint16x8_t a, uint16x8_t v, const int lane)	a -> Vn.8H v -> Vm.8H 0 <= lane <= 7	UMULL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
uint64x2_t vmull_high_laneq_u32(uint32x4_t a, uint32x4_t v, const int lane)	a -> Vn.4S v -> Vm.4S 0 <= lane <= 3	UMULL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
int32x4_t vqdmull_n_s16(int16x4_t a, int16_t b)	a -> Vn.4H b -> Vm.H[0]	SQDMULL Vd.4S,Vn.4H,Vm.H[0]	Vd.4S -> result	v7/A32/A64
int64x2_t vqdmull_n_s32(int32x2_t a, int32_t b)	a -> Vn.2S b -> Vm.S[0]	SQDMULL Vd.2D,Vn.2S,Vm.S[0]	Vd.2D -> result	v7/A32/A64
int32x4_t vqdmull_high_n_s16(int16x8_t a, int16_t b)	a -> Vn.8H b -> Vm.H[0]	SQDMULL2 Vd.4S,Vn.8H,Vm.H[0]	Vd.4S -> result	A64
int64x2_t vqdmull_high_n_s32(int32x4_t a, int32_t b)	a -> Vn.4S b -> Vm.S[0]	SQDMULL2 Vd.2D,Vn.4S,Vm.S[0]	Vd.2D -> result	A64
int32x4_t vqdmull_lane_s16(int16x4_t a, int16x4_t v, const int lane)	a -> Vn.4H v -> Vm.4H 0 <= lane <= 3	SQDMULL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	v7/A32/A64
int64x2_t vqdmull_lane_s32(int32x2_t a, int32x2_t v, const int lane)	a -> Vn.2S v -> Vm.2S 0 <= lane <= 1	SQDMULL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	v7/A32/A64
int32_t vqdmullh_lane_s16(int16_t a, int16x4_t v, const int lane)	a -> Hn v -> Vm.4H 0 <= lane <= 3	SQDMULL Sd,Hn,Vm.H[lane]	Sd -> result	A64
int64_t vqdmulls_lane_s32(int32_t a, int32x2_t v, const int lane)	a -> Sn v -> Vm.2S 0 <= lane <= 1	SQDMULL Dd,Sn,Vm.S[lane]	Dd -> result	A64
int32x4_t vqdmull_high_lane_s16(int16x8_t a, int16x4_t v, const int lane)	a -> Vn.8H v -> Vm.4H 0 <= lane <= 3	SQDMULL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vqdmull_high_lane_s32(int32x4_t a, int32x2_t v, const int lane)	a -> Vn.4S v -> Vm.2S 0 <= lane <= 1	SQDMULL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
int32x4_t vqdmull_laneq_s16(int16x4_t a, int16x8_t v, const int lane)	a -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SQDMULL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vqdmull_laneq_s32(int32x2_t a, int32x4_t v, const int lane)	a -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SQDMULL Vd.2D,Vn.2S,Vm.S[lane]	Vd.2D -> result	A64
int32_t vqdmullh_laneq_s16(int16_t a, int16x8_t v, const int lane)	a -> Hn v -> Vm.8H 0 <= lane <= 7	SQDMULL Sd,Hn,Vm.H[lane]	Sd -> result	A64
int64_t vqdmulls_laneq_s32(int32_t a, int32x4_t v, const int lane)	a -> Sn v -> Vm.4S 0 <= lane <= 3	SQDMULL Dd,Sn,Vm.S[lane]	Dd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t vqdmull_high_laneq_s16(int16x8_t a, int16x8_t v, const int lane)	a -> Vn.8H v -> Vm.8H 0 <= lane <= 7	SQDMULL2 Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A64
int64x2_t vqdmull_high_laneq_s32(int32x4_t a, int32x4_t v, const int lane)	a -> Vn.4S v -> Vm.4S 0 <= lane <= 3	SQDMULL2 Vd.2D,Vn.4S,Vm.S[lane]	Vd.2D -> result	A64
int16x4_t vqdmulh_n_s16(int16x4_t a, int16_t b)	a -> Vn.4H b -> Vm.H[0]	SQDMULH Vd.4H,Vn.4H,Vm.H[0]	Vd.4H -> result	v7/A32/A64
int16x8_t vqdmulh_n_s16(int16x8_t a, int16_t b)	a -> Vn.8H b -> Vm.H[0]	SQDMULH Vd.8H,Vn.8H,Vm.H[0]	Vd.8H -> result	v7/A32/A64
int32x2_t vqdmulh_n_s32(int32x2_t a, int32_t b)	a -> Vn.2S b -> Vm.S[0]	SQDMULH Vd.2S,Vn.2S,Vm.S[0]	Vd.2S -> result	v7/A32/A64
int32x4_t vqdmulh_n_s32(int32x4_t a, int32_t b)	a -> Vn.4S b -> Vm.S[0]	SQDMULH Vd.4S,Vn.4S,Vm.S[0]	Vd.4S -> result	v7/A32/A64
int16x4_t vqdmulh_lane_s16(int16x4_t a, int16x4_t v, const int lane)	a -> Vn.4H v -> Vm.4H 0 <= lane <= 3	SQDMULH Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	v7/A32/A64
int16x8_t vqdmulh_lane_s16(int16x8_t a, int16x4_t v, const int lane)	a -> Vn.8H v -> Vm.4H 0 <= lane <= 3	SQDMULH Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	v7/A32/A64
int32x2_t vqdmulh_lane_s32(int32x2_t a, int32x2_t v, const int lane)	a -> Vn.2S v -> Vm.2S 0 <= lane <= 1	SQDMULH Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	v7/A32/A64
int32x4_t vqdmulh_lane_s32(int32x4_t a, int32x2_t v, const int lane)	a -> Vn.4S v -> Vm.2S 0 <= lane <= 1	SQDMULH Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	v7/A32/A64
int16_t vqdmulhh_lane_s16(int16_t a, int16x4_t v, const int lane)	a -> Hn v -> Vm.4H 0 <= lane <= 3	SQDMULH Hd,Hn,Vm.H[lane]	Hd -> result	A64
int32_t vqdmulhs_lane_s32(int32_t a, int32x2_t v, const int lane)	a -> Sn v -> Vm.2S 0 <= lane <= 1	SQDMULH Sd,Sn,Vm.H[lane]	Sd -> result	A64
int16x4_t vqdmulh_laneq_s16(int16x4_t a, int16x8_t v, const int lane)	a -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SQDMULH Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
int16x8_t vqdmulh_laneq_s16(int16x8_t a, int16x8_t v, const int lane)	a -> Vn.8H v -> Vm.8H 0 <= lane <= 7	SQDMULH Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
int32x2_t vqdmulh_laneq_s32(int32x2_t a, int32x4_t v, const int lane)	a -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SQDMULH Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
int32x4_t vqdmulh_laneq_s32(int32x4_t a, int32x4_t v, const int lane)	a -> Vn.4S v -> Vm.4S 0 <= lane <= 3	SQDMULH Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
int16_t vqdmulhh_laneq_s16(int16_t a, int16x8_t v, const int lane)	a -> Hn v -> Vm.8H 0 <= lane <= 7	SQDMULH Hd,Hn,Vm.H[lane]	Hd -> result	A64
int32_t vqdmulhs_laneq_s32(int32_t a, int32x4_t v, const int lane)	a -> Sn v -> Vm.4S 0 <= lane <= 3	SQDMULH Sd,Sn,Vm.H[lane]	Sd -> result	A64
int16x4_t vqrdmulh_n_s16(int16x4_t a, int16_t b)	a -> Vn.4H b -> Vm.H[0]	SQRDMULH Vd.4H,Vn.4H,Vm.H[0]	Vd.4H -> result	v7/A32/A64
int16x8_t vqrdmulh_n_s16(int16x8_t a, int16_t b)	a -> Vn.8H b -> Vm.H[0]	SQRDMULH Vd.8H,Vn.8H,Vm.H[0]	Vd.8H -> result	v7/A32/A64
int32x2_t vqrdmulh_n_s32(int32x2_t a, int32_t b)	a -> Vn.2S b -> Vm.S[0]	SQRDMULH Vd.2S,Vn.2S,Vm.S[0]	Vd.2S -> result	v7/A32/A64
int32x4_t vqrdmulh_n_s32(int32x4_t a, int32_t b)	a -> Vn.4S b -> Vm.S[0]	SQRDMULH Vd.4S,Vn.4S,Vm.S[0]	Vd.4S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x4_t vqrdmulh_lane_s16(int16x4_t a, int16x4_t v, const int lane)	a -> Vn.4H v -> Vm.4H 0 <= lane <= 3	SQRDMULH Vd.4H, Vn.4H, Vm.H[lane]	Vd.4H -> result	v7/A32/A64
int16x8_t vqrdmulhq_lane_s16(int16x8_t a, int16x4_t v, const int lane)	a -> Vn.8H v -> Vm.4H 0 <= lane <= 3	SQRDMULH Vd.8H, Vn.8H, Vm.H[lane]	Vd.8H -> result	v7/A32/A64
int32x2_t vqrdmulh_lane_s32(int32x2_t a, int32x2_t v, const int lane)	a -> Vn.2S v -> Vm.2S 0 <= lane <= 1	SQRDMULH Vd.2S, Vn.2S, Vm.S[lane]	Vd.2S -> result	v7/A32/A64
int32x4_t vqrdmulhq_lane_s32(int32x4_t a, int32x2_t v, const int lane)	a -> Vn.4S v -> Vm.2S 0 <= lane <= 1	SQRDMULH Vd.4S, Vn.4S, Vm.S[lane]	Vd.4S -> result	v7/A32/A64
int16_t vqrdmulhh_lane_s16(int16_t a, int16x4_t v, const int lane)	a -> Hn v -> Vm.4H 0 <= lane <= 3	SQRDMULH Hd, Hn, Vm.H[lane]	Hd -> result	A64
int32_t vqrdmulhs_lane_s32(int32_t a, int32x2_t v, const int lane)	a -> Sn v -> Vm.2S 0 <= lane <= 1	SQRDMULH Sd, Sn, Vm.S[lane]	Sd -> result	A64
int16x4_t vqrdmulh_laneq_s16(int16x4_t a, int16x8_t v, const int lane)	a -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SQRDMULH Vd.4H, Vn.4H, Vm.H[lane]	Vd.4H -> result	A64
int16x8_t vqrdmulhq_laneq_s16(int16x8_t a, int16x8_t v, const int lane)	a -> Vn.8H v -> Vm.8H 0 <= lane <= 7	SQRDMULH Vd.8H, Vn.8H, Vm.H[lane]	Vd.8H -> result	A64
int32x2_t vqrdmulh_laneq_s32(int32x2_t a, int32x4_t v, const int lane)	a -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SQRDMULH Vd.2S, Vn.2S, Vm.S[lane]	Vd.2S -> result	A64
int32x4_t vqrdmulhq_laneq_s32(int32x4_t a, int32x4_t v, const int lane)	a -> Vn.4S v -> Vm.4S 0 <= lane <= 3	SQRDMULH Vd.4S, Vn.4S, Vm.S[lane]	Vd.4S -> result	A64
int16_t vqrdmulhh_laneq_s16(int16_t a, int16x8_t v, const int lane)	a -> Hn v -> Vm.8H 0 <= lane <= 7	SQRDMULH Hd, Hn, Vm.H[lane]	Hd -> result	A64
int32_t vqrdmulhs_laneq_s32(int32_t a, int32x4_t v, const int lane)	a -> Sn v -> Vm.4S 0 <= lane <= 3	SQRDMULH Sd, Sn, Vm.S[lane]	Sd -> result	A64
int16x4_t vmla_n_s16(int16x4_t a, int16x4_t b, int16_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.H[0]	MLA Vd.4H, Vn.4H, Vm.H[0]	Vd.4H -> result	v7/A32/A64
int16x8_t vmlaq_n_s16(int16x8_t a, int16x8_t b, int16_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.H[0]	MLA Vd.8H, Vn.8H, Vm.H[0]	Vd.8H -> result	v7/A32/A64
int32x2_t vmla_n_s32(int32x2_t a, int32x2_t b, int32_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.S[0]	MLA Vd.2S, Vn.2S, Vm.S[0]	Vd.2S -> result	v7/A32/A64
int32x4_t vmlaq_n_s32(int32x4_t a, int32x4_t b, int32_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.S[0]	MLA Vd.4S, Vn.4S, Vm.S[0]	Vd.4S -> result	v7/A32/A64
uint16x4_t vmla_n_u16(uint16x4_t a, uint16x4_t b, uint16_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.H[0]	MLA Vd.4H, Vn.4H, Vm.H[0]	Vd.4H -> result	v7/A32/A64
uint16x8_t vmlaq_n_u16(uint16x8_t a, uint16x8_t b, uint16_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.H[0]	MLA Vd.8H, Vn.8H, Vm.H[0]	Vd.8H -> result	v7/A32/A64
uint32x2_t vmla_n_u32(uint32x2_t a, uint32x2_t b, uint32_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.S[0]	MLA Vd.2S, Vn.2S, Vm.S[0]	Vd.2S -> result	v7/A32/A64
uint32x4_t vmlaq_n_u32(uint32x4_t a, uint32x4_t b, uint32_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.S[0]	MLA Vd.4S, Vn.4S, Vm.S[0]	Vd.4S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x2_t vmla_n_f32(float32x2_t a, float32x2_t b, float32_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] + (b[i] * c) for i = 0 to 1	N/A -> result	v7/A32/A64
float32x4_t vmlaq_n_f32(float32x4_t a, float32x4_t b, float32_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] + (b[i] * c) for i = 0 to 3	N/A -> result	v7/A32/A64
int32x4_t vmlal_n_s16(int32x4_t a, int16x4_t b, int16_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.H[0]	SMLAL Vd.4S,Vn.4H,Vm.H[0]	Vd.4S -> result	v7/A32/A64
int64x2_t vmlal_n_s32(int64x2_t a, int32x2_t b, int32_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.S[0]	SMLAL Vd.2D,Vn.2S,Vm.S[0]	Vd.2D -> result	v7/A32/A64
uint32x4_t vmlal_n_u16(uint32x4_t a, uint16x4_t b, uint16_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.H[0]	UMLAL Vd.4S,Vn.4H,Vm.H[0]	Vd.4S -> result	v7/A32/A64
uint64x2_t vmlal_n_u32(uint64x2_t a, uint32x2_t b, uint32_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.S[0]	UMLAL Vd.2D,Vn.2S,Vm.S[0]	Vd.2D -> result	v7/A32/A64
int32x4_t vmlal_high_n_s16(int32x4_t a, int16x8_t b, int16_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.H[0]	SMLAL2 Vd.4S,Vn.8H,Vm.H[0]	Vd.4S -> result	A64
int64x2_t vmlal_high_n_s32(int64x2_t a, int32x4_t b, int32_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.S[0]	SMLAL2 Vd.2D,Vn.4S,Vm.S[0]	Vd.2D -> result	A64
uint32x4_t vmlal_high_n_u16(uint32x4_t a, uint16x8_t b, uint16_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.H[0]	UMLAL2 Vd.4S,Vn.8H,Vm.H[0]	Vd.4S -> result	A64
uint64x2_t vmlal_high_n_u32(uint64x2_t a, uint32x4_t b, uint32_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.S[0]	UMLAL2 Vd.2D,Vn.4S,Vm.S[0]	Vd.2D -> result	A64
int32x4_t vqdmmlal_n_s16(int32x4_t a, int16x4_t b, int16_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.H[0]	SQDMLAL Vd.4S,Vn.4H,Vm.H[0]	Vd.4S -> result	v7/A32/A64
int64x2_t vqdmmlal_n_s32(int64x2_t a, int32x2_t b, int32_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.S[0]	SQDMLAL Vd.2D,Vn.2S,Vm.S[0]	Vd.2D -> result	v7/A32/A64
int32x4_t vqdmmlal_high_n_s16(int32x4_t a, int16x8_t b, int16_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.H[0]	SQDMLAL2 Vd.4S,Vn.8H,Vm.H[0]	Vd.4S -> result	A64
int64x2_t vqdmmlal_high_n_s32(int64x2_t a, int32x4_t b, int32_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.S[0]	SQDMLAL2 Vd.2D,Vn.4S,Vm.S[0]	Vd.2D -> result	A64
int16x4_t vmls_n_s16(int16x4_t a, int16x4_t b, int16_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.H[0]	MLS Vd.4H,Vn.4H,Vm.H[0]	Vd.4H -> result	v7/A32/A64
int16x8_t vmlsq_n_s16(int16x8_t a, int16x8_t b, int16_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.H[0]	MLS Vd.8H,Vn.8H,Vm.H[0]	Vd.8H -> result	v7/A32/A64
int32x2_t vmls_n_s32(int32x2_t a, int32x2_t b, int32_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.S[0]	MLS Vd.2S,Vn.2S,Vm.S[0]	Vd.2S -> result	v7/A32/A64
int32x4_t vmlsq_n_s32(int32x4_t a, int32x4_t b, int32_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.S[0]	MLS Vd.4S,Vn.4S,Vm.S[0]	Vd.4S -> result	v7/A32/A64
uint16x4_t vmls_n_u16(uint16x4_t a, uint16x4_t b, uint16_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.H[0]	MLS Vd.4H,Vn.4H,Vm.H[0]	Vd.4H -> result	v7/A32/A64
uint16x8_t vmlsq_n_u16(uint16x8_t a, uint16x8_t b, uint16_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.H[0]	MLS Vd.8H,Vn.8H,Vm.H[0]	Vd.8H -> result	v7/A32/A64
uint32x2_t vmls_n_u32(uint32x2_t a, uint32x2_t b, uint32_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.S[0]	MLS Vd.2S,Vn.2S,Vm.S[0]	Vd.2S -> result	v7/A32/A64
uint32x4_t vmlsq_n_u32(uint32x4_t a, uint32x4_t b, uint32_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.S[0]	MLS Vd.4S,Vn.4S,Vm.S[0]	Vd.4S -> result	v7/A32/A64
float32x2_t vmls_n_f32(float32x2_t a, float32x2_t b, float32_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] - (b[i] * c) for i = 0 to 1	N/A -> result	v7/A32/A64
float32x4_t vmlsq_n_f32(float32x4_t a, float32x4_t b, float32_t c)	a -> N/A b -> N/A c -> N/A	RESULT[I] = a[i] - (b[i] * c) for i = 0 to 3	N/A -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t vmlsl_n_s16(int32x4_t a, int16x4_t b, int16_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.H[0]	SMLSL Vd.4S,Vn.4H,Vm.H[0]	Vd.4S -> result	v7/A32/A64
int64x2_t vmlsl_n_s32(int64x2_t a, int32x2_t b, int32_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.S[0]	SMLSL Vd.2D,Vn.2S,Vm.S[0]	Vd.2D -> result	v7/A32/A64
uint32x4_t vmlsl_n_u16(uint32x4_t a, uint16x4_t b, uint16_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.H[0]	UMLSL Vd.4S,Vn.4H,Vm.H[0]	Vd.4S -> result	v7/A32/A64
uint64x2_t vmlsl_n_u32(uint64x2_t a, uint32x2_t b, uint32_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.S[0]	UMLSL Vd.2D,Vn.2S,Vm.S[0]	Vd.2D -> result	v7/A32/A64
int32x4_t vmlsl_high_n_s16(int32x4_t a, int16x8_t b, int16_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.H[0]	SMLSL2 Vd.4S,Vn.8H,Vm.H[0]	Vd.4S -> result	A64
int64x2_t vmlsl_high_n_s32(int64x2_t a, int32x4_t b, int32_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.S[0]	SMLSL2 Vd.2D,Vn.4S,Vm.S[0]	Vd.2D -> result	A64
uint32x4_t vmlsl_high_n_u16(uint32x4_t a, uint16x8_t b, uint16_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.H[0]	UMLSL2 Vd.4S,Vn.8H,Vm.H[0]	Vd.4S -> result	A64
uint64x2_t vmlsl_high_n_u32(uint64x2_t a, uint32x4_t b, uint32_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.S[0]	UMLSL2 Vd.2D,Vn.4S,Vm.S[0]	Vd.2D -> result	A64
int32x4_t vqdmmlsl_n_s16(int32x4_t a, int16x4_t b, int16_t c)	a -> Vd.4S b -> Vn.4H c -> Vm.H[0]	SQDMLSL Vd.4S,Vn.4H,Vm.H[0]	Vd.4S -> result	v7/A32/A64
int64x2_t vqdmmlsl_n_s32(int64x2_t a, int32x2_t b, int32_t c)	a -> Vd.2D b -> Vn.2S c -> Vm.S[0]	SQDMLSL Vd.2D,Vn.2S,Vm.S[0]	Vd.2D -> result	v7/A32/A64
int32x4_t vqdmmlsl_high_n_s16(int32x4_t a, int16x8_t b, int16_t c)	a -> Vd.4S b -> Vn.8H c -> Vm.H[0]	SQDMLSL2 Vd.4S,Vn.8H,Vm.H[0]	Vd.4S -> result	A64
int64x2_t vqdmmlsl_high_n_s32(int64x2_t a, int32x4_t b, int32_t c)	a -> Vd.2D b -> Vn.4S c -> Vm.S[0]	SQDMLSL2 Vd.2D,Vn.4S,Vm.S[0]	Vd.2D -> result	A64
int8x8_t vabs_s8(int8x8_t a)	a -> Vn.8B	ABS Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vabsq_s8(int8x16_t a)	a -> Vn.16B	ABS Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vabs_s16(int16x4_t a)	a -> Vn.4H	ABS Vd.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vabsq_s16(int16x8_t a)	a -> Vn.8H	ABS Vd.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vabs_s32(int32x2_t a)	a -> Vn.2S	ABS Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vabsq_s32(int32x4_t a)	a -> Vn.4S	ABS Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
float32x2_t vabs_f32(float32x2_t a)	a -> Vn.2S	FABS Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vabsq_f32(float32x4_t a)	a -> Vn.4S	FABS Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
int64x1_t vabs_s64(int64x1_t a)	a -> Dn	ABS Dd,Dn	Dd -> result	A64
int64_t vabsd_s64(int64_t a)	a -> Dn	ABS Dd,Dn	Dd -> result	A64
int64x2_t vabsq_s64(int64x2_t a)	a -> Vn.2D	ABS Vd.2D,Vn.2D	Vd.2D -> result	A64
float64x1_t vabs_f64(float64x1_t a)	a -> Dn	FABS Dd,Dn	Dd -> result	A64
float64x2_t vabsq_f64(float64x2_t a)	a -> Vn.2D	FABS Vd.2D,Vn.2D	Vd.2D -> result	A64
int8x8_t vqabs_s8(int8x8_t a)	a -> Vn.8B	SQABS Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vqabsq_s8(int8x16_t a)	a -> Vn.16B	SQABS Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vqabs_s16(int16x4_t a)	a -> Vn.4H	SQABS Vd.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vqabsq_s16(int16x8_t a)	a -> Vn.8H	SQABS Vd.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vqabs_s32(int32x2_t a)	a -> Vn.2S	SQABS Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vqabsq_s32(int32x4_t a)	a -> Vn.4S	SQABS Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
int64x1_t vqabs_s64(int64x1_t a)	a -> Dn	SQABS Dd,Dn	Dd -> result	A64
int64x2_t vqabsq_s64(int64x2_t a)	a -> Vn.2D	SQABS Vd.2D,Vn.2D	Vd.2D -> result	A64
int8_t vqabsb_s8(int8_t a)	a -> Bn	SQABS Bd,Bn	Bd -> result	A64
int16_t vqabsh_s16(int16_t a)	a -> Hn	SQABS Hd,Hn	Hd -> result	A64
int32_t vqabss_s32(int32_t a)	a -> Sn	SQABS Sd,Sn	Sd -> result	A64
int64_t vqabsd_s64(int64_t a)	a -> Dn	SQABS Dd,Dn	Dd -> result	A64
int8x8_t vneg_s8(int8x8_t a)	a -> Vn.8B	NEG Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vnegq_s8(int8x16_t a)	a -> Vn.16B	NEG Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vneg_s16(int16x4_t a)	a -> Vn.4H	NEG Vd.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vnegq_s16(int16x8_t a)	a -> Vn.8H	NEG Vd.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vneg_s32(int32x2_t a)	a -> Vn.2S	NEG Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vnegq_s32(int32x4_t a)	a -> Vn.4S	NEG Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
float32x2_t vneg_f32(float32x2_t a)	a -> Vn.2S	FNEG Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vnegq_f32(float32x4_t a)	a -> Vn.4S	FNEG Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64x1_tvneg_s64(int64x1_t a)	a -> Dn	NEG Dd,Dn	Dd -> result	A64
int64_tvnegd_s64(int64_t a)	a -> Dn	NEG Dd,Dn	Dd -> result	A64
int64x2_tvnegq_s64(int64x2_t a)	a -> Vn.2D	NEG Vd.2D,Vn.2D	Vd.2D -> result	A64
float64x1_tvneg_f64(float64x1_t a)	a -> Dn	FNEG Dd,Dn	Dd -> result	A64
float64x2_tvnegq_f64(float64x2_t a)	a -> Vn.2D	FNEG Vd.2D,Vn.2D	Vd.2D -> result	A64
int8x8_tvqneg_s8(int8x8_t a)	a -> Vn.8B	SQNEG Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_tvqnegq_s8(int8x16_t a)	a -> Vn.16B	SQNEG Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
int16x4_tvqneg_s16(int16x4_t a)	a -> Vn.4H	SQNEG Vd.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
int16x8_tvqnegq_s16(int16x8_t a)	a -> Vn.8H	SQNEG Vd.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
int32x2_tvqneg_s32(int32x2_t a)	a -> Vn.2S	SQNEG Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
int32x4_tvqnegq_s32(int32x4_t a)	a -> Vn.4S	SQNEG Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
int64x1_tvqneg_s64(int64x1_t a)	a -> Dn	SQNEG Dd,Dn	Dd -> result	A64
int64x2_tvqnegq_s64(int64x2_t a)	a -> Vn.2D	SQNEG Vd.2D,Vn.2D	Vd.2D -> result	A64
int8_tvqnegb_s8(int8_t a)	a -> Bn	SQNEG Bd,Bn	Bd -> result	A64
int16_tvqnegh_s16(int16_t a)	a -> Hn	SQNEG Hd,Hn	Hd -> result	A64
int32_tvqnegs_s32(int32_t a)	a -> Sn	SQNEG Sd,Sn	Sd -> result	A64
int64_tvqnegd_s64(int64_t a)	a -> Dn	SQNEG Dd,Dn	Dd -> result	A64
int8x8_tvcls_s8(int8x8_t a)	a -> Vn.8B	CLS Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_tvclsq_s8(int8x16_t a)	a -> Vn.16B	CLS Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
int16x4_tvcls_s16(int16x4_t a)	a -> Vn.4H	CLS Vd.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
int16x8_tvclsq_s16(int16x8_t a)	a -> Vn.8H	CLS Vd.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
int32x2_tvcls_s32(int32x2_t a)	a -> Vn.2S	CLS Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
int32x4_tvclsq_s32(int32x4_t a)	a -> Vn.4S	CLS Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
int8x8_tvcls_u8(int8x8_t a)	a -> Vn.8B	CLS Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_tvclsq_u8(int8x16_t a)	a -> Vn.16B	CLS Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
int16x4_tvcls_u16(int16x4_t a)	a -> Vn.4H	CLS Vd.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
int16x8_tvclsq_u16(int16x8_t a)	a -> Vn.8H	CLS Vd.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
int32x2_tvcls_u32(int32x2_t a)	a -> Vn.2S	CLS Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
int32x4_tvclsq_u32(int32x4_t a)	a -> Vn.4S	CLS Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
int8x8_tvclz_s8(int8x8_t a)	a -> Vn.8B	CLZ Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_tvclzq_s8(int8x16_t a)	a -> Vn.16B	CLZ Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
int16x4_tvclz_s16(int16x4_t a)	a -> Vn.4H	CLZ Vd.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
int16x8_tvclzq_s16(int16x8_t a)	a -> Vn.8H	CLZ Vd.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
int32x2_tvclz_s32(int32x2_t a)	a -> Vn.2S	CLZ Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
int32x4_tvclzq_s32(int32x4_t a)	a -> Vn.4S	CLZ Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
uint8x8_tvclz_u8(int8x8_t a)	a -> Vn.8B	CLZ Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_tvclzq_u8(int8x16_t a)	a -> Vn.16B	CLZ Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint16x4_tvclz_u16(int16x4_t a)	a -> Vn.4H	CLZ Vd.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
uint16x8_tvclzq_u16(int16x8_t a)	a -> Vn.8H	CLZ Vd.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
uint32x2_tvclz_u32(int32x2_t a)	a -> Vn.2S	CLZ Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_tvclzq_u32(int32x4_t a)	a -> Vn.4S	CLZ Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
int8x8_tvcnt_s8(int8x8_t a)	a -> Vn.8B	CNT Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_tvcntq_s8(int8x16_t a)	a -> Vn.16B	CNT Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint8x8_tvcnt_u8(int8x8_t a)	a -> Vn.8B	CNT Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_tvcntq_u8(int8x16_t a)	a -> Vn.16B	CNT Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
poly8x8_tvcnt_p8(poly8x8_t a)	a -> Vn.8B	CNT Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
poly8x16_tvcntq_p8(poly8x16_t a)	a -> Vn.16B	CNT Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint32x2_tvrecpe_u32(uint32x2_t a)	a -> Vn.2S	URECPE Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_tvrecpeq_u32(uint32x4_t a)	a -> Vn.4S	URECPE Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
float32x2_tvrecpe_f32(float32x2_t a)	a -> Vn.2S	FRECPE Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
float32x4_tvrecpeq_f32(float32x4_t a)	a -> Vn.4S	FRECPE Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
float64x1_tvrecpe_f64(float64x1_t a)	a -> Dn	FRECPE Dd,Dn	Dd -> result	A64
float64x2_tvrecpeq_f64(float64x2_t a)	a -> Vn.2D	FRECPE Vd.2D,Vn.2D	Vd.2D -> result	A64
float32_tvrecpes_f32(float32_t a)	a -> Sn	FRECPE Sd,Sn	Sd -> result	A64
float64_tvrecped_f64(float64_t a)	a -> Dn	FRECPE Dd,Dn	Dd -> result	A64
float32x2_tvrecps_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FRECPS Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
float32x4_tvrecpsq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FRECPS Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float64x1_tvrecps_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FRECPS Dd,Dn,Dm	Dd -> result	A64
float64x2_tvrecpsq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FRECPS Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32_t vrecps_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FRECPS Sd,Sn,Sm	Sd -> result	A64
float64_t vrecpsd_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FRECPS Dd,Dn,Dm	Dd -> result	A64
float32x2_t vsqrt_f32(float32x2_t a)	a -> Vn.2S	FSQRT Vd.2S,Vn.2S	Vd.2S -> result	A64
float32x4_t vsqrtq_f32(float32x4_t a)	a -> Vn.4S	FSQRT Vd.4S,Vn.4S	Vd.4S -> result	A64
float64x1_t vsqrt_f64(float64x1_t a)	a -> Dn	FSQRT Dd,Dn	Dd -> result	A64
float64x2_t vsqrtq_f64(float64x2_t a)	a -> Vn.2D	FSQRT Vd.2D,Vn.2D	Vd.2D -> result	A64
uint32x2_t vrsqrte_u32(uint32x2_t a)	a -> Vn.2S	URSQRTE Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vrsqrteq_u32(uint32x4_t a)	a -> Vn.4S	URSQRTE Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
float32x2_t vrsqrte_f32(float32x2_t a)	a -> Vn.2S	FRSQRTE Vd.2S,Vn.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vrsqrteq_f32(float32x4_t a)	a -> Vn.4S	FRSQRTE Vd.4S,Vn.4S	Vd.4S -> result	v7/A32/A64
float64x1_t vrsqrte_f64(float64x1_t a)	a -> Dn	FRSQRTE Dd,Dn	Dd -> result	A64
float64x2_t vrsqrteq_f64(float64x2_t a)	a -> Vn.2D	FRSQRTE Vd.2D,Vn.2D	Vd.2D -> result	A64
float32_t vrsqrtes_f32(float32_t a)	a -> Sn	FRSQRTE Sd,Sn	Sd -> result	A64
float64_t vrsqrtesd_f64(float64_t a)	a -> Dn	FRSQRTE Dd,Dn	Dd -> result	A64
float32x2_t vrsqrts_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FRSQRTS Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vrsqrtsq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FRSQRTS Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	v7/A32/A64
float64x1_t vrsqrts_f64(float64x1_t a, float64x1_t b)	a -> Dn b -> Dm	FRSQRTS Dd,Dn,Dm	Dd -> result	A64
float64x2_t vrsqrtsq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FRSQRTS Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32_t vrsqrtsd_f32(float32_t a, float32_t b)	a -> Sn b -> Sm	FRSQRTS Sd,Sn,Sm	Sd -> result	A64
float64_t vrsqrtsd_f64(float64_t a, float64_t b)	a -> Dn b -> Dm	FRSQRTS Dd,Dn,Dm	Dd -> result	A64
int8x8_t vmvn_s8(int8x8_t a)	a -> Vn.8B	MVN Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vmvnq_s8(int8x16_t a)	a -> Vn.16B	MVN Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vmvn_s16(int16x4_t a)	a -> Vn.8B	MVN Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
int16x8_t vmvnq_s16(int16x8_t a)	a -> Vn.16B	MVN Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
int32x2_t vmvn_s32(int32x2_t a)	a -> Vn.8B	MVN Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
int32x4_t vmvnq_s32(int32x4_t a)	a -> Vn.16B	MVN Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint8x8_t vmvn_u8(uint8x8_t a)	a -> Vn.8B	MVN Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vmvnq_u8(uint8x16_t a)	a -> Vn.16B	MVN Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vmvn_u16(uint16x4_t a)	a -> Vn.8B	MVN Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
uint16x8_t vmvnq_u16(uint16x8_t a)	a -> Vn.16B	MVN Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
uint32x2_t vmvn_u32(uint32x2_t a)	a -> Vn.8B	MVN Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
uint32x4_t vmvnq_u32(uint32x4_t a)	a -> Vn.16B	MVN Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
poly8x8_t vmvn_p8(poly8x8_t a)	a -> Vn.8B	MVN Vd.8B,Vn.8B	Vd.8B -> result	v7/A32/A64
poly8x16_t vmvnq_p8(poly8x16_t a)	a -> Vn.16B	MVN Vd.16B,Vn.16B	Vd.16B -> result	v7/A32/A64
int8x8_t vand_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	AND Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vandq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	AND Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vand_s16(int16x4_t a, int16x4_t b)	a -> Vn.8B b -> Vm.8B	AND Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int16x8_t vandq_s16(int16x8_t a, int16x8_t b)	a -> Vn.16B b -> Vm.16B	AND Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int32x2_t vand_s32(int32x2_t a, int32x2_t b)	a -> Vn.8B b -> Vm.8B	AND Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int32x4_t vandq_s32(int32x4_t a, int32x4_t b)	a -> Vn.16B b -> Vm.16B	AND Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int64x1_t vand_s64(int64x1_t a, int64x1_t b)	a -> Dn b -> Dm	AND Dd,Dn,Dm	Dd -> result	v7/A32/A64
int64x2_t vandq_s64(int64x2_t a, int64x2_t b)	a -> Vn.16B b -> Vm.16B	AND Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint8x8_t vand_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	AND Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vandq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	AND Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vand_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.8B b -> Vm.8B	AND Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t vandq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.16B b -> Vm.16B	AND Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint32x2_t vand_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.8B b -> Vm.8B	AND Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint32x4_t vandq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.16B b -> Vm.16B	AND Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint64x1_t vand_u64(uint64x1_t a, uint64x1_t b)	a -> Vn.8B b -> Vm.8B	AND Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint64x2_t vandq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.16B b -> Vm.16B	AND Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int8x8_t vorr_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	ORR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vorrq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	ORR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vorr_s16(int16x4_t a, int16x4_t b)	a -> Vn.8B b -> Vm.8B	ORR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int16x8_t vorrq_s16(int16x8_t a, int16x8_t b)	a -> Vn.16B b -> Vm.16B	ORR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int32x2_t vorr_s32(int32x2_t a, int32x2_t b)	a -> Vn.8B b -> Vm.8B	ORR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int32x4_t vorrq_s32(int32x4_t a, int32x4_t b)	a -> Vn.16B b -> Vm.16B	ORR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int64x1_t vorr_s64(int64x1_t a, int64x1_t b)	a -> Vn.8B b -> Vm.8B	ORR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int64x2_t vorrq_s64(int64x2_t a, int64x2_t b)	a -> Vn.16B b -> Vm.16B	ORR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint8x8_t vorr_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	ORR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vorrq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	ORR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vorr_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.8B b -> Vm.8B	ORR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint16x8_t vorrq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.16B b -> Vm.16B	ORR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint32x2_t vorr_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.8B b -> Vm.8B	ORR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint32x4_t vorrq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.16B b -> Vm.16B	ORR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint64x1_t vorr_u64(uint64x1_t a, uint64x1_t b)	a -> Vn.8B b -> Vm.8B	ORR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint64x2_t vorrq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.16B b -> Vm.16B	ORR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int8x8_t veor_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	EOR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t veorq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t veor_s16(int16x4_t a, int16x4_t b)	a -> Vn.8B b -> Vm.8B	EOR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int16x8_t veorq_s16(int16x8_t a, int16x8_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int32x2_t veor_s32(int32x2_t a, int32x2_t b)	a -> Vn.8B b -> Vm.8B	EOR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int32x4_t veorq_s32(int32x4_t a, int32x4_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int64x1_t veor_s64(int64x1_t a, int64x1_t b)	a -> Vn.8B b -> Vm.8B	EOR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int64x2_t veorq_s64(int64x2_t a, int64x2_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint8x8_t veor_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	EOR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t veorq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t veor_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.8B b -> Vm.8B	EOR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint16x8_t veorq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint32x2_t veor_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.8B b -> Vm.8B	EOR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint32x4_t veorq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint64x1_t veor_u64(uint64x1_t a, uint64x1_t b)	a -> Vn.8B b -> Vm.8B	EOR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64x2_t veorq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int8x8_t vbic_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	BIC Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vbicq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	BIC Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vbic_s16(int16x4_t a, int16x4_t b)	a -> Vn.8B b -> Vm.8B	BIC Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int16x8_t vbicq_s16(int16x8_t a, int16x8_t b)	a -> Vn.16B b -> Vm.16B	BIC Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int32x2_t vbic_s32(int32x2_t a, int32x2_t b)	a -> Vn.8B b -> Vm.8B	BIC Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int32x4_t vbicq_s32(int32x4_t a, int32x4_t b)	a -> Vn.16B b -> Vm.16B	BIC Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int64x1_t vbic_s64(int64x1_t a, int64x1_t b)	a -> Vn.8B b -> Vm.8B	BIC Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int64x2_t vbicq_s64(int64x2_t a, int64x2_t b)	a -> Vn.16B b -> Vm.16B	BIC Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint8x8_t vbic_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	BIC Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vbicq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	BIC Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vbic_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.8B b -> Vm.8B	BIC Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint16x8_t vbicq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.16B b -> Vm.16B	BIC Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint32x2_t vbic_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.8B b -> Vm.8B	BIC Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint32x4_t vbicq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.16B b -> Vm.16B	BIC Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint64x1_t vbic_u64(uint64x1_t a, uint64x1_t b)	a -> Vn.8B b -> Vm.8B	BIC Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint64x2_t vbicq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.16B b -> Vm.16B	BIC Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int8x8_t vorn_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	ORN Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vornq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	ORN Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vorn_s16(int16x4_t a, int16x4_t b)	a -> Vn.8B b -> Vm.8B	ORN Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int16x8_t vornq_s16(int16x8_t a, int16x8_t b)	a -> Vn.16B b -> Vm.16B	ORN Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int32x2_t vorn_s32(int32x2_t a, int32x2_t b)	a -> Vn.8B b -> Vm.8B	ORN Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int32x4_t vornq_s32(int32x4_t a, int32x4_t b)	a -> Vn.16B b -> Vm.16B	ORN Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int64x1_t vorn_s64(int64x1_t a, int64x1_t b)	a -> Vn.8B b -> Vm.8B	ORN Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int64x2_t vornq_s64(int64x2_t a, int64x2_t b)	a -> Vn.16B b -> Vm.16B	ORN Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint8x8_t vorn_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	ORN Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vornq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	ORN Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vorn_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.8B b -> Vm.8B	ORN Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint16x8_t vornq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.16B b -> Vm.16B	ORN Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint32x2_t vorn_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.8B b -> Vm.8B	ORN Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint32x4_t vornq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.16B b -> Vm.16B	ORN Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint64x1_t vorn_u64(uint64x1_t a, uint64x1_t b)	a -> Vn.8B b -> Vm.8B	ORN Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint64x2_t vornq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.16B b -> Vm.16B	ORN Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int8x8_t vbsl_s8(uint8x8_t a, int8x8_t b, int8x8_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vbslq_s8(uint8x16_t a, int8x16_t b, int8x16_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x4_t vbsl_s16(uint16x4_t a, int16x4_t b, int16x4_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int16x8_t vbslq_s16(uint16x8_t a, int16x8_t b, int16x8_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int32x2_t vbsl_s32(uint32x2_t a, int32x2_t b, int32x2_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int32x4_t vbslq_s32(uint32x4_t a, int32x4_t b, int32x4_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
int64x1_t vbsl_s64(uint64x1_t a, int64x1_t b, int64x1_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int64x2_t vbslq_s64(uint64x2_t a, int64x2_t b, int64x2_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint8x8_t vbsl_u8(uint8x8_t a, uint8x8_t b, uint8x8_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vbslq_u8(uint8x16_t a, uint8x16_t b, uint8x16_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vbsl_u16(uint16x4_t a, uint16x4_t b, uint16x4_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint16x8_t vbslq_u16(uint16x8_t a, uint16x8_t b, uint16x8_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint32x2_t vbsl_u32(uint32x2_t a, uint32x2_t b, uint32x2_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint32x4_t vbslq_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint64x1_t vbsl_u64(uint64x1_t a, uint64x1_t b, uint64x1_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint64x2_t vbslq_u64(uint64x2_t a, uint64x2_t b, uint64x2_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
poly64x1_t vbsl_p64(poly64x1_t a, poly64x1_t b, poly64x1_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A32/A64
poly64x2_t vbslq_p64(poly64x2_t a, poly64x2_t b, poly64x2_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A32/A64
float32x2_t vbsl_f32(uint32x2_t a, float32x2_t b, float32x2_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
float32x4_t vbslq_f32(uint32x4_t a, float32x4_t b, float32x4_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
poly8x8_t vbsl_p8(uint8x8_t a, poly8x8_t b, poly8x8_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
poly8x16_t vbslq_p8(uint8x16_t a, poly8x16_t b, poly8x16_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
poly16x4_t vbsl_p16(uint16x4_t a, poly16x4_t b, poly16x4_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
poly16x8_t vbslq_p16(uint16x8_t a, poly16x8_t b, poly16x8_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
float64x1_t vbsl_f64(uint64x1_t a, float64x1_t b, float64x1_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
float64x2_t vbslq_f64(uint64x2_t a, float64x2_t b, float64x2_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x8_t vcopy_lane_s8(int8x8_t a, const int lane1, int8x8_t b, const int lane2)	a -> Vd.8B 0 <= lane1 <= 7 b -> Vn.8B 0 <= lane2 <= 7	INS Vd.B[lane1], Vn.B[lane2]	Vd.8B -> result	A64
int8x16_t vcopyq_lane_s8(int8x16_t a, const int lane1, int8x8_t b, const int lane2)	a -> Vd.16B 0 <= lane1 <= 15 b -> Vn.8B 0 <= lane2 <= 7	INS Vd.B[lane1], Vn.B[lane2]	Vd.16B -> result	A64
int16x4_t vcopy_lane_s16(int16x4_t a, const int lane1, int16x4_t b, const int lane2)	a -> Vd.4H 0 <= lane1 <= 3 b -> Vn.4H 0 <= lane2 <= 3	INS Vd.H[lane1], Vn.H[lane2]	Vd.4H -> result	A64
int16x8_t vcopyq_lane_s16(int16x8_t a, const int lane1, int16x4_t b, const int lane2)	a -> Vd.8H 0 <= lane1 <= 7 b -> Vn.4H 0 <= lane2 <= 3	INS Vd.H[lane1], Vn.H[lane2]	Vd.8H -> result	A64
int32x2_t vcopy_lane_s32(int32x2_t a, const int lane1, int32x2_t b, const int lane2)	a -> Vd.2S 0 <= lane1 <= 1 b -> Vn.2S 0 <= lane2 <= 1	INS Vd.S[lane1], Vn.S[lane2]	Vd.2S -> result	A64
int32x4_t vcopyq_lane_s32(int32x4_t a, const int lane1, int32x2_t b, const int lane2)	a -> Vd.4S 0 <= lane1 <= 3 b -> Vn.2S 0 <= lane2 <= 1	INS Vd.S[lane1], Vn.S[lane2]	Vd.4S -> result	A64
int64x1_t vcopy_lane_s64(int64x1_t a, const int lane1, int64x1_t b, const int lane2)	a -> UNUSED lane1 == 0 b -> Vn.1D lane2 == 0	DUP Dd, Vn.D[lane2]	Dd -> result	A64
int64x2_t vcopyq_lane_s64(int64x2_t a, const int lane1, int64x1_t b, const int lane2)	a -> Vd.2D 0 <= lane1 <= 1 b -> Vn.1D lane2 == 0	INS Vd.D[lane1], Vn.D[lane2]	Vd.2D -> result	A64
uint8x8_t vcopy_lane_u8(uint8x8_t a, const int lane1, uint8x8_t b, const int lane2)	a -> Vd.8B 0 <= lane1 <= 7 b -> Vn.8B 0 <= lane2 <= 7	INS Vd.B[lane1], Vn.B[lane2]	Vd.8B -> result	A64
uint8x16_t vcopyq_lane_u8(uint8x16_t a, const int lane1, uint8x8_t b, const int lane2)	a -> Vd.16B 0 <= lane1 <= 15 b -> Vn.8B 0 <= lane2 <= 7	INS Vd.B[lane1], Vn.B[lane2]	Vd.16B -> result	A64
uint16x4_t vcopy_lane_u16(uint16x4_t a, const int lane1, uint16x4_t b, const int lane2)	a -> Vd.4H 0 <= lane1 <= 3 b -> Vn.4H 0 <= lane2 <= 3	INS Vd.H[lane1], Vn.H[lane2]	Vd.4H -> result	A64
uint16x8_t vcopyq_lane_u16(uint16x8_t a, const int lane1, uint16x4_t b, const int lane2)	a -> Vd.8H 0 <= lane1 <= 7 b -> Vn.4H 0 <= lane2 <= 3	INS Vd.H[lane1], Vn.H[lane2]	Vd.8H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x2_t vcopy_lane_u32(uint32x2_t a, const int lane1, uint32x2_t b, const int lane2)	a -> Vd.2S 0 <= lane1 <= 1 b -> Vn.2S 0 <= lane2 <= 1	INS Vd.S[lane1], Vn.S[lane2]	Vd.2S -> result	A64
uint32x4_t vcopyq_lane_u32(uint32x4_t a, const int lane1, uint32x2_t b, const int lane2)	a -> Vd.4S 0 <= lane1 <= 3 b -> Vn.2S 0 <= lane2 <= 1	INS Vd.S[lane1], Vn.S[lane2]	Vd.4S -> result	A64
uint64x1_t vcopy_lane_u64(uint64x1_t a, const int lane1, uint64x1_t b, const int lane2)	a -> UNUSED lane1 == 0 b -> Vn.1D lane2 == 0	DUP Dd, Vn.D[lane2]	Dd -> result	A64
uint64x2_t vcopyq_lane_u64(uint64x2_t a, const int lane1, uint64x1_t b, const int lane2)	a -> Vd.2D 0 <= lane1 <= 1 b -> Vn.1D lane2 == 0	INS Vd.D[lane1], Vn.D[lane2]	Vd.2D -> result	A64
poly64x1_t vcopy_lane_p64(poly64x1_t a, const int lane1, poly64x1_t b, const int lane2)	a -> UNUSED lane1 == 0 b -> Vn.1D lane2 == 0	DUP Dd, Vn.D[lane2]	Dd -> result	A32/A64
poly64x2_t vcopyq_lane_p64(poly64x2_t a, const int lane1, poly64x1_t b, const int lane2)	a -> Vd.2D 0 <= lane1 <= 1 b -> Vn.1D lane2 == 0	INS Vd.D[lane1], Vn.D[lane2]	Vd.2D -> result	A32/A64
float32x2_t vcopy_lane_f32(float32x2_t a, const int lane1, float32x2_t b, const int lane2)	a -> Vd.2S 0 <= lane1 <= 1 b -> Vn.2S 0 <= lane2 <= 1	INS Vd.S[lane1], Vn.S[lane2]	Vd.2S -> result	A64
float32x4_t vcopyq_lane_f32(float32x4_t a, const int lane1, float32x2_t b, const int lane2)	a -> Vd.4S 0 <= lane1 <= 3 b -> Vn.2S 0 <= lane2 <= 1	INS Vd.S[lane1], Vn.S[lane2]	Vd.4S -> result	A64
float64x1_t vcopy_lane_f64(float64x1_t a, const int lane1, float64x1_t b, const int lane2)	a -> UNUSED lane1 == 0 b -> Vn.1D lane2 == 0	DUP Dd, Vn.D[lane2]	Dd -> result	A64
float64x2_t vcopyq_lane_f64(float64x2_t a, const int lane1, float64x1_t b, const int lane2)	a -> Vd.2D 0 <= lane1 <= 1 b -> Vn.1D lane2 == 0	INS Vd.D[lane1], Vn.D[lane2]	Vd.2D -> result	A64
poly8x8_t vcopy_lane_p8(poly8x8_t a, const int lane1, poly8x8_t b, const int lane2)	a -> Vd.8B 0 <= lane1 <= 7 b -> Vn.8B 0 <= lane2 <= 7	INS Vd.B[lane1], Vn.B[lane2]	Vd.8B -> result	A64
poly8x16_t vcopyq_lane_p8(poly8x16_t a, const int lane1, poly8x8_t b, const int lane2)	a -> Vd.16B 0 <= lane1 <= 15 b -> Vn.8B 0 <= lane2 <= 7	INS Vd.B[lane1], Vn.B[lane2]	Vd.16B -> result	A64
poly16x4_t vcopy_lane_p16(poly16x4_t a, const int lane1, poly16x4_t b, const int lane2)	a -> Vd.4H 0 <= lane1 <= 3 b -> Vn.4H 0 <= lane2 <= 3	INS Vd.H[lane1], Vn.H[lane2]	Vd.4H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly16x8_t vcopyq_lane_p16(poly16x8_t a, const int lane1, poly16x4_t b, const int lane2)	a -> Vd.8H 0 <= lane1 <= 7 b -> Vn.4H 0 <= lane2 <= 3	INS Vd.H[lane1], Vn.H[lane2]	Vd.8H -> result	A64
int8x8_t vcopy_laneq_s8(int8x8_t a, const int lane1, int8x16_t b, const int lane2)	a -> Vd.8B 0 <= lane1 <= 7 b -> Vn.16B 0 <= lane2 <= 15	INS Vd.B[lane1], Vn.B[lane2]	Vd.8B -> result	A64
int8x16_t vcopyq_laneq_s8(int8x16_t a, const int lane1, int8x16_t b, const int lane2)	a -> Vd.16B 0 <= lane1 <= 15 b -> Vn.16B 0 <= lane2 <= 15	INS Vd.B[lane1], Vn.B[lane2]	Vd.16B -> result	A64
int16x4_t vcopy_laneq_s16(int16x4_t a, const int lane1, int16x8_t b, const int lane2)	a -> Vd.4H 0 <= lane1 <= 3 b -> Vn.8H 0 <= lane2 <= 7	INS Vd.H[lane1], Vn.H[lane2]	Vd.4H -> result	A64
int16x8_t vcopyq_laneq_s16(int16x8_t a, const int lane1, int16x8_t b, const int lane2)	a -> Vd.8H 0 <= lane1 <= 7 b -> Vn.8H 0 <= lane2 <= 7	INS Vd.H[lane1], Vn.H[lane2]	Vd.8H -> result	A64
int32x2_t vcopy_laneq_s32(int32x2_t a, const int lane1, int32x4_t b, const int lane2)	a -> Vd.2S 0 <= lane1 <= 1 b -> Vn.4S 0 <= lane2 <= 3	INS Vd.S[lane1], Vn.S[lane2]	Vd.2S -> result	A64
int32x4_t vcopyq_laneq_s32(int32x4_t a, const int lane1, int32x4_t b, const int lane2)	a -> Vd.4S 0 <= lane1 <= 3 b -> Vn.4S 0 <= lane2 <= 3	INS Vd.S[lane1], Vn.S[lane2]	Vd.4S -> result	A64
int64x1_t vcopy_laneq_s64(int64x1_t a, const int lane1, int64x2_t b, const int lane2)	a -> UNUSED lane1 == 0 b -> Vn.2D 0 <= lane2 <= 1	DUP Dd, Vn.D[lane2]	Dd -> result	A64
int64x2_t vcopyq_laneq_s64(int64x2_t a, const int lane1, int64x2_t b, const int lane2)	a -> Vd.2D 0 <= lane1 <= 1 b -> Vn.2D 0 <= lane2 <= 1	INS Vd.D[lane1], Vn.D[lane2]	Vd.2D -> result	A64
uint8x8_t vcopy_laneq_u8(uint8x8_t a, const int lane1, uint8x16_t b, const int lane2)	a -> Vd.8B 0 <= lane1 <= 7 b -> Vn.16B 0 <= lane2 <= 15	INS Vd.B[lane1], Vn.B[lane2]	Vd.8B -> result	A64
uint8x16_t vcopyq_laneq_u8(uint8x16_t a, const int lane1, uint8x16_t b, const int lane2)	a -> Vd.16B 0 <= lane1 <= 15 b -> Vn.16B 0 <= lane2 <= 15	INS Vd.B[lane1], Vn.B[lane2]	Vd.16B -> result	A64
uint16x4_t vcopy_laneq_u16(uint16x4_t a, const int lane1, uint16x8_t b, const int lane2)	a -> Vd.4H 0 <= lane1 <= 3 b -> Vn.8H 0 <= lane2 <= 7	INS Vd.H[lane1], Vn.H[lane2]	Vd.4H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t vcopyq_laneq_u16(uint16x8_t a, const int lane1, uint16x8_t b, const int lane2)	a -> Vd.8H 0 <= lane1 <= 7 b -> Vn.8H 0 <= lane2 <= 7	INS Vd.H[lane1], Vn.H[lane2]	Vd.8H -> result	A64
uint32x2_t vcopy_laneq_u32(uint32x2_t a, const int lane1, uint32x4_t b, const int lane2)	a -> Vd.2S 0 <= lane1 <= 1 b -> Vn.4S 0 <= lane2 <= 3	INS Vd.S[lane1], Vn.S[lane2]	Vd.2S -> result	A64
uint32x4_t vcopyq_laneq_u32(uint32x4_t a, const int lane1, uint32x4_t b, const int lane2)	a -> Vd.4S 0 <= lane1 <= 3 b -> Vn.4S 0 <= lane2 <= 3	INS Vd.S[lane1], Vn.S[lane2]	Vd.4S -> result	A64
uint64x1_t vcopy_laneq_u64(uint64x1_t a, const int lane1, uint64x2_t b, const int lane2)	a -> UNUSED lane1 == 0 b -> Vn.2D 0 <= lane2 <= 1	DUP Dd, Vn.D[lane2]	Dd -> result	A64
uint64x2_t vcopyq_laneq_u64(uint64x2_t a, const int lane1, uint64x2_t b, const int lane2)	a -> Vd.2D 0 <= lane1 <= 1 b -> Vn.2D 0 <= lane2 <= 1	INS Vd.D[lane1], Vn.D[lane2]	Vd.2D -> result	A64
poly64x1_t vcopy_laneq_p64(poly64x1_t a, const int lane1, poly64x2_t b, const int lane2)	a -> UNUSED lane1 == 0 b -> Vn.2D 0 <= lane2 <= 1	DUP Dd, Vn.D[lane2]	Dd -> result	A32/A64
poly64x2_t vcopyq_laneq_p64(poly64x2_t a, const int lane1, poly64x2_t b, const int lane2)	a -> Vd.2D 0 <= lane1 <= 1 b -> Vn.2D 0 <= lane2 <= 1	INS Vd.D[lane1], Vn.D[lane2]	Vd.2D -> result	A32/A64
float32x2_t vcopy_laneq_f32(float32x2_t a, const int lane1, float32x4_t b, const int lane2)	a -> Vd.2S 0 <= lane1 <= 1 b -> Vn.4S 0 <= lane2 <= 3	INS Vd.S[lane1], Vn.S[lane2]	Vd.2S -> result	A64
float32x4_t vcopyq_laneq_f32(float32x4_t a, const int lane1, float32x4_t b, const int lane2)	a -> Vd.4S 0 <= lane1 <= 3 b -> Vn.4S 0 <= lane2 <= 3	INS Vd.S[lane1], Vn.S[lane2]	Vd.4S -> result	A64
float64x1_t vcopy_laneq_f64(float64x1_t a, const int lane1, float64x2_t b, const int lane2)	a -> UNUSED lane1 == 0 b -> Vn.2D 0 <= lane2 <= 1	DUP Dd, Vn.D[lane2]	Dd -> result	A64
float64x2_t vcopyq_laneq_f64(float64x2_t a, const int lane1, float64x2_t b, const int lane2)	a -> Vd.2D 0 <= lane1 <= 1 b -> Vn.2D 0 <= lane2 <= 1	INS Vd.D[lane1], Vn.D[lane2]	Vd.2D -> result	A64
poly8x8_t vcopy_laneq_p8(poly8x8_t a, const int lane1, poly8x16_t b, const int lane2)	a -> Vd.8B 0 <= lane1 <= 7 b -> Vn.16B 0 <= lane2 <= 15	INS Vd.B[lane1], Vn.B[lane2]	Vd.8B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly8x16_tvcopyq_laneq_p8(poly8x16_t a, const int lane1, poly8x16_t b, const int lane2)	a -> Vd.16B 0 <= lane1 <= 15 b -> Vn.16B 0 <= lane2 <= 15	INS Vd.B[lane1], Vn.B[lane2]	Vd.16B -> result	A64
poly16x4_tvcopy_laneq_p16(poly16x4_t a, const int lane1, poly16x8_t b, const int lane2)	a -> Vd.4H 0 <= lane1 <= 3 b -> Vn.8H 0 <= lane2 <= 7	INS Vd.H[lane1], Vn.H[lane2]	Vd.4H -> result	A64
poly16x8_tvcopyq_laneq_p16(poly16x8_t a, const int lane1, poly16x8_t b, const int lane2)	a -> Vd.8H 0 <= lane1 <= 7 b -> Vn.8H 0 <= lane2 <= 7	INS Vd.H[lane1], Vn.H[lane2]	Vd.8H -> result	A64
int8x8_tvrbit_s8(int8x8_t a)	a -> Vn.8B	RBIT Vd.8B, Vn.8B	Vd.8B -> result	A64
int8x16_tvrbitq_s8(int8x16_t a)	a -> Vn.16B	RBIT Vd.16B, Vn.16B	Vd.16B -> result	A64
uint8x8_tvrbit_u8(uint8x8_t a)	a -> Vn.8B	RBIT Vd.8B, Vn.8B	Vd.8B -> result	A64
uint8x16_tvrbitq_u8(uint8x16_t a)	a -> Vn.16B	RBIT Vd.16B, Vn.16B	Vd.16B -> result	A64
poly8x8_tvrbit_p8(poly8x8_t a)	a -> Vn.8B	RBIT Vd.8B, Vn.8B	Vd.8B -> result	A64
poly8x16_tvrbitq_p8(poly8x16_t a)	a -> Vn.16B	RBIT Vd.16B, Vn.16B	Vd.16B -> result	A64
int8x8_tvcreate_s8(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.8B -> result	v7/A32/A64
int16x4_tvcreate_s16(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.4H -> result	v7/A32/A64
int32x2_tvcreate_s32(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.2S -> result	v7/A32/A64
int64x1_tvcreate_s64(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.1D -> result	v7/A32/A64
uint8x8_tvcreate_u8(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.8B -> result	v7/A32/A64
uint16x4_tvcreate_u16(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.4H -> result	v7/A32/A64
uint32x2_tvcreate_u32(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.2S -> result	v7/A32/A64
uint64x1_tvcreate_u64(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.1D -> result	v7/A32/A64
poly64x1_tvcreate_p64(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.1D -> result	A32/A64
float16x4_tvcreate_f16(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.4H -> result	v7/A32/A64
float32x2_tvcreate_f32(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.2S -> result	v7/A32/A64
poly8x8_tvcreate_p8(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.8B -> result	v7/A32/A64
poly16x4_tvcreate_p16(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.4H -> result	v7/A32/A64
float64x1_tvcreate_f64(uint64_t a)	a -> Xn	INS Vd.D[0], Xn	Vd.1D -> result	A64
int8x8_tvdup_n_s8(int8_t value)	value -> rn	DUP Vd.8B, rn	Vd.8B -> result	v7/A32/A64
int8x16_tvdupq_n_s8(int8_t value)	value -> rn	DUP Vd.16B, rn	Vd.16B -> result	v7/A32/A64
int16x4_tvdup_n_s16(int16_t value)	value -> rn	DUP Vd.4H, rn	Vd.4H -> result	v7/A32/A64
int16x8_tvdupq_n_s16(int16_t value)	value -> rn	DUP Vd.8H, rn	Vd.8H -> result	v7/A32/A64
int32x2_tvdup_n_s32(int32_t value)	value -> rn	DUP Vd.2S, rn	Vd.2S -> result	v7/A32/A64
int32x4_tvdupq_n_s32(int32_t value)	value -> rn	DUP Vd.4S, rn	Vd.4S -> result	v7/A32/A64
int64x1_tvdup_n_s64(int64_t value)	value -> rn	INS Dd.D[0], xn	Vd.1D -> result	v7/A32/A64
int64x2_tvdupq_n_s64(int64_t value)	value -> rn	DUP Vd.2D, rn	Vd.2D -> result	v7/A32/A64
uint8x8_tvdup_n_u8(uint8_t value)	value -> rn	DUP Vd.8B, rn	Vd.8B -> result	v7/A32/A64
uint8x16_tvdupq_n_u8(uint8_t value)	value -> rn	DUP Vd.16B, rn	Vd.16B -> result	v7/A32/A64
uint16x4_tvdup_n_u16(uint16_t value)	value -> rn	DUP Vd.4H, rn	Vd.4H -> result	v7/A32/A64
uint16x8_tvdupq_n_u16(uint16_t value)	value -> rn	DUP Vd.8H, rn	Vd.8H -> result	v7/A32/A64
uint32x2_tvdup_n_u32(uint32_t value)	value -> rn	DUP Vd.2S, rn	Vd.2S -> result	v7/A32/A64
uint32x4_tvdupq_n_u32(uint32_t value)	value -> rn	DUP Vd.4S, rn	Vd.4S -> result	v7/A32/A64
uint64x1_tvdup_n_u64(uint64_t value)	value -> rn	INS Dd.D[0], xn	Vd.1D -> result	v7/A32/A64
uint64x2_tvdupq_n_u64(uint64_t value)	value -> rn	DUP Vd.2D, rn	Vd.2D -> result	v7/A32/A64
poly64x1_tvdup_n_p64(poly64_t value)	value -> rn	INS Dd.D[0], xn	Vd.1D -> result	A32/A64
poly64x2_tvdupq_n_p64(poly64_t value)	value -> rn	DUP Vd.2D, rn	Vd.2D -> result	A32/A64
float32x2_tvdup_n_f32(float32_t value)	value -> rn	DUP Vd.2S, rn	Vd.2S -> result	v7/A32/A64
float32x4_tvdupq_n_f32(float32_t value)	value -> rn	DUP Vd.4S, rn	Vd.4S -> result	v7/A32/A64
poly8x8_tvdup_n_p8(poly8_t value)	value -> rn	DUP Vd.8B, rn	Vd.8B -> result	v7/A32/A64
poly8x16_tvdupq_n_p8(poly8_t value)	value -> rn	DUP Vd.16B, rn	Vd.16B -> result	v7/A32/A64
poly16x4_tvdup_n_p16(poly16_t value)	value -> rn	DUP Vd.4H, rn	Vd.4H -> result	v7/A32/A64
poly16x8_tvdupq_n_p16(poly16_t value)	value -> rn	DUP Vd.8H, rn	Vd.8H -> result	v7/A32/A64
float64x1_tvdup_n_f64(float64_t value)	value -> rn	INS Dd.D[0], xn	Vd.1D -> result	A64
float64x2_tvdupq_n_f64(float64_t value)	value -> rn	DUP Vd.2D, rn	Vd.2D -> result	A64
int8x8_tvmov_n_s8(int8_t value)	value -> rn	DUP Vd.8B, rn	Vd.8B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t vmovq_n_s8(int8_t value)	value -> rn	DUP Vd.16B,rn	Vd.16B -> result	v7/A32/A64
int16x4_t vmov_n_s16(int16_t value)	value -> rn	DUP Vd.4H,rn	Vd.4H -> result	v7/A32/A64
int16x8_t vmovq_n_s16(int16_t value)	value -> rn	DUP Vd.8H,rn	Vd.8H -> result	v7/A32/A64
int32x2_t vmov_n_s32(int32_t value)	value -> rn	DUP Vd.2S,rn	Vd.2S -> result	v7/A32/A64
int32x4_t vmovq_n_s32(int32_t value)	value -> rn	DUP Vd.4S,rn	Vd.4S -> result	v7/A32/A64
int64x1_t vmov_n_s64(int64_t value)	value -> rn	DUP Vd.1D,rn	Vd.1D -> result	v7/A32/A64
int64x2_t vmovq_n_s64(int64_t value)	value -> rn	DUP Vd.2D,rn	Vd.2D -> result	v7/A32/A64
uint8x8_t vmov_n_u8(uint8_t value)	value -> rn	DUP Vd.8B,rn	Vd.8B -> result	v7/A32/A64
uint8x16_t vmovq_n_u8(uint8_t value)	value -> rn	DUP Vd.16B,rn	Vd.16B -> result	v7/A32/A64
uint16x4_t vmov_n_u16(uint16_t value)	value -> rn	DUP Vd.4H,rn	Vd.4H -> result	v7/A32/A64
uint16x8_t vmovq_n_u16(uint16_t value)	value -> rn	DUP Vd.8H,rn	Vd.8H -> result	v7/A32/A64
uint32x2_t vmov_n_u32(uint32_t value)	value -> rn	DUP Vd.2S,rn	Vd.2S -> result	v7/A32/A64
uint32x4_t vmovq_n_u32(uint32_t value)	value -> rn	DUP Vd.4S,rn	Vd.4S -> result	v7/A32/A64
uint64x1_t vmov_n_u64(uint64_t value)	value -> rn	DUP Vd.1D,rn	Vd.1D -> result	v7/A32/A64
uint64x2_t vmovq_n_u64(uint64_t value)	value -> rn	DUP Vd.2D,rn	Vd.2D -> result	v7/A32/A64
float32x2_t vmov_n_f32(float32_t value)	value -> rn	DUP Vd.2S,rn	Vd.2S -> result	v7/A32/A64
float32x4_t vmovq_n_f32(float32_t value)	value -> rn	DUP Vd.4S,rn	Vd.4S -> result	v7/A32/A64
poly8x8_t vmov_n_p8(poly8_t value)	value -> rn	DUP Vd.8B,rn	Vd.8B -> result	v7/A32/A64
poly8x16_t vmovq_n_p8(poly8_t value)	value -> rn	DUP Vd.16B,rn	Vd.16B -> result	v7/A32/A64
poly16x4_t vmov_n_p16(poly16_t value)	value -> rn	DUP Vd.4H,rn	Vd.4H -> result	v7/A32/A64
poly16x8_t vmovq_n_p16(poly16_t value)	value -> rn	DUP Vd.8H,rn	Vd.8H -> result	v7/A32/A64
float64x1_t vmov_n_f64(float64_t value)	value -> rn	DUP Vd.1D,rn	Vd.1D -> result	A64
float64x2_t vmovq_n_f64(float64_t value)	value -> rn	DUP Vd.2D,rn	Vd.2D -> result	A64
int8x8_t vdup_lane_s8(int8x8_t vec, const int lane)	vec -> Vn.8B 0 <= lane <= 7	DUP Vd.8B,Vn.B[lane]	Vd.8B -> result	v7/A32/A64
int8x16_t vdupq_lane_s8(int8x8_t vec, const int lane)	vec -> Vn.8B 0 <= lane <= 7	DUP Vd.16B,Vn.B[lane]	Vd.16B -> result	v7/A32/A64
int16x4_t vdup_lane_s16(int16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Vd.4H,Vn.H[lane]	Vd.4H -> result	v7/A32/A64
int16x8_t vdupq_lane_s16(int16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Vd.8H,Vn.H[lane]	Vd.8H -> result	v7/A32/A64
int32x2_t vdup_lane_s32(int32x2_t vec, const int lane)	vec -> Vn.2S 0 <= lane <= 1	DUP Vd.2S,Vn.S[lane]	Vd.2S -> result	v7/A32/A64
int32x4_t vdupq_lane_s32(int32x2_t vec, const int lane)	vec -> Vn.2S 0 <= lane <= 1	DUP Vd.4S,Vn.S[lane]	Vd.4S -> result	v7/A32/A64
int64x1_t vdup_lane_s64(int64x1_t vec, const int lane)	vec -> Vn.1D lane == 0	DUP Dd,Vn.D[lane]	Dd -> result	v7/A32/A64
int64x2_t vdupq_lane_s64(int64x1_t vec, const int lane)	vec -> Vn.1D lane == 0	DUP Vd.2D,Vn.D[lane]	Vd.2D -> result	v7/A32/A64
uint8x8_t vdup_lane_u8(uint8x8_t vec, const int lane)	vec -> Vn.8B 0 <= lane <= 7	DUP Vd.8B,Vn.B[lane]	Vd.8B -> result	v7/A32/A64
uint8x16_t vdupq_lane_u8(uint8x8_t vec, const int lane)	vec -> Vn.8B 0 <= lane <= 7	DUP Vd.16B,Vn.B[lane]	Vd.16B -> result	v7/A32/A64
uint16x4_t vdup_lane_u16(uint16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Vd.4H,Vn.H[lane]	Vd.4H -> result	v7/A32/A64
uint16x8_t vdupq_lane_u16(uint16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Vd.8H,Vn.H[lane]	Vd.8H -> result	v7/A32/A64
uint32x2_t vdup_lane_u32(uint32x2_t vec, const int lane)	vec -> Vn.2S 0 <= lane <= 1	DUP Vd.2S,Vn.S[lane]	Vd.2S -> result	v7/A32/A64
uint32x4_t vdupq_lane_u32(uint32x2_t vec, const int lane)	vec -> Vn.2S 0 <= lane <= 1	DUP Vd.4S,Vn.S[lane]	Vd.4S -> result	v7/A32/A64
uint64x1_t vdup_lane_u64(uint64x1_t vec, const int lane)	vec -> Vn.1D lane == 0	DUP Dd,Vn.D[lane]	Dd -> result	v7/A32/A64
uint64x2_t vdupq_lane_u64(uint64x1_t vec, const int lane)	vec -> Vn.1D lane == 0	DUP Vd.2D,Vn.D[lane]	Vd.2D -> result	v7/A32/A64
poly64x1_t vdup_lane_p64(poly64x1_t vec, const int lane)	vec -> Vn.1D lane == 0	DUP Dd,Vn.D[lane]	Dd -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly64x2_t vdupq_lane_p64(poly64x1_t vec, const int lane)	vec -> Vn.1D lane == 0	DUP Vd.2D,Vn.D[lane]	Vd.2D -> result	A32/A64
float32x2_t vdup_lane_f32(float32x2_t vec, const int lane)	vec -> Vn.2S 0 <= lane <= 1	DUP Vd.2S,Vn.S[lane]	Vd.2S -> result	v7/A32/A64
float32x4_t vdupq_lane_f32(float32x2_t vec, const int lane)	vec -> Vn.2S 0 <= lane <= 1	DUP Vd.4S,Vn.S[lane]	Vd.4S -> result	v7/A32/A64
poly8x8_t vdup_lane_p8(poly8x8_t vec, const int lane)	vec -> Vn.8B 0 <= lane <= 7	DUP Vd.8B,Vn.B[lane]	Vd.8B -> result	v7/A32/A64
poly8x16_t vdupq_lane_p8(poly8x8_t vec, const int lane)	vec -> Vn.8B 0 <= lane <= 7	DUP Vd.16B,Vn.B[lane]	Vd.16B -> result	v7/A32/A64
poly16x4_t vdup_lane_p16(poly16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Vd.4H,Vn.H[lane]	Vd.4H -> result	v7/A32/A64
poly16x8_t vdupq_lane_p16(poly16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Vd.8H,Vn.H[lane]	Vd.8H -> result	v7/A32/A64
float64x1_t vdup_lane_f64(float64x1_t vec, const int lane)	vec -> Vn.1D lane == 0	DUP Dd,Vn.D[lane]	Dd -> result	A64
float64x2_t vdupq_lane_f64(float64x1_t vec, const int lane)	vec -> Vn.1D lane == 0	DUP Vd.2D,Vn.D[lane]	Vd.2D -> result	A64
int8x8_t vdup_laneq_s8(int8x16_t vec, const int lane)	vec -> Vn.16B 0 <= lane <= 15	DUP Vd.8B,Vn.B[lane]	Vd.8B -> result	A64
int8x16_t vdupq_laneq_s8(int8x16_t vec, const int lane)	vec -> Vn.16B 0 <= lane <= 15	DUP Vd.16B,Vn.B[lane]	Vd.16B -> result	A64
int16x4_t vdup_laneq_s16(int16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Vd.4H,Vn.H[lane]	Vd.4H -> result	A64
int16x8_t vdupq_laneq_s16(int16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Vd.8H,Vn.H[lane]	Vd.8H -> result	A64
int32x2_t vdup_laneq_s32(int32x4_t vec, const int lane)	vec -> Vn.4S 0 <= lane <= 3	DUP Vd.2S,Vn.S[lane]	Vd.2S -> result	A64
int32x4_t vdupq_laneq_s32(int32x4_t vec, const int lane)	vec -> Vn.4S 0 <= lane <= 3	DUP Vd.4S,Vn.S[lane]	Vd.4S -> result	A64
int64x1_t vdup_laneq_s64(int64x2_t vec, const int lane)	vec -> Vn.2D 0 <= lane <= 1	DUP Dd,Vn.D[lane]	Dd -> result	A64
int64x2_t vdupq_laneq_s64(int64x2_t vec, const int lane)	vec -> Vn.2D 0 <= lane <= 1	DUP Vd.2D,Vn.D[lane]	Vd.2D -> result	A64
uint8x8_t vdup_laneq_u8(uint8x16_t vec, const int lane)	vec -> Vn.16B 0 <= lane <= 15	DUP Vd.8B,Vn.B[lane]	Vd.8B -> result	A64
uint8x16_t vdupq_laneq_u8(uint8x16_t vec, const int lane)	vec -> Vn.16B 0 <= lane <= 15	DUP Vd.16B,Vn.B[lane]	Vd.16B -> result	A64
uint16x4_t vdup_laneq_u16(uint16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Vd.4H,Vn.H[lane]	Vd.4H -> result	A64
uint16x8_t vdupq_laneq_u16(uint16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Vd.8H,Vn.H[lane]	Vd.8H -> result	A64
uint32x2_t vdup_laneq_u32(uint32x4_t vec, const int lane)	vec -> Vn.4S 0 <= lane <= 3	DUP Vd.2S,Vn.S[lane]	Vd.2S -> result	A64
uint32x4_t vdupq_laneq_u32(uint32x4_t vec, const int lane)	vec -> Vn.4S 0 <= lane <= 3	DUP Vd.4S,Vn.S[lane]	Vd.4S -> result	A64
uint64x1_t vdup_laneq_u64(uint64x2_t vec, const int lane)	vec -> Vn.2D 0 <= lane <= 1	DUP Dd,Vn.D[lane]	Dd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64x2_t vdupq_laneq_u64(uint64x2_t vec, const int lane)	vec -> Vn.2D 0 <= lane <= 1	DUP Vd.2D,Vn.D[lane]	Vd.2D -> result	A64
poly64x1_t vdup_laneq_p64(poly64x2_t vec, const int lane)	vec -> Vn.2D 0 <= lane <= 1	DUP Dd,Vn.D[lane]	Dd -> result	A64
poly64x2_t vdupq_laneq_p64(poly64x2_t vec, const int lane)	vec -> Vn.2D 0 <= lane <= 1	DUP Vd.2D,Vn.D[lane]	Vd.2D -> result	A64
float32x2_t vdup_laneq_f32(float32x4_t vec, const int lane)	vec -> Vn.4S 0 <= lane <= 3	DUP Vd.2S,Vn.S[lane]	Vd.2S -> result	A64
float32x4_t vdupq_laneq_f32(float32x4_t vec, const int lane)	vec -> Vn.4S 0 <= lane <= 3	DUP Vd.4S,Vn.S[lane]	Vd.4S -> result	A64
poly8x8_t vdup_laneq_p8(poly8x16_t vec, const int lane)	vec -> Vn.16B 0 <= lane <= 15	DUP Vd.8B,Vn.B[lane]	Vd.8B -> result	A64
poly8x16_t vdupq_laneq_p8(poly8x16_t vec, const int lane)	vec -> Vn.16B 0 <= lane <= 15	DUP Vd.16B,Vn.B[lane]	Vd.16B -> result	A64
poly16x4_t vdup_laneq_p16(poly16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Vd.4H,Vn.H[lane]	Vd.4H -> result	A64
poly16x8_t vdupq_laneq_p16(poly16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Vd.8H,Vn.H[lane]	Vd.8H -> result	A64
float64x1_t vdup_laneq_f64(float64x2_t vec, const int lane)	vec -> Vn.2D 0 <= lane <= 1	DUP Dd,Vn.D[lane]	Dd -> result	A64
float64x2_t vdupq_laneq_f64(float64x2_t vec, const int lane)	vec -> Vn.2D 0 <= lane <= 1	DUP Vd.2D,Vn.D[lane]	Vd.2D -> result	A64
int8x16_t vcombine_s8(int8x8_t low, int8x8_t high)	low -> Vn.8B high -> Vm.8B	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.16B -> result	v7/A32/A64
int16x8_t vcombine_s16(int16x4_t low, int16x4_t high)	low -> Vn.4H high -> Vm.4H	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.8H -> result	v7/A32/A64
int32x4_t vcombine_s32(int32x2_t low, int32x2_t high)	low -> Vn.2S high -> Vm.2S	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.4S -> result	v7/A32/A64
int64x2_t vcombine_s64(int64x1_t low, int64x1_t high)	low -> Vn.1D high -> Vm.1D	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.2D -> result	v7/A32/A64
uint8x16_t vcombine_u8(uint8x8_t low, uint8x8_t high)	low -> Vn.8B high -> Vm.8B	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.16B -> result	v7/A32/A64
uint16x8_t vcombine_u16(uint16x4_t low, uint16x4_t high)	low -> Vn.4H high -> Vm.4H	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.8H -> result	v7/A32/A64
uint32x4_t vcombine_u32(uint32x2_t low, uint32x2_t high)	low -> Vn.2S high -> Vm.2S	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.4S -> result	v7/A32/A64
uint64x2_t vcombine_u64(uint64x1_t low, uint64x1_t high)	low -> Vn.1D high -> Vm.1D	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.2D -> result	v7/A32/A64
poly64x2_t vcombine_p64(poly64x1_t low, poly64x1_t high)	low -> Vn.1D high -> Vm.1D	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.2D -> result	A32/A64
float16x8_t vcombine_f16(float16x4_t low, float16x4_t high)	low -> Vn.4H high -> Vm.4H	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.8H -> result	v7/A32/A64
float32x4_t vcombine_f32(float32x2_t low, float32x2_t high)	low -> Vn.2S high -> Vm.2S	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.4S -> result	v7/A32/A64
poly8x16_t vcombine_p8(poly8x8_t low, poly8x8_t high)	low -> Vn.8B high -> Vm.8B	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.16B -> result	v7/A32/A64
poly16x8_t vcombine_p16(poly16x4_t low, poly16x4_t high)	low -> Vn.4H high -> Vm.4H	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.8H -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64x2_t vcombine_f64(float64x1_t low, float64x1_t high)	low -> Vn.1D high -> Vm.1D	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.2D -> result	A64
int8x8_t vget_high_s8(int8x16_t a)	a -> Vn.16B	DUP Vd.1D,Vn.D[1]	Vd.8B -> result	v7/A32/A64
int16x4_t vget_high_s16(int16x8_t a)	a -> Vn.8H	DUP Vd.1D,Vn.D[1]	Vd.4H -> result	v7/A32/A64
int32x2_t vget_high_s32(int32x4_t a)	a -> Vn.4S	DUP Vd.1D,Vn.D[1]	Vd.2S -> result	v7/A32/A64
int64x1_t vget_high_s64(int64x2_t a)	a -> Vn.2D	DUP Vd.1D,Vn.D[1]	Vd.1D -> result	v7/A32/A64
uint8x8_t vget_high_u8(uint8x16_t a)	a -> Vn.16B	DUP Vd.1D,Vn.D[1]	Vd.8B -> result	v7/A32/A64
uint16x4_t vget_high_u16(uint16x8_t a)	a -> Vn.8H	DUP Vd.1D,Vn.D[1]	Vd.4H -> result	v7/A32/A64
uint32x2_t vget_high_u32(uint32x4_t a)	a -> Vn.4S	DUP Vd.1D,Vn.D[1]	Vd.2S -> result	v7/A32/A64
uint64x1_t vget_high_u64(uint64x2_t a)	a -> Vn.2D	DUP Vd.1D,Vn.D[1]	Vd.1D -> result	v7/A32/A64
poly64x1_t vget_high_p64(poly64x2_t a)	a -> Vn.2D	DUP Vd.1D,Vn.D[1]	Vd.1D -> result	A32/A64
float16x4_t vget_high_f16(float16x8_t a)	a -> Vn.8H	DUP Vd.1D,Vn.D[1]	Vd.4H -> result	v7/A32/A64
float32x2_t vget_high_f32(float32x4_t a)	a -> Vn.4S	DUP Vd.1D,Vn.D[1]	Vd.2S -> result	v7/A32/A64
poly8x8_t vget_high_p8(poly8x16_t a)	a -> Vn.16B	DUP Vd.1D,Vn.D[1]	Vd.8B -> result	v7/A32/A64
poly16x4_t vget_high_p16(poly16x8_t a)	a -> Vn.8H	DUP Vd.1D,Vn.D[1]	Vd.4H -> result	v7/A32/A64
float64x1_t vget_high_f64(float64x2_t a)	a -> Vn.2D	DUP Vd.1D,Vn.D[1]	Vd.1D -> result	A64
int8x8_t vget_low_s8(int8x16_t a)	a -> Vn.16B	DUP Vd.1D,Vn.D[0]	Vd.8B -> result	v7/A32/A64
int16x4_t vget_low_s16(int16x8_t a)	a -> Vn.8H	DUP Vd.1D,Vn.D[0]	Vd.4H -> result	v7/A32/A64
int32x2_t vget_low_s32(int32x4_t a)	a -> Vn.4S	DUP Vd.1D,Vn.D[0]	Vd.2S -> result	v7/A32/A64
int64x1_t vget_low_s64(int64x2_t a)	a -> Vn.2D	DUP Vd.1D,Vn.D[0]	Vd.1D -> result	v7/A32/A64
uint8x8_t vget_low_u8(uint8x16_t a)	a -> Vn.16B	DUP Vd.1D,Vn.D[0]	Vd.8B -> result	v7/A32/A64
uint16x4_t vget_low_u16(uint16x8_t a)	a -> Vn.8H	DUP Vd.1D,Vn.D[0]	Vd.4H -> result	v7/A32/A64
uint32x2_t vget_low_u32(uint32x4_t a)	a -> Vn.4S	DUP Vd.1D,Vn.D[0]	Vd.2S -> result	v7/A32/A64
uint64x1_t vget_low_u64(uint64x2_t a)	a -> Vn.2D	DUP Vd.1D,Vn.D[0]	Vd.1D -> result	v7/A32/A64
poly64x1_t vget_low_p64(poly64x2_t a)	a -> Vn.2D	DUP Vd.1D,Vn.D[0]	Vd.1D -> result	A32/A64
float16x4_t vget_low_f16(float16x8_t a)	a -> Vn.8H	DUP Vd.1D,Vn.D[0]	Vd.4H -> result	v7/A32/A64
float32x2_t vget_low_f32(float32x4_t a)	a -> Vn.4S	DUP Vd.1D,Vn.D[0]	Vd.2S -> result	v7/A32/A64
poly8x8_t vget_low_p8(poly8x16_t a)	a -> Vn.16B	DUP Vd.1D,Vn.D[0]	Vd.8B -> result	v7/A32/A64
poly16x4_t vget_low_p16(poly16x8_t a)	a -> Vn.8H	DUP Vd.1D,Vn.D[0]	Vd.4H -> result	v7/A32/A64
float64x1_t vget_low_f64(float64x2_t a)	a -> Vn.2D	DUP Vd.1D,Vn.D[0]	Vd.1D -> result	A64
int8_t vdupb_lane_s8(int8x8_t vec, const int lane)	vec -> Vn.8B 0 <= lane <= 7	DUP Bd,Vn.B[lane]	Bd -> result	A64
int16_t vduph_lane_s16(int16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Hd,Vn.H[lane]	Hd -> result	A64
int32_t vdups_lane_s32(int32x2_t vec, const int lane)	vec -> Vn.2S 0 <= lane <= 1	DUP Sd,Vn.S[lane]	Sd -> result	A64
int64_t vdupd_lane_s64(int64x1_t vec, const int lane)	vec -> Vn.1D lane == 0	DUP Dd,Vn.D[lane]	Dd -> result	A64
uint8_t vdupb_lane_u8(uint8x8_t vec, const int lane)	vec -> Vn.8B 0 <= lane <= 7	DUP Bd,Vn.B[lane]	Bd -> result	A64
uint16_t vduph_lane_u16(uint16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Hd,Vn.H[lane]	Hd -> result	A64
uint32_t vdups_lane_u32(uint32x2_t vec, const int lane)	vec -> Vn.2S 0 <= lane <= 1	DUP Sd,Vn.S[lane]	Sd -> result	A64
uint64_t vdupd_lane_u64(uint64x1_t vec, const int lane)	vec -> Vn.1D lane == 0	DUP Dd,Vn.D[lane]	Dd -> result	A64
float32_t vdups_lane_f32(float32x2_t vec, const int lane)	vec -> Vn.2S 0 <= lane <= 1	DUP Sd,Vn.S[lane]	Sd -> result	A64
float64_t vdupd_lane_f64(float64x1_t vec, const int lane)	vec -> Vn.1D lane == 0	DUP Dd,Vn.D[lane]	Dd -> result	A64
poly8_t vdupb_lane_p8(poly8x8_t vec, const int lane)	vec -> Vn.8B 0 <= lane <= 7	DUP Bd,Vn.B[lane]	Bd -> result	A64
poly16_t vduph_lane_p16(poly16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Hd,Vn.H[lane]	Hd -> result	A64
int8_t vdupb_laneq_s8(int8x16_t vec, const int lane)	vec -> Vn.16B 0 <= lane <= 15	DUP Bd,Vn.B[lane]	Bd -> result	A64
int16_t vduph_laneq_s16(int16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Hd,Vn.H[lane]	Hd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32_t vdups_laneq_s32(int32x4_t vec, const int lane)	vec -> Vn.4S 0 <= lane <= 3	DUP Sd,Vn.S[lane]	Sd -> result	A64
int64_t vdupd_laneq_s64(int64x2_t vec, const int lane)	vec -> Vn.2D 0 <= lane <= 1	DUP Dd,Vn.D[lane]	Dd -> result	A64
uint8_t vdupb_laneq_u8(uint8x16_t vec, const int lane)	vec -> Vn.16B 0 <= lane <= 15	DUP Bd,Vn.B[lane]	Bd -> result	A64
uint16_t vduph_laneq_u16(uint16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Hd,Vn.H[lane]	Hd -> result	A64
uint32_t vdups_laneq_u32(uint32x4_t vec, const int lane)	vec -> Vn.4S 0 <= lane <= 3	DUP Sd,Vn.S[lane]	Sd -> result	A64
uint64_t vdupd_laneq_u64(uint64x2_t vec, const int lane)	vec -> Vn.2D 0 <= lane <= 1	DUP Dd,Vn.D[lane]	Dd -> result	A64
float32_t vdups_laneq_f32(float32x4_t vec, const int lane)	vec -> Vn.4S 0 <= lane <= 3	DUP Sd,Vn.S[lane]	Sd -> result	A64
float64_t vdupd_laneq_f64(float64x2_t vec, const int lane)	vec -> Vn.2D 0 <= lane <= 1	DUP Dd,Vn.D[lane]	Dd -> result	A64
poly8_t vdupb_laneq_p8(poly8x16_t vec, const int lane)	vec -> Vn.16B 0 <= lane <= 15	DUP Bd,Vn.B[lane]	Bd -> result	A64
poly16_t vduph_laneq_p16(poly16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Hd,Vn.H[lane]	Hd -> result	A64
int8x8_t vld1_s8(int8_t const * ptr)	ptr -> Xn	LD1 {Vt.8B},[Xn]	Vt.8B -> result	v7/A32/A64
int8x16_t vld1q_s8(int8_t const * ptr)	ptr -> Xn	LD1 {Vt.16B},[Xn]	Vt.16B -> result	v7/A32/A64
int16x4_t vld1_s16(int16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H},[Xn]	Vt.4H -> result	v7/A32/A64
int16x8_t vld1q_s16(int16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H},[Xn]	Vt.8H -> result	v7/A32/A64
int32x2_t vld1_s32(int32_t const * ptr)	ptr -> Xn	LD1 {Vt.2S},[Xn]	Vt.2S -> result	v7/A32/A64
int32x4_t vld1q_s32(int32_t const * ptr)	ptr -> Xn	LD1 {Vt.4S},[Xn]	Vt.4S -> result	v7/A32/A64
int64x1_t vld1_s64(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D},[Xn]	Vt.1D -> result	v7/A32/A64
int64x2_t vld1q_s64(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D},[Xn]	Vt.2D -> result	v7/A32/A64
uint8x8_t vld1_u8(uint8_t const * ptr)	ptr -> Xn	LD1 {Vt.8B},[Xn]	Vt.8B -> result	v7/A32/A64
uint8x16_t vld1q_u8(uint8_t const * ptr)	ptr -> Xn	LD1 {Vt.16B},[Xn]	Vt.16B -> result	v7/A32/A64
uint16x4_t vld1_u16(uint16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H},[Xn]	Vt.4H -> result	v7/A32/A64
uint16x8_t vld1q_u16(uint16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H},[Xn]	Vt.8H -> result	v7/A32/A64
uint32x2_t vld1_u32(uint32_t const * ptr)	ptr -> Xn	LD1 {Vt.2S},[Xn]	Vt.2S -> result	v7/A32/A64
uint32x4_t vld1q_u32(uint32_t const * ptr)	ptr -> Xn	LD1 {Vt.4S},[Xn]	Vt.4S -> result	v7/A32/A64
uint64x1_t vld1_u64(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D},[Xn]	Vt.1D -> result	v7/A32/A64
uint64x2_t vld1q_u64(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D},[Xn]	Vt.2D -> result	v7/A32/A64
poly64x1_t vld1_p64(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D},[Xn]	Vt.1D -> result	A32/A64
poly64x2_t vld1q_p64(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D},[Xn]	Vt.2D -> result	A32/A64
float16x4_t vld1_f16(float16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H},[Xn]	Vt.4H -> result	v7/A32/A64
float16x8_t vld1q_f16(float16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H},[Xn]	Vt.8H -> result	v7/A32/A64
float32x2_t vld1_f32(float32_t const * ptr)	ptr -> Xn	LD1 {Vt.2S},[Xn]	Vt.2S -> result	v7/A32/A64
float32x4_t vld1q_f32(float32_t const * ptr)	ptr -> Xn	LD1 {Vt.4S},[Xn]	Vt.4S -> result	v7/A32/A64
poly8x8_t vld1_p8(poly8_t const * ptr)	ptr -> Xn	LD1 {Vt.8B},[Xn]	Vt.8B -> result	v7/A32/A64
poly8x16_t vld1q_p8(poly8_t const * ptr)	ptr -> Xn	LD1 {Vt.16B},[Xn]	Vt.16B -> result	v7/A32/A64
poly16x4_t vld1_p16(poly16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H},[Xn]	Vt.4H -> result	v7/A32/A64
poly16x8_t vld1q_p16(poly16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H},[Xn]	Vt.8H -> result	v7/A32/A64
float64x1_t vld1_f64(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D},[Xn]	Vt.1D -> result	A64
float64x2_t vld1q_f64(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D},[Xn]	Vt.2D -> result	A64
int8x8_t vld1_lane_s8(int8_t const * ptr, int8x8_t src, const int lane)	ptr -> Xn src -> Vt.8B 0 <= lane <= 7	LD1 {Vt.b}[lane],[Xn]	Vt.8B -> result	v7/A32/A64
int8x16_t vld1q_lane_s8(int8_t const * ptr, int8x16_t src, const int lane)	ptr -> Xn src -> Vt.16B 0 <= lane <= 15	LD1 {Vt.b}[lane],[Xn]	Vt.16B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x4_t vld1_lane_s16(int16_t const * ptr, int16x4_t src, const int lane)	ptr -> Xn src -> Vt.4H 0 <= lane <= 3	LD1 {Vt.H}[lane],[Xn]	Vt.4H -> result	v7/A32/A64
int16x8_t vld1q_lane_s16(int16_t const * ptr, int16x8_t src, const int lane)	ptr -> Xn src -> Vt.8H 0 <= lane <= 7	LD1 {Vt.H}[lane],[Xn]	Vt.8H -> result	v7/A32/A64
int32x2_t vld1_lane_s32(int32_t const * ptr, int32x2_t src, const int lane)	ptr -> Xn src -> Vt.2S 0 <= lane <= 1	LD1 {Vt.S}[lane],[Xn]	Vt.2S -> result	v7/A32/A64
int32x4_t vld1q_lane_s32(int32_t const * ptr, int32x4_t src, const int lane)	ptr -> Xn src -> Vt.4S 0 <= lane <= 3	LD1 {Vt.S}[lane],[Xn]	Vt.4S -> result	v7/A32/A64
int64x1_t vld1_lane_s64(int64_t const * ptr, int64x1_t src, const int lane)	ptr -> Xn src -> Vt.1D lane == 0	LD1 {Vt.D}[lane],[Xn]	Vt.1D -> result	v7/A32/A64
int64x2_t vld1q_lane_s64(int64_t const * ptr, int64x2_t src, const int lane)	ptr -> Xn src -> Vt.2D 0 <= lane <= 1	LD1 {Vt.D}[lane],[Xn]	Vt.2D -> result	v7/A32/A64
uint8x8_t vld1_lane_u8(uint8_t const * ptr, uint8x8_t src, const int lane)	ptr -> Xn src -> Vt.8B 0 <= lane <= 7	LD1 {Vt.B}[lane],[Xn]	Vt.8B -> result	v7/A32/A64
uint8x16_t vld1q_lane_u8(uint8_t const * ptr, uint8x16_t src, const int lane)	ptr -> Xn src -> Vt.16B 0 <= lane <= 15	LD1 {Vt.B}[lane],[Xn]	Vt.16B -> result	v7/A32/A64
uint16x4_t vld1_lane_u16(uint16_t const * ptr, uint16x4_t src, const int lane)	ptr -> Xn src -> Vt.4H 0 <= lane <= 3	LD1 {Vt.H}[lane],[Xn]	Vt.4H -> result	v7/A32/A64
uint16x8_t vld1q_lane_u16(uint16_t const * ptr, uint16x8_t src, const int lane)	ptr -> Xn src -> Vt.8H 0 <= lane <= 7	LD1 {Vt.H}[lane],[Xn]	Vt.8H -> result	v7/A32/A64
uint32x2_t vld1_lane_u32(uint32_t const * ptr, uint32x2_t src, const int lane)	ptr -> Xn src -> Vt.2S 0 <= lane <= 1	LD1 {Vt.S}[lane],[Xn]	Vt.2S -> result	v7/A32/A64
uint32x4_t vld1q_lane_u32(uint32_t const * ptr, uint32x4_t src, const int lane)	ptr -> Xn src -> Vt.4S 0 <= lane <= 3	LD1 {Vt.S}[lane],[Xn]	Vt.4S -> result	v7/A32/A64
uint64x1_t vld1_lane_u64(uint64_t const * ptr, uint64x1_t src, const int lane)	ptr -> Xn src -> Vt.1D lane == 0	LD1 {Vt.D}[lane],[Xn]	Vt.1D -> result	v7/A32/A64
uint64x2_t vld1q_lane_u64(uint64_t const * ptr, uint64x2_t src, const int lane)	ptr -> Xn src -> Vt.2D 0 <= lane <= 1	LD1 {Vt.D}[lane],[Xn]	Vt.2D -> result	v7/A32/A64
poly64x1_t vld1_lane_p64(poly64_t const * ptr, poly64x1_t src, const int lane)	ptr -> Xn src -> Vt.1D lane == 0	LD1 {Vt.D}[lane],[Xn]	Vt.1D -> result	A32/A64
poly64x2_t vld1q_lane_p64(poly64_t const * ptr, poly64x2_t src, const int lane)	ptr -> Xn src -> Vt.2D 0 <= lane <= 1	LD1 {Vt.D}[lane],[Xn]	Vt.2D -> result	A32/A64
float16x4_t vld1_lane_f16(float16_t const * ptr, float16x4_t src, const int lane)	ptr -> Xn src -> Vt.4H 0 <= lane <= 3	LD1 {Vt.H}[lane],[Xn]	Vt.4H -> result	v7/A32/A64
float16x8_t vld1q_lane_f16(float16_t const * ptr, float16x8_t src, const int lane)	ptr -> Xn src -> Vt.8H 0 <= lane <= 7	LD1 {Vt.H}[lane],[Xn]	Vt.8H -> result	v7/A32/A64
float32x2_t vld1_lane_f32(float32_t const * ptr, float32x2_t src, const int lane)	ptr -> Xn src -> Vt.2S 0 <= lane <= 1	LD1 {Vt.S}[lane],[Xn]	Vt.2S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t vld1q_lane_f32(float32_t const * ptr, float32x4_t src, const int lane)	ptr -> Xn src -> Vt.4S 0 <= lane <= 3	LD1 {Vt.S}[lane],[Xn]	Vt.4S -> result	v7/A32/A64
poly8x8_t vld1_lane_p8(poly8_t const * ptr, poly8x8_t src, const int lane)	ptr -> Xn src -> Vt.8B 0 <= lane <= 7	LD1 {Vt.B}[lane],[Xn]	Vt.8B -> result	v7/A32/A64
poly8x16_t vld1q_lane_p8(poly8_t const * ptr, poly8x16_t src, const int lane)	ptr -> Xn src -> Vt.16B 0 <= lane <= 15	LD1 {Vt.B}[lane],[Xn]	Vt.16B -> result	v7/A32/A64
poly16x4_t vld1_lane_p16(poly16_t const * ptr, poly16x4_t src, const int lane)	ptr -> Xn src -> Vt.4H 0 <= lane <= 3	LD1 {Vt.H}[lane],[Xn]	Vt.4H -> result	v7/A32/A64
poly16x8_t vld1q_lane_p16(poly16_t const * ptr, poly16x8_t src, const int lane)	ptr -> Xn src -> Vt.8H 0 <= lane <= 7	LD1 {Vt.H}[lane],[Xn]	Vt.8H -> result	v7/A32/A64
float64x1_t vld1_lane_f64(float64_t const * ptr, float64x1_t src, const int lane)	ptr -> Xn src -> Vt.1D lane == 0	LD1 {Vt.D}[lane],[Xn]	Vt.1D -> result	A64
float64x2_t vld1q_lane_f64(float64_t const * ptr, float64x2_t src, const int lane)	ptr -> Xn src -> Vt.2D 0 <= lane <= 1	LD1 {Vt.D}[lane],[Xn]	Vt.2D -> result	A64
int8x8_t vld1_dup_s8(int8_t const * ptr)	ptr -> Xn	LD1R {Vt.8B},[Xn]	Vt.8B -> result	v7/A32/A64
int8x16_t vld1q_dup_s8(int8_t const * ptr)	ptr -> Xn	LD1R {Vt.16B},[Xn]	Vt.16B -> result	v7/A32/A64
int16x4_t vld1_dup_s16(int16_t const * ptr)	ptr -> Xn	LD1R {Vt.4H},[Xn]	Vt.4H -> result	v7/A32/A64
int16x8_t vld1q_dup_s16(int16_t const * ptr)	ptr -> Xn	LD1R {Vt.8H},[Xn]	Vt.8H -> result	v7/A32/A64
int32x2_t vld1_dup_s32(int32_t const * ptr)	ptr -> Xn	LD1R {Vt.2S},[Xn]	Vt.2S -> result	v7/A32/A64
int32x4_t vld1q_dup_s32(int32_t const * ptr)	ptr -> Xn	LD1R {Vt.4S},[Xn]	Vt.4S -> result	v7/A32/A64
int64x1_t vld1_dup_s64(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D},[Xn]	Vt.1D -> result	v7/A32/A64
int64x2_t vld1q_dup_s64(int64_t const * ptr)	ptr -> Xn	LD1R {Vt.2D},[Xn]	Vt.2D -> result	v7/A32/A64
uint8x8_t vld1_dup_u8(uint8_t const * ptr)	ptr -> Xn	LD1R {Vt.8B},[Xn]	Vt.8B -> result	v7/A32/A64
uint8x16_t vld1q_dup_u8(uint8_t const * ptr)	ptr -> Xn	LD1R {Vt.16B},[Xn]	Vt.16B -> result	v7/A32/A64
uint16x4_t vld1_dup_u16(uint16_t const * ptr)	ptr -> Xn	LD1R {Vt.4H},[Xn]	Vt.4H -> result	v7/A32/A64
uint16x8_t vld1q_dup_u16(uint16_t const * ptr)	ptr -> Xn	LD1R {Vt.8H},[Xn]	Vt.8H -> result	v7/A32/A64
uint32x2_t vld1_dup_u32(uint32_t const * ptr)	ptr -> Xn	LD1R {Vt.2S},[Xn]	Vt.2S -> result	v7/A32/A64
uint32x4_t vld1q_dup_u32(uint32_t const * ptr)	ptr -> Xn	LD1R {Vt.4S},[Xn]	Vt.4S -> result	v7/A32/A64
uint64x1_t vld1_dup_u64(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D},[Xn]	Vt.1D -> result	v7/A32/A64
uint64x2_t vld1q_dup_u64(uint64_t const * ptr)	ptr -> Xn	LD1R {Vt.2D},[Xn]	Vt.2D -> result	v7/A32/A64
poly64x1_t vld1_dup_p64(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D},[Xn]	Vt.1D -> result	A32/A64
poly64x2_t vld1q_dup_p64(poly64_t const * ptr)	ptr -> Xn	LD1R {Vt.2D},[Xn]	Vt.2D -> result	A32/A64
float16x4_t vld1_dup_f16(float16_t const * ptr)	ptr -> Xn	LD1R {Vt.4H},[Xn]	Vt.4H -> result	v7/A32/A64
float16x8_t vld1q_dup_f16(float16_t const * ptr)	ptr -> Xn	LD1R {Vt.8H},[Xn]	Vt.8H -> result	v7/A32/A64
float32x2_t vld1_dup_f32(float32_t const * ptr)	ptr -> Xn	LD1R {Vt.2S},[Xn]	Vt.2S -> result	v7/A32/A64
float32x4_t vld1q_dup_f32(float32_t const * ptr)	ptr -> Xn	LD1R {Vt.4S},[Xn]	Vt.4S -> result	v7/A32/A64
poly8x8_t vld1_dup_p8(poly8_t const * ptr)	ptr -> Xn	LD1R {Vt.8B},[Xn]	Vt.8B -> result	v7/A32/A64
poly8x16_t vld1q_dup_p8(poly8_t const * ptr)	ptr -> Xn	LD1R {Vt.16B},[Xn]	Vt.16B -> result	v7/A32/A64
poly16x4_t vld1_dup_p16(poly16_t const * ptr)	ptr -> Xn	LD1R {Vt.4H},[Xn]	Vt.4H -> result	v7/A32/A64
poly16x8_t vld1q_dup_p16(poly16_t const * ptr)	ptr -> Xn	LD1R {Vt.8H},[Xn]	Vt.8H -> result	v7/A32/A64
float64x1_t vld1_dup_f64(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D},[Xn]	Vt.1D -> result	A64
float64x2_t vld1q_dup_f64(float64_t const * ptr)	ptr -> Xn	LD1R {Vt.2D},[Xn]	Vt.2D -> result	A64
void vst1_s8(int8_t * ptr, int8x8_t val)	ptr -> Xn val -> Vt.8B	ST1 {Vt.8B},[Xn]	void -> result	v7/A32/A64
void vst1q_s8(int8_t * ptr, int8x16_t val)	ptr -> Xn val -> Vt.16B	ST1 {Vt.16B},[Xn]	void -> result	v7/A32/A64
void vst1_s16(int16_t * ptr, int16x4_t val)	ptr -> Xn val -> Vt.4H	ST1 {Vt.4H},[Xn]	void -> result	v7/A32/A64
void vst1q_s16(int16_t * ptr, int16x8_t val)	ptr -> Xn val -> Vt.8H	ST1 {Vt.8H},[Xn]	void -> result	v7/A32/A64
void vst1_s32(int32_t * ptr, int32x2_t val)	ptr -> Xn val -> Vt.2S	ST1 {Vt.2S},[Xn]	void -> result	v7/A32/A64
void vst1q_s32(int32_t * ptr, int32x4_t val)	ptr -> Xn val -> Vt.4S	ST1 {Vt.4S},[Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void vst1_s64(int64_t * ptr, int64x1_t val)	ptr -> Xn val -> Vt.1D	ST1 {Vt.1D},[Xn]	void -> result	v7/A32/A64
void vst1q_s64(int64_t * ptr, int64x2_t val)	ptr -> Xn val -> Vt.2D	ST1 {Vt.2D},[Xn]	void -> result	v7/A32/A64
void vst1_u8(uint8_t * ptr, uint8x8_t val)	ptr -> Xn val -> Vt.8B	ST1 {Vt.8B},[Xn]	void -> result	v7/A32/A64
void vst1q_u8(uint8_t * ptr, uint8x16_t val)	ptr -> Xn val -> Vt.16B	ST1 {Vt.16B},[Xn]	void -> result	v7/A32/A64
void vst1_u16(uint16_t * ptr, uint16x4_t val)	ptr -> Xn val -> Vt.4H	ST1 {Vt.4H},[Xn]	void -> result	v7/A32/A64
void vst1q_u16(uint16_t * ptr, uint16x8_t val)	ptr -> Xn val -> Vt.8H	ST1 {Vt.8H},[Xn]	void -> result	v7/A32/A64
void vst1_u32(uint32_t * ptr, uint32x2_t val)	ptr -> Xn val -> Vt.2S	ST1 {Vt.2S},[Xn]	void -> result	v7/A32/A64
void vst1q_u32(uint32_t * ptr, uint32x4_t val)	ptr -> Xn val -> Vt.4S	ST1 {Vt.4S},[Xn]	void -> result	v7/A32/A64
void vst1_u64(uint64_t * ptr, uint64x1_t val)	ptr -> Xn val -> Vt.1D	ST1 {Vt.1D},[Xn]	void -> result	v7/A32/A64
void vst1q_u64(uint64_t * ptr, uint64x2_t val)	ptr -> Xn val -> Vt.2D	ST1 {Vt.2D},[Xn]	void -> result	v7/A32/A64
void vst1_p64(poly64_t * ptr, poly64x1_t val)	ptr -> Xn val -> Vt.1D	ST1 {Vt.1D},[Xn]	void -> result	A32/A64
void vst1q_p64(poly64_t * ptr, poly64x2_t val)	ptr -> Xn val -> Vt.2D	ST1 {Vt.2D},[Xn]	void -> result	A32/A64
void vst1_f16(float16_t * ptr, float16x4_t val)	ptr -> Xn val -> Vt.4H	ST1 {Vt.4H},[Xn]	void -> result	v7/A32/A64
void vst1q_f16(float16_t * ptr, float16x8_t val)	ptr -> Xn val -> Vt.8H	ST1 {Vt.8H},[Xn]	void -> result	v7/A32/A64
void vst1_f32(float32_t * ptr, float32x2_t val)	ptr -> Xn val -> Vt.2S	ST1 {Vt.2S},[Xn]	void -> result	v7/A32/A64
void vst1q_f32(float32_t * ptr, float32x4_t val)	ptr -> Xn val -> Vt.4S	ST1 {Vt.4S},[Xn]	void -> result	v7/A32/A64
void vst1_p8(poly8_t * ptr, poly8x8_t val)	ptr -> Xn val -> Vt.8B	ST1 {Vt.8B},[Xn]	void -> result	v7/A32/A64
void vst1q_p8(poly8_t * ptr, poly8x16_t val)	ptr -> Xn val -> Vt.16B	ST1 {Vt.16B},[Xn]	void -> result	v7/A32/A64
void vst1_p16(poly16_t * ptr, poly16x4_t val)	ptr -> Xn val -> Vt.4H	ST1 {Vt.4H},[Xn]	void -> result	v7/A32/A64
void vst1q_p16(poly16_t * ptr, poly16x8_t val)	ptr -> Xn val -> Vt.8H	ST1 {Vt.8H},[Xn]	void -> result	v7/A32/A64
void vst1_f64(float64_t * ptr, float64x1_t val)	ptr -> Xn val -> Vt.1D	ST1 {Vt.1D},[Xn]	void -> result	A64
void vst1q_f64(float64_t * ptr, float64x2_t val)	ptr -> Xn val -> Vt.2D	ST1 {Vt.2D},[Xn]	void -> result	A64
void vst1_lane_s8(int8_t * ptr, int8x8_t val, const int lane)	ptr -> Xn val -> Vt.8B 0 <= lane <= 7	ST1 {Vt.b}[lane],[Xn]	void -> result	v7/A32/A64
void vst1q_lane_s8(int8_t * ptr, int8x16_t val, const int lane)	ptr -> Xn val -> Vt.16B 0 <= lane <= 15	ST1 {Vt.b}[lane],[Xn]	void -> result	v7/A32/A64
void vst1_lane_s16(int16_t * ptr, int16x4_t val, const int lane)	ptr -> Xn val -> Vt.4H 0 <= lane <= 3	ST1 {Vt.h}[lane],[Xn]	void -> result	v7/A32/A64
void vst1q_lane_s16(int16_t * ptr, int16x8_t val, const int lane)	ptr -> Xn val -> Vt.8H 0 <= lane <= 7	ST1 {Vt.h}[lane],[Xn]	void -> result	v7/A32/A64
void vst1_lane_s32(int32_t * ptr, int32x2_t val, const int lane)	ptr -> Xn val -> Vt.2S 0 <= lane <= 1	ST1 {Vt.s}[lane],[Xn]	void -> result	v7/A32/A64
void vst1q_lane_s32(int32_t * ptr, int32x4_t val, const int lane)	ptr -> Xn val -> Vt.4S 0 <= lane <= 3	ST1 {Vt.s}[lane],[Xn]	void -> result	v7/A32/A64
void vst1_lane_s64(int64_t * ptr, int64x1_t val, const int lane)	ptr -> Xn val -> Vt.1D lane == 0	ST1 {Vt.d}[lane],[Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void vst1q_lane_s64(int64_t * ptr, int64x2_t val, const int lane)	ptr -> Xn val -> Vt.2D 0 <= lane <= 1	ST1 {Vt.d}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1_lane_u8(uint8_t * ptr, uint8x8_t val, const int lane)	ptr -> Xn val -> Vt.8B 0 <= lane <= 7	ST1 {Vt.b}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1q_lane_u8(uint8_t * ptr, uint8x16_t val, const int lane)	ptr -> Xn val -> Vt.16B 0 <= lane <= 15	ST1 {Vt.b}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1_lane_u16(uint16_t * ptr, uint16x4_t val, const int lane)	ptr -> Xn val -> Vt.4H 0 <= lane <= 3	ST1 {Vt.h}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1q_lane_u16(uint16_t * ptr, uint16x8_t val, const int lane)	ptr -> Xn val -> Vt.8H 0 <= lane <= 7	ST1 {Vt.h}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1_lane_u32(uint32_t * ptr, uint32x2_t val, const int lane)	ptr -> Xn val -> Vt.2S 0 <= lane <= 1	ST1 {Vt.s}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1q_lane_u32(uint32_t * ptr, uint32x4_t val, const int lane)	ptr -> Xn val -> Vt.4S 0 <= lane <= 3	ST1 {Vt.s}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1_lane_u64(uint64_t * ptr, uint64x1_t val, const int lane)	ptr -> Xn val -> Vt.1D lane == 0	ST1 {Vt.d}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1q_lane_u64(uint64_t * ptr, uint64x2_t val, const int lane)	ptr -> Xn val -> Vt.2D 0 <= lane <= 1	ST1 {Vt.d}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1_lane_p64(poly64_t * ptr, poly64x1_t val, const int lane)	ptr -> Xn val -> Vt.1D lane == 0	ST1 {Vt.d}[,lane],[Xn]	void -> result	A32/A64
void vst1q_lane_p64(poly64_t * ptr, poly64x2_t val, const int lane)	ptr -> Xn val -> Vt.2D 0 <= lane <= 1	ST1 {Vt.d}[,lane],[Xn]	void -> result	A32/A64
void vst1_lane_f16(float16_t * ptr, float16x4_t val, const int lane)	ptr -> Xn val -> Vt.4H 0 <= lane <= 3	ST1 {Vt.h}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1q_lane_f16(float16_t * ptr, float16x8_t val, const int lane)	ptr -> Xn val -> Vt.8H 0 <= lane <= 7	ST1 {Vt.h}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1_lane_f32(float32_t * ptr, float32x2_t val, const int lane)	ptr -> Xn val -> Vt.2S 0 <= lane <= 1	ST1 {Vt.s}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1q_lane_f32(float32_t * ptr, float32x4_t val, const int lane)	ptr -> Xn val -> Vt.4S 0 <= lane <= 3	ST1 {Vt.s}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1_lane_p8(poly8_t * ptr, poly8x8_t val, const int lane)	ptr -> Xn val -> Vt.8B 0 <= lane <= 7	ST1 {Vt.b}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1q_lane_p8(poly8_t * ptr, poly8x16_t val, const int lane)	ptr -> Xn val -> Vt.16B 0 <= lane <= 15	ST1 {Vt.b}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1_lane_p16(poly16_t * ptr, poly16x4_t val, const int lane)	ptr -> Xn val -> Vt.4H 0 <= lane <= 3	ST1 {Vt.h}[,lane],[Xn]	void -> result	v7/A32/A64
void vst1q_lane_p16(poly16_t * ptr, poly16x8_t val, const int lane)	ptr -> Xn val -> Vt.8H 0 <= lane <= 7	ST1 {Vt.h}[,lane],[Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void vst1_lane_f64(float64_t * ptr, float64x1_t val, const int lane)	ptr -> Xn val -> Vt.1D lane == 0	ST1 {Vt.d}[lane],[Xn]	void -> result	A64
void vst1q_lane_f64(float64_t * ptr, float64x2_t val, const int lane)	ptr -> Xn val -> Vt.2D 0 <= lane <= 1	ST1 {Vt.d}[lane],[Xn]	void -> result	A64
int8x8x2_t vld2_s8(int8_t const * ptr)	ptr -> Xn	LD2 {Vt.8B - Vt.2.8B},[Xn]	Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
int8x16x2_t vld2q_s8(int8_t const * ptr)	ptr -> Xn	LD2 {Vt.16B - Vt.2.16B},[Xn]	Vt.2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
int16x4x2_t vld2_s16(int16_t const * ptr)	ptr -> Xn	LD2 {Vt.4H - Vt.2.4H},[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x2_t vld2q_s16(int16_t const * ptr)	ptr -> Xn	LD2 {Vt.8H - Vt.2.8H},[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int32x2x2_t vld2_s32(int32_t const * ptr)	ptr -> Xn	LD2 {Vt.2S - Vt.2.2S},[Xn]	Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
int32x4x2_t vld2q_s32(int32_t const * ptr)	ptr -> Xn	LD2 {Vt.4S - Vt.2.4S},[Xn]	Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint8x8x2_t vld2_u8(uint8_t const * ptr)	ptr -> Xn	LD2 {Vt.8B - Vt.2.8B},[Xn]	Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
uint8x16x2_t vld2q_u8(uint8_t const * ptr)	ptr -> Xn	LD2 {Vt.16B - Vt.2.16B},[Xn]	Vt.2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
uint16x4x2_t vld2_u16(uint16_t const * ptr)	ptr -> Xn	LD2 {Vt.4H - Vt.2.4H},[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
uint16x8x2_t vld2q_u16(uint16_t const * ptr)	ptr -> Xn	LD2 {Vt.8H - Vt.2.8H},[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
uint32x2x2_t vld2_u32(uint32_t const * ptr)	ptr -> Xn	LD2 {Vt.2S - Vt.2.2S},[Xn]	Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
uint32x4x2_t vld2q_u32(uint32_t const * ptr)	ptr -> Xn	LD2 {Vt.4S - Vt.2.4S},[Xn]	Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
float16x4x2_t vld2_f16(float16_t const * ptr)	ptr -> Xn	LD2 {Vt.4H - Vt.2.4H},[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
float16x8x2_t vld2q_f16(float16_t const * ptr)	ptr -> Xn	LD2 {Vt.8H - Vt.2.8H},[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
float32x2x2_t vld2_f32(float32_t const * ptr)	ptr -> Xn	LD2 {Vt.2S - Vt.2.2S},[Xn]	Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
float32x4x2_t vld2q_f32(float32_t const * ptr)	ptr -> Xn	LD2 {Vt.4S - Vt.2.4S},[Xn]	Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly8x8x2_t_vld2_p8(poly8_t const * ptr)	ptr -> Xn	LD2 {Vt.8B - Vt.2.8B},{Xn}	Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
poly8x16x2_t_vld2q_p8(poly8_t const * ptr)	ptr -> Xn	LD2 {Vt.16B - Vt.2.16B},{Xn}	Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
poly16x4x2_t_vld2_p16(poly16_t const * ptr)	ptr -> Xn	LD2 {Vt.4H - Vt.2.4H},{Xn}	Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
poly16x8x2_t_vld2q_p16(poly16_t const * ptr)	ptr -> Xn	LD2 {Vt.8H - Vt.2.8H},{Xn}	Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int64x1x2_t_vld2_s64(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.2.1D},{Xn}	Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
uint64x1x2_t_vld2_u64(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.2.1D},{Xn}	Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
poly64x1x2_t_vld2_p64(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.2.1D},{Xn}	Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A32/A64
int64x2x2_t_vld2q_s64(int64_t const * ptr)	ptr -> Xn	LD2 {Vt.2D - Vt.2.2D},{Xn}	Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
uint64x2x2_t_vld2q_u64(uint64_t const * ptr)	ptr -> Xn	LD2 {Vt.2D - Vt.2.2D},{Xn}	Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
poly64x2x2_t_vld2q_p64(poly64_t const * ptr)	ptr -> Xn	LD2 {Vt.2D - Vt.2.2D},{Xn}	Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
float64x1x2_t_vld2_f64(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.2.1D},{Xn}	Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
float64x2x2_t_vld2q_f64(float64_t const * ptr)	ptr -> Xn	LD2 {Vt.2D - Vt.2.2D},{Xn}	Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
int8x8x3_t_vld3_s8(int8_t const * ptr)	ptr -> Xn	LD3 {Vt.8B - Vt.3.8B},{Xn}	Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
int8x16x3_t_vld3q_s8(int8_t const * ptr)	ptr -> Xn	LD3 {Vt.16B - Vt.3.16B},{Xn}	Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
int16x4x3_t_vld3_s16(int16_t const * ptr)	ptr -> Xn	LD3 {Vt.4H - Vt.3.4H},{Xn}	Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x3_t_vld3q_s16(int16_t const * ptr)	ptr -> Xn	LD3 {Vt.8H - Vt.3.8H},{Xn}	Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x2x3_t_vld3_s32(int32_t const * ptr)	ptr -> Xn	LD3 {Vt.2S - Vt.3.2S},{Xn}	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
int32x4x3_t_vld3q_s32(int32_t const * ptr)	ptr -> Xn	LD3 {Vt.4S - Vt.3.4S},{Xn}	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint8x8x3_t_vld3_u8(uint8_t const * ptr)	ptr -> Xn	LD3 {Vt.8B - Vt.3.8B},{Xn}	Vt.3.8B -> result.val[2] Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
uint8x16x3_t_vld3q_u8(uint8_t const * ptr)	ptr -> Xn	LD3 {Vt.16B - Vt.3.16B},{Xn}	Vt.3.16B -> result.val[2] Vt.2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
uint16x4x3_t_vld3_u16(uint16_t const * ptr)	ptr -> Xn	LD3 {Vt.4H - Vt.3.4H},{Xn}	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
uint16x8x3_t_vld3q_u16(uint16_t const * ptr)	ptr -> Xn	LD3 {Vt.8H - Vt.3.8H},{Xn}	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
uint32x2x3_t_vld3_u32(uint32_t const * ptr)	ptr -> Xn	LD3 {Vt.2S - Vt.3.2S},{Xn}	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
uint32x4x3_t_vld3q_u32(uint32_t const * ptr)	ptr -> Xn	LD3 {Vt.4S - Vt.3.4S},{Xn}	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
float16x4x3_t_vld3_f16(float16_t const * ptr)	ptr -> Xn	LD3 {Vt.4H - Vt.3.4H},{Xn}	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
float16x8x3_t_vld3q_f16(float16_t const * ptr)	ptr -> Xn	LD3 {Vt.8H - Vt.3.8H},{Xn}	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
float32x2x3_t_vld3_f32(float32_t const * ptr)	ptr -> Xn	LD3 {Vt.2S - Vt.3.2S},{Xn}	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
float32x4x3_t_vld3q_f32(float32_t const * ptr)	ptr -> Xn	LD3 {Vt.4S - Vt.3.4S},{Xn}	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly8x8x3_t_vld3_p8(poly8_t const * ptr)	ptr -> Xn	LD3 {Vt.8B - Vt3.8B},{Xn}	Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
poly8x16x3_t_vld3_p8(poly8_t const * ptr)	ptr -> Xn	LD3 {Vt.16B - Vt3.16B},{Xn}	Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
poly16x4x3_t_vld3_p16(poly16_t const * ptr)	ptr -> Xn	LD3 {Vt.4H - Vt3.4H},{Xn}	Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
poly16x8x3_t_vld3_p16(poly16_t const * ptr)	ptr -> Xn	LD3 {Vt.8H - Vt3.8H},{Xn}	Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int64x1x3_t_vld3_s64(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt3.1D},{Xn}	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
uint64x1x3_t_vld3_u64(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt3.1D},{Xn}	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
poly64x1x3_t_vld3_p64(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt3.1D},{Xn}	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A32/A64
int64x2x3_t_vld3q_s64(int64_t const * ptr)	ptr -> Xn	LD3 {Vt.2D - Vt3.2D},{Xn}	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
uint64x2x3_t_vld3q_u64(uint64_t const * ptr)	ptr -> Xn	LD3 {Vt.2D - Vt3.2D},{Xn}	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
poly64x2x3_t_vld3q_p64(poly64_t const * ptr)	ptr -> Xn	LD3 {Vt.2D - Vt3.2D},{Xn}	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
float64x1x3_t_vld3_f64(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt3.1D},{Xn}	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
float64x2x3_t_vld3q_f64(float64_t const * ptr)	ptr -> Xn	LD3 {Vt.2D - Vt3.2D},{Xn}	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x8x4_t_vld4_s8(int8_t const * ptr)	ptr -> Xn	LD4 {Vt.8B - Vt4.8B},{Xn}	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
int8x16x4_t_vld4q_s8(int8_t const * ptr)	ptr -> Xn	LD4 {Vt.16B - Vt4.16B},{Xn}	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
int16x4x4_t_vld4_s16(int16_t const * ptr)	ptr -> Xn	LD4 {Vt.4H - Vt4.4H},{Xn}	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x4_t_vld4q_s16(int16_t const * ptr)	ptr -> Xn	LD4 {Vt.8H - Vt4.8H},{Xn}	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int32x2x4_t_vld4_s32(int32_t const * ptr)	ptr -> Xn	LD4 {Vt.2S - Vt4.2S},{Xn}	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
int32x4x4_t_vld4q_s32(int32_t const * ptr)	ptr -> Xn	LD4 {Vt.4S - Vt4.4S},{Xn}	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint8x8x4_t_vld4_u8(uint8_t const * ptr)	ptr -> Xn	LD4 {Vt.8B - Vt4.8B},{Xn}	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
uint8x16x4_t_vld4q_u8(uint8_t const * ptr)	ptr -> Xn	LD4 {Vt.16B - Vt4.16B},{Xn}	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
uint16x4x4_t_vld4_u16(uint16_t const * ptr)	ptr -> Xn	LD4 {Vt.4H - Vt4.4H},{Xn}	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8x4_t_vld4q_u16(uint16_t const * ptr)	ptr -> Xn	LD4 {Vt.8H - Vt.4.8H},[Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
uint32x2x4_t_vld4q_u32(uint32_t const * ptr)	ptr -> Xn	LD4 {Vt.2S - Vt.4.2S},[Xn]	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
uint32x4x4_t_vld4q_u32(uint32_t const * ptr)	ptr -> Xn	LD4 {Vt.4S - Vt.4.4S},[Xn]	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
float16x4x4_t_vld4_f16(float16_t const * ptr)	ptr -> Xn	LD4 {Vt.4H - Vt.4.4H},[Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
float16x8x4_t_vld4q_f16(float16_t const * ptr)	ptr -> Xn	LD4 {Vt.8H - Vt.4.8H},[Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
float32x2x4_t_vld4_f32(float32_t const * ptr)	ptr -> Xn	LD4 {Vt.2S - Vt.4.2S},[Xn]	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
float32x4x4_t_vld4q_f32(float32_t const * ptr)	ptr -> Xn	LD4 {Vt.4S - Vt.4.4S},[Xn]	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
poly8x8x4_t_vld4_p8(poly8_t const * ptr)	ptr -> Xn	LD4 {Vt.8B - Vt.4.8B},[Xn]	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
poly8x16x4_t_vld4q_p8(poly8_t const * ptr)	ptr -> Xn	LD4 {Vt.16B - Vt.4.16B},[Xn]	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly16x4x4_t_vld4_p16(poly16_t const * ptr)	ptr -> Xn	LD4 {Vt.4H - Vt.4.4H},{Xn}	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
poly16x8x4_t_vld4q_p16(poly16_t const * ptr)	ptr -> Xn	LD4 {Vt.8H - Vt.4.8H},{Xn}	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int64x1x4_t_vld4_s64(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.4.1D},{Xn}	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
uint64x1x4_t_vld4_u64(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.4.1D},{Xn}	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
poly64x1x4_t_vld4_p64(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.4.1D},{Xn}	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A32/A64
int64x2x4_t_vld4q_s64(int64_t const * ptr)	ptr -> Xn	LD4 {Vt.2D - Vt.4.2D},{Xn}	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
uint64x2x4_t_vld4q_u64(uint64_t const * ptr)	ptr -> Xn	LD4 {Vt.2D - Vt.4.2D},{Xn}	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
poly64x2x4_t_vld4q_p64(poly64_t const * ptr)	ptr -> Xn	LD4 {Vt.2D - Vt.4.2D},{Xn}	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
float64x1x4_t_vld4_f64(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.4.1D},{Xn}	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64x2x4_t_vld4q_f64(float64_t const * ptr)	ptr -> Xn	LD4 {Vt.2D - Vt.4.2D},[Xn]	Vt.4.2D -> result.val[3] Vt.3.2D -> result.val[2] Vt.2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
int8x8x2_t_vld2_dup_s8(int8_t const * ptr)	ptr -> Xn	LD2R {Vt.8B - Vt.2.8B},[Xn]	Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
int8x16x2_t_vld2q_dup_s8(int8_t const * ptr)	ptr -> Xn	LD2R {Vt.16B - Vt.2.16B},[Xn]	Vt.2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
int16x4x2_t_vld2_dup_s16(int16_t const * ptr)	ptr -> Xn	LD2R {Vt.4H - Vt.2.4H},[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x2_t_vld2q_dup_s16(int16_t const * ptr)	ptr -> Xn	LD2R {Vt.8H - Vt.2.8H},[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int32x2x2_t_vld2_dup_s32(int32_t const * ptr)	ptr -> Xn	LD2R {Vt.2S - Vt.2.2S},[Xn]	Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
int32x4x2_t_vld2q_dup_s32(int32_t const * ptr)	ptr -> Xn	LD2R {Vt.4S - Vt.2.4S},[Xn]	Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint8x8x2_t_vld2_dup_u8(uint8_t const * ptr)	ptr -> Xn	LD2R {Vt.8B - Vt.2.8B},[Xn]	Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
uint8x16x2_t_vld2q_dup_u8(uint8_t const * ptr)	ptr -> Xn	LD2R {Vt.16B - Vt.2.16B},[Xn]	Vt.2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
uint16x4x2_t_vld2_dup_u16(uint16_t const * ptr)	ptr -> Xn	LD2R {Vt.4H - Vt.2.4H},[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
uint16x8x2_t_vld2q_dup_u16(uint16_t const * ptr)	ptr -> Xn	LD2R {Vt.8H - Vt.2.8H},[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
uint32x2x2_t_vld2_dup_u32(uint32_t const * ptr)	ptr -> Xn	LD2R {Vt.2S - Vt.2.2S},[Xn]	Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
uint32x4x2_t_vld2q_dup_u32(uint32_t const * ptr)	ptr -> Xn	LD2R {Vt.4S - Vt.2.4S},[Xn]	Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
float16x4x2_t_vld2_dup_f16(float16_t const * ptr)	ptr -> Xn	LD2R {Vt.4H - Vt.2.4H},[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
float16x8x2_t_vld2q_dup_f16(float16_t const * ptr)	ptr -> Xn	LD2R {Vt.8H - Vt.2.8H},[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
float32x2x2_t_vld2_dup_f32(float32_t const * ptr)	ptr -> Xn	LD2R {Vt.2S - Vt.2.2S},[Xn]	Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
float32x4x2_t_vld2q_dup_f32(float32_t const * ptr)	ptr -> Xn	LD2R {Vt.4S - Vt.2.4S},[Xn]	Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly8x8x2_t_vld2_dup_p8(poly8_t const * ptr)	ptr -> Xn	LD2R {Vt.8B - Vt.2.8B},[Xn]	Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
poly8x16x2_t_vld2q_dup_p8(poly8_t const * ptr)	ptr -> Xn	LD2R {Vt.16B - Vt.2.16B},[Xn]	Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
poly16x4x2_t_vld2_dup_p16(poly16_t const * ptr)	ptr -> Xn	LD2R {Vt.4H - Vt.2.4H},[Xn]	Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
poly16x8x2_t_vld2q_dup_p16(poly16_t const * ptr)	ptr -> Xn	LD2R {Vt.8H - Vt.2.8H},[Xn]	Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int64x1x2_t_vld2_dup_s64(int64_t const * ptr)	ptr -> Xn	LD2R {Vt.1D - Vt.2.1D},[Xn]	Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
uint64x1x2_t_vld2_dup_u64(uint64_t const * ptr)	ptr -> Xn	LD2R {Vt.1D - Vt.2.1D},[Xn]	Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
poly64x1x2_t_vld2_dup_p64(poly64_t const * ptr)	ptr -> Xn	LD2R {Vt.1D - Vt.2.1D},[Xn]	Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A32/A64
int64x2x2_t_vld2q_dup_s64(int64_t const * ptr)	ptr -> Xn	LD2R {Vt.2D - Vt.2.2D},[Xn]	Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
uint64x2x2_t_vld2q_dup_u64(uint64_t const * ptr)	ptr -> Xn	LD2R {Vt.2D - Vt.2.2D},[Xn]	Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
poly64x2x2_t_vld2q_dup_p64(poly64_t const * ptr)	ptr -> Xn	LD2R {Vt.2D - Vt.2.2D},[Xn]	Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
float64x1x2_t_vld2_dup_f64(float64_t const * ptr)	ptr -> Xn	LD2R {Vt.1D - Vt.2.1D},[Xn]	Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
float64x2x2_t_vld2q_dup_f64(float64_t const * ptr)	ptr -> Xn	LD2R {Vt.2D - Vt.2.2D},[Xn]	Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
int8x8x3_t_vld3_dup_s8(int8_t const * ptr)	ptr -> Xn	LD3R {Vt.8B - Vt.3.8B},[Xn]	Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
int8x16x3_t_vld3q_dup_s8(int8_t const * ptr)	ptr -> Xn	LD3R {Vt.16B - Vt.3.16B},[Xn]	Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
int16x4x3_t_vld3_dup_s16(int16_t const * ptr)	ptr -> Xn	LD3R {Vt.4H - Vt.3.4H},[Xn]	Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x3_t_vld3q_dup_s16(int16_t const * ptr)	ptr -> Xn	LD3R {Vt.8H - Vt.3.8H},[Xn]	Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x2x3_t_vld3_dup_s32(int32_t const * ptr)	ptr -> Xn	LD3R {Vt.2S - Vt.3.2S},[Xn]	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
int32x4x3_t_vld3q_dup_s32(int32_t const * ptr)	ptr -> Xn	LD3R {Vt.4S - Vt.3.4S},[Xn]	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint8x8x3_t_vld3_dup_u8(uint8_t const * ptr)	ptr -> Xn	LD3R {Vt.8B - Vt.3.8B},[Xn]	Vt.3.8B -> result.val[2] Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
uint8x16x3_t_vld3q_dup_u8(uint8_t const * ptr)	ptr -> Xn	LD3R {Vt.16B - Vt.3.16B},[Xn]	Vt.3.16B -> result.val[2] Vt.2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
uint16x4x3_t_vld3_dup_u16(uint16_t const * ptr)	ptr -> Xn	LD3R {Vt.4H - Vt.3.4H},[Xn]	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
uint16x8x3_t_vld3q_dup_u16(uint16_t const * ptr)	ptr -> Xn	LD3R {Vt.8H - Vt.3.8H},[Xn]	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
uint32x2x3_t_vld3_dup_u32(uint32_t const * ptr)	ptr -> Xn	LD3R {Vt.2S - Vt.3.2S},[Xn]	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
uint32x4x3_t_vld3q_dup_u32(uint32_t const * ptr)	ptr -> Xn	LD3R {Vt.4S - Vt.3.4S},[Xn]	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
float16x4x3_t_vld3_dup_f16(float16_t const * ptr)	ptr -> Xn	LD3R {Vt.4H - Vt.3.4H},[Xn]	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
float16x8x3_t_vld3q_dup_f16(float16_t const * ptr)	ptr -> Xn	LD3R {Vt.8H - Vt.3.8H},[Xn]	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
float32x2x3_t_vld3_dup_f32(float32_t const * ptr)	ptr -> Xn	LD3R {Vt.2S - Vt.3.2S},[Xn]	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
float32x4x3_t_vld3q_dup_f32(float32_t const * ptr)	ptr -> Xn	LD3R {Vt.4S - Vt.3.4S},[Xn]	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly8x8x3_t_vld3_dup_p8(poly8_t const * ptr)	ptr -> Xn	LD3R {Vt.8B - Vt.3.8B},[Xn]	Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
poly8x16x3_t_vld3q_dup_p8(poly8_t const * ptr)	ptr -> Xn	LD3R {Vt.16B - Vt.3.16B},[Xn]	Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
poly16x4x3_t_vld3_dup_p16(poly16_t const * ptr)	ptr -> Xn	LD3R {Vt.4H - Vt.3.4H},[Xn]	Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
poly16x8x3_t_vld3q_dup_p16(poly16_t const * ptr)	ptr -> Xn	LD3R {Vt.8H - Vt.3.8H},[Xn]	Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int64x1x3_t_vld3_dup_s64(int64_t const * ptr)	ptr -> Xn	LD3R {Vt.1D - Vt.3.1D},[Xn]	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
uint64x1x3_t_vld3_dup_u64(uint64_t const * ptr)	ptr -> Xn	LD3R {Vt.1D - Vt.3.1D},[Xn]	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
poly64x1x3_t_vld3_dup_p64(poly64_t const * ptr)	ptr -> Xn	LD3R {Vt.1D - Vt.3.1D},[Xn]	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A32/A64
int64x2x3_t_vld3q_dup_s64(int64_t const * ptr)	ptr -> Xn	LD3R {Vt.2D - Vt.3.2D},[Xn]	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
uint64x2x3_t_vld3q_dup_u64(uint64_t const * ptr)	ptr -> Xn	LD3R {Vt.2D - Vt.3.2D},[Xn]	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
poly64x2x3_t_vld3q_dup_p64(poly64_t const * ptr)	ptr -> Xn	LD3R {Vt.2D - Vt.3.2D},[Xn]	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
float64x1x3_t_vld3_dup_f64(float64_t const * ptr)	ptr -> Xn	LD3R {Vt.1D - Vt.3.1D},[Xn]	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
float64x2x3_t_vld3q_dup_f64(float64_t const * ptr)	ptr -> Xn	LD3R {Vt.2D - Vt.3.2D},[Xn]	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x8x4_t_vld4_dup_s8(int8_t const * ptr)	ptr -> Xn	LD4R {Vt.8B - Vt4.8B},[Xn]	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
int8x16x4_t_vld4q_dup_s8(int8_t const * ptr)	ptr -> Xn	LD4R {Vt.16B - Vt4.16B},[Xn]	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
int16x4x4_t_vld4_dup_s16(int16_t const * ptr)	ptr -> Xn	LD4R {Vt.4H - Vt4.4H},[Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x4_t_vld4q_dup_s16(int16_t const * ptr)	ptr -> Xn	LD4R {Vt.8H - Vt4.8H},[Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int32x2x4_t_vld4_dup_s32(int32_t const * ptr)	ptr -> Xn	LD4R {Vt.2S - Vt4.2S},[Xn]	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
int32x4x4_t_vld4q_dup_s32(int32_t const * ptr)	ptr -> Xn	LD4R {Vt.4S - Vt4.4S},[Xn]	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint8x8x4_t_vld4_dup_u8(uint8_t const * ptr)	ptr -> Xn	LD4R {Vt.8B - Vt4.8B},[Xn]	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
uint8x16x4_t_vld4q_dup_u8(uint8_t const * ptr)	ptr -> Xn	LD4R {Vt.16B - Vt4.16B},[Xn]	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
uint16x4x4_t_vld4_dup_u16(uint16_t const * ptr)	ptr -> Xn	LD4R {Vt.4H - Vt4.4H},[Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8x4_t_vld4q_dup_u16(uint16_t const * ptr)	ptr -> Xn	LD4R {Vt.8H - Vt.4.8H},[Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
uint32x2x4_t_vld4q_dup_u32(uint32_t const * ptr)	ptr -> Xn	LD4R {Vt.2S - Vt.4.2S},[Xn]	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
uint32x4x4_t_vld4q_dup_u32(uint32_t const * ptr)	ptr -> Xn	LD4R {Vt.4S - Vt.4.4S},[Xn]	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
float16x4x4_t_vld4q_dup_f16(float16_t const * ptr)	ptr -> Xn	LD4R {Vt.4H - Vt.4.4H},[Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
float16x8x4_t_vld4q_dup_f16(float16_t const * ptr)	ptr -> Xn	LD4R {Vt.8H - Vt.4.8H},[Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
float32x2x4_t_vld4q_dup_f32(float32_t const * ptr)	ptr -> Xn	LD4R {Vt.2S - Vt.4.2S},[Xn]	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
float32x4x4_t_vld4q_dup_f32(float32_t const * ptr)	ptr -> Xn	LD4R {Vt.4S - Vt.4.4S},[Xn]	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
poly8x8x4_t_vld4q_dup_p8(poly8_t const * ptr)	ptr -> Xn	LD4R {Vt.8B - Vt.4.8B},[Xn]	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
poly8x16x4_t_vld4q_dup_p8(poly8_t const * ptr)	ptr -> Xn	LD4R {Vt.16B - Vt.4.16B},[Xn]	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly16x4x4_t vld4_dup_p16(poly16_t const * ptr)	ptr -> Xn	LD4R {Vt.4H - Vt.4.4H},[Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
poly16x8x4_t vld4q_dup_p16(poly16_t const * ptr)	ptr -> Xn	LD4R {Vt.8H - Vt.4.8H},[Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int64x1x4_t vld4_dup_s64(int64_t const * ptr)	ptr -> Xn	LD4R {Vt.1D - Vt.4.1D},[Xn]	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
uint64x1x4_t vld4_dup_u64(uint64_t const * ptr)	ptr -> Xn	LD4R {Vt.1D - Vt.4.1D},[Xn]	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
poly64x1x4_t vld4_dup_p64(poly64_t const * ptr)	ptr -> Xn	LD4R {Vt.1D - Vt.4.1D},[Xn]	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A32/A64
int64x2x4_t vld4q_dup_s64(int64_t const * ptr)	ptr -> Xn	LD4R {Vt.2D - Vt.4.2D},[Xn]	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
uint64x2x4_t vld4q_dup_u64(uint64_t const * ptr)	ptr -> Xn	LD4R {Vt.2D - Vt.4.2D},[Xn]	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
poly64x2x4_t vld4q_dup_p64(poly64_t const * ptr)	ptr -> Xn	LD4R {Vt.2D - Vt.4.2D},[Xn]	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
float64x1x4_t vld4_dup_f64(float64_t const * ptr)	ptr -> Xn	LD4R {Vt.1D - Vt.4.1D},[Xn]	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64x2x4_t vld4q_dup_f64(float64_t const * ptr)	ptr -> Xn	LD4R {Vt.2D - Vt.4.2D},[Xn]	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
void vst2_s8(int8_t * ptr, int8x8x2_t val)	ptr -> Xn val.val[1] -> Vt2.8B val.val[0] -> Vt.8B	ST2 {Vt.8B - Vt.2.8B},[Xn]	void -> result	v7/A32/A64
void vst2q_s8(int8_t * ptr, int8x16x2_t val)	ptr -> Xn val.val[1] -> Vt2.16B val.val[0] -> Vt.16B	ST2 {Vt.16B - Vt.2.16B},[Xn]	void -> result	v7/A32/A64
void vst2_s16(int16_t * ptr, int16x4x2_t val)	ptr -> Xn val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST2 {Vt.4H - Vt.2.4H},[Xn]	void -> result	v7/A32/A64
void vst2q_s16(int16_t * ptr, int16x8x2_t val)	ptr -> Xn val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST2 {Vt.8H - Vt.2.8H},[Xn]	void -> result	v7/A32/A64
void vst2_s32(int32_t * ptr, int32x2x2_t val)	ptr -> Xn val.val[1] -> Vt2.2S val.val[0] -> Vt.2S	ST2 {Vt.2S - Vt.2.2S},[Xn]	void -> result	v7/A32/A64
void vst2q_s32(int32_t * ptr, int32x4x2_t val)	ptr -> Xn val.val[1] -> Vt2.4S val.val[0] -> Vt.4S	ST2 {Vt.4S - Vt.2.4S},[Xn]	void -> result	v7/A32/A64
void vst2_u8(uint8_t * ptr, uint8x8x2_t val)	ptr -> Xn val.val[1] -> Vt2.8B val.val[0] -> Vt.8B	ST2 {Vt.8B - Vt.2.8B},[Xn]	void -> result	v7/A32/A64
void vst2q_u8(uint8_t * ptr, uint8x16x2_t val)	ptr -> Xn val.val[1] -> Vt2.16B val.val[0] -> Vt.16B	ST2 {Vt.16B - Vt.2.16B},[Xn]	void -> result	v7/A32/A64
void vst2_u16(uint16_t * ptr, uint16x4x2_t val)	ptr -> Xn val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST2 {Vt.4H - Vt.2.4H},[Xn]	void -> result	v7/A32/A64
void vst2q_u16(uint16_t * ptr, uint16x8x2_t val)	ptr -> Xn val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST2 {Vt.8H - Vt.2.8H},[Xn]	void -> result	v7/A32/A64
void vst2_u32(uint32_t * ptr, uint32x2x2_t val)	ptr -> Xn val.val[1] -> Vt2.2S val.val[0] -> Vt.2S	ST2 {Vt.2S - Vt.2.2S},[Xn]	void -> result	v7/A32/A64
void vst2q_u32(uint32_t * ptr, uint32x4x2_t val)	ptr -> Xn val.val[1] -> Vt2.4S val.val[0] -> Vt.4S	ST2 {Vt.4S - Vt.2.4S},[Xn]	void -> result	v7/A32/A64
void vst2_f16(float16_t * ptr, float16x4x2_t val)	ptr -> Xn val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST2 {Vt.4H - Vt.2.4H},[Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst2q_f16(float16_t * ptr, float16x8x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.8H val.val[0] -> Vt.8H</code>	<code>ST2 {Vt.8H - Vt.2.8H},[Xn]</code>	void -> result	v7/A32/A64
<code>void vst2_f32(float32_t * ptr, float32x2x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.2S val.val[0] -> Vt.2S</code>	<code>ST2 {Vt.2S - Vt.2.2S},[Xn]</code>	void -> result	v7/A32/A64
<code>void vst2q_f32(float32_t * ptr, float32x4x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.4S val.val[0] -> Vt.4S</code>	<code>ST2 {Vt.4S - Vt.2.4S},[Xn]</code>	void -> result	v7/A32/A64
<code>void vst2_p8(poly8_t * ptr, poly8x8x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.8B val.val[0] -> Vt.8B</code>	<code>ST2 {Vt.8B - Vt.2.8B},[Xn]</code>	void -> result	v7/A32/A64
<code>void vst2q_p8(poly8_t * ptr, poly8x16x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.16B val.val[0] -> Vt.16B</code>	<code>ST2 {Vt.16B - Vt.2.16B},[Xn]</code>	void -> result	v7/A32/A64
<code>void vst2_p16(poly16_t * ptr, poly16x4x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.4H val.val[0] -> Vt.4H</code>	<code>ST2 {Vt.4H - Vt.2.4H},[Xn]</code>	void -> result	v7/A32/A64
<code>void vst2q_p16(poly16_t * ptr, poly16x8x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.8H val.val[0] -> Vt.8H</code>	<code>ST2 {Vt.8H - Vt.2.8H},[Xn]</code>	void -> result	v7/A32/A64
<code>void vst2_s64(int64_t * ptr, int64x1x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.1D val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt.2.1D},[Xn]</code>	void -> result	v7/A32/A64
<code>void vst2_u64(uint64_t * ptr, uint64x1x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.1D val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt.2.1D},[Xn]</code>	void -> result	v7/A32/A64
<code>void vst2_p64(poly64_t * ptr, poly64x1x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.1D val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt.2.1D},[Xn]</code>	void -> result	A32/A64
<code>void vst2q_s64(int64_t * ptr, int64x2x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.2D val.val[0] -> Vt.2D</code>	<code>ST2 {Vt.2D - Vt.2.2D},[Xn]</code>	void -> result	A64
<code>void vst2q_u64(uint64_t * ptr, uint64x2x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.2D val.val[0] -> Vt.2D</code>	<code>ST2 {Vt.2D - Vt.2.2D},[Xn]</code>	void -> result	A64
<code>void vst2q_p64(poly64_t * ptr, poly64x2x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.2D val.val[0] -> Vt.2D</code>	<code>ST2 {Vt.2D - Vt.2.2D},[Xn]</code>	void -> result	A64
<code>void vst2_f64(float64_t * ptr, float64x1x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.1D val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt.2.1D},[Xn]</code>	void -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst2q_f64(float64_t * ptr, float64x2x2_t val)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.2.2D</code> <code>val.val[0] -> Vt.2D</code>	<code>ST2 {Vt.2D - Vt.2.2D}, [Xn]</code>	<code>void -> result</code>	A64
<code>void vst3_s8(int8_t * ptr, int8x8x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt.3.8B</code> <code>val.val[1] -> Vt.2.8B</code> <code>val.val[0] -> Vt.8B</code>	<code>ST3 {Vt.8B - Vt.3.8B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_s8(int8_t * ptr, int8x16x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt.3.16B</code> <code>val.val[1] -> Vt.2.16B</code> <code>val.val[0] -> Vt.16B</code>	<code>ST3 {Vt.16B - Vt.3.16B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_s16(int16_t * ptr, int16x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt.3.4H</code> <code>val.val[1] -> Vt.2.4H</code> <code>val.val[0] -> Vt.4H</code>	<code>ST3 {Vt.4H - Vt.3.4H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_s16(int16_t * ptr, int16x8x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt.3.8H</code> <code>val.val[1] -> Vt.2.8H</code> <code>val.val[0] -> Vt.8H</code>	<code>ST3 {Vt.8H - Vt.3.8H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_s32(int32_t * ptr, int32x2x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt.3.2S</code> <code>val.val[1] -> Vt.2.2S</code> <code>val.val[0] -> Vt.2S</code>	<code>ST3 {Vt.2S - Vt.3.2S}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_s32(int32_t * ptr, int32x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt.3.4S</code> <code>val.val[1] -> Vt.2.4S</code> <code>val.val[0] -> Vt.4S</code>	<code>ST3 {Vt.4S - Vt.3.4S}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_u8(uint8_t * ptr, uint8x8x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt.3.8B</code> <code>val.val[1] -> Vt.2.8B</code> <code>val.val[0] -> Vt.8B</code>	<code>ST3 {Vt.8B - Vt.3.8B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_u8(uint8_t * ptr, uint8x16x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt.3.16B</code> <code>val.val[1] -> Vt.2.16B</code> <code>val.val[0] -> Vt.16B</code>	<code>ST3 {Vt.16B - Vt.3.16B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_u16(uint16_t * ptr, uint16x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt.3.4H</code> <code>val.val[1] -> Vt.2.4H</code> <code>val.val[0] -> Vt.4H</code>	<code>ST3 {Vt.4H - Vt.3.4H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst3q_u16(uint16_t * ptr, uint16x8x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.8H</code> <code>val.val[1] -> Vt2.8H</code> <code>val.val[0] -> Vt.8H</code>	<code>ST3 {Vt.8H - Vt3.8H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_u32(uint32_t * ptr, uint32x2x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.2S</code> <code>val.val[1] -> Vt2.2S</code> <code>val.val[0] -> Vt.2S</code>	<code>ST3 {Vt.2S - Vt3.2S}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_u32(uint32_t * ptr, uint32x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.4S</code> <code>val.val[1] -> Vt2.4S</code> <code>val.val[0] -> Vt.4S</code>	<code>ST3 {Vt.4S - Vt3.4S}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_f16(float16_t * ptr, float16x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.4H</code> <code>val.val[1] -> Vt2.4H</code> <code>val.val[0] -> Vt.4H</code>	<code>ST3 {Vt.4H - Vt3.4H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_f16(float16_t * ptr, float16x8x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.8H</code> <code>val.val[1] -> Vt2.8H</code> <code>val.val[0] -> Vt.8H</code>	<code>ST3 {Vt.8H - Vt3.8H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_f32(float32_t * ptr, float32x2x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.2S</code> <code>val.val[1] -> Vt2.2S</code> <code>val.val[0] -> Vt.2S</code>	<code>ST3 {Vt.2S - Vt3.2S}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_f32(float32_t * ptr, float32x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.4S</code> <code>val.val[1] -> Vt2.4S</code> <code>val.val[0] -> Vt.4S</code>	<code>ST3 {Vt.4S - Vt3.4S}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_p8(poly8_t * ptr, poly8x8x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.8B</code> <code>val.val[1] -> Vt2.8B</code> <code>val.val[0] -> Vt.8B</code>	<code>ST3 {Vt.8B - Vt3.8B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_p8(poly8_t * ptr, poly8x16x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.16B</code> <code>val.val[1] -> Vt2.16B</code> <code>val.val[0] -> Vt.16B</code>	<code>ST3 {Vt.16B - Vt3.16B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_p16(poly16_t * ptr, poly16x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.4H</code> <code>val.val[1] -> Vt2.4H</code> <code>val.val[0] -> Vt.4H</code>	<code>ST3 {Vt.4H - Vt3.4H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void vst3q_p16(poly16_t * ptr, poly16x8x3_t val)	ptr -> Xn val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST3 {Vt.8H - Vt3.8H},[Xn]	void -> result	v7/A32/A64
void vst3_s64(int64_t * ptr, int64x1x3_t val)	ptr -> Xn val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D	ST1 {Vt.1D - Vt3.1D},[Xn]	void -> result	v7/A32/A64
void vst3_u64(uint64_t * ptr, uint64x1x3_t val)	ptr -> Xn val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D	ST1 {Vt.1D - Vt3.1D},[Xn]	void -> result	v7/A32/A64
void vst3_p64(poly64_t * ptr, poly64x1x3_t val)	ptr -> Xn val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D	ST1 {Vt.1D - Vt3.1D},[Xn]	void -> result	A32/A64
void vst3q_s64(int64_t * ptr, int64x2x3_t val)	ptr -> Xn val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D	ST3 {Vt.2D - Vt3.2D},[Xn]	void -> result	A64
void vst3q_u64(uint64_t * ptr, uint64x2x3_t val)	ptr -> Xn val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D	ST3 {Vt.2D - Vt3.2D},[Xn]	void -> result	A64
void vst3q_p64(poly64_t * ptr, poly64x2x3_t val)	ptr -> Xn val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D	ST3 {Vt.2D - Vt3.2D},[Xn]	void -> result	A64
void vst3_f64(float64_t * ptr, float64x1x3_t val)	ptr -> Xn val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D	ST1 {Vt.1D - Vt3.1D},[Xn]	void -> result	A64
void vst3q_f64(float64_t * ptr, float64x2x3_t val)	ptr -> Xn val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D	ST3 {Vt.2D - Vt3.2D},[Xn]	void -> result	A64
void vst4_s8(int8_t * ptr, int8x8x4_t val)	ptr -> Xn val.val[3] -> Vt4.8B val.val[2] -> Vt3.8B val.val[1] -> Vt2.8B val.val[0] -> Vt.8B	ST4 {Vt.8B - Vt4.8B},[Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst4q_s8(int8_t * ptr, int8x16x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.16B val.val[2] -> Vt3.16B val.val[1] -> Vt2.16B val.val[0] -> Vt.16B	ST4 {Vt.16B - Vt4.16B}, [Xn]	void -> result	v7/A32/A64
<code>void vst4_s16(int16_t * ptr, int16x4x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST4 {Vt.4H - Vt4.4H}, [Xn]	void -> result	v7/A32/A64
<code>void vst4q_s16(int16_t * ptr, int16x8x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST4 {Vt.8H - Vt4.8H}, [Xn]	void -> result	v7/A32/A64
<code>void vst4_s32(int32_t * ptr, int32x2x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.2S val.val[2] -> Vt3.2S val.val[1] -> Vt2.2S val.val[0] -> Vt.2S	ST4 {Vt.2S - Vt4.2S}, [Xn]	void -> result	v7/A32/A64
<code>void vst4q_s32(int32_t * ptr, int32x4x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.4S val.val[2] -> Vt3.4S val.val[1] -> Vt2.4S val.val[0] -> Vt.4S	ST4 {Vt.4S - Vt4.4S}, [Xn]	void -> result	v7/A32/A64
<code>void vst4_u8(uint8_t * ptr, uint8x8x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.8B val.val[2] -> Vt3.8B val.val[1] -> Vt2.8B val.val[0] -> Vt.8B	ST4 {Vt.8B - Vt4.8B}, [Xn]	void -> result	v7/A32/A64
<code>void vst4q_u8(uint8_t * ptr, uint8x16x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.16B val.val[2] -> Vt3.16B val.val[1] -> Vt2.16B val.val[0] -> Vt.16B	ST4 {Vt.16B - Vt4.16B}, [Xn]	void -> result	v7/A32/A64
<code>void vst4_u16(uint16_t * ptr, uint16x4x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST4 {Vt.4H - Vt4.4H}, [Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void vst4q_u16(uint16_t * ptr, uint16x8x4_t val)	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST4 {Vt.8H - Vt4.8H},[Xn]	void -> result	v7/A32/A64
void vst4_u32(uint32_t * ptr, uint32x2x4_t val)	ptr -> Xn val.val[3] -> Vt4.2S val.val[2] -> Vt3.2S val.val[1] -> Vt2.2S val.val[0] -> Vt.2S	ST4 {Vt.2S - Vt4.2S},[Xn]	void -> result	v7/A32/A64
void vst4q_u32(uint32_t * ptr, uint32x4x4_t val)	ptr -> Xn val.val[3] -> Vt4.4S val.val[2] -> Vt3.4S val.val[1] -> Vt2.4S val.val[0] -> Vt.4S	ST4 {Vt.4S - Vt4.4S},[Xn]	void -> result	v7/A32/A64
void vst4_f16(float16_t * ptr, float16x4x4_t val)	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST4 {Vt.4H - Vt4.4H},[Xn]	void -> result	v7/A32/A64
void vst4q_f16(float16_t * ptr, float16x8x4_t val)	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST4 {Vt.8H - Vt4.8H},[Xn]	void -> result	v7/A32/A64
void vst4_f32(float32_t * ptr, float32x2x4_t val)	ptr -> Xn val.val[3] -> Vt4.2S val.val[2] -> Vt3.2S val.val[1] -> Vt2.2S val.val[0] -> Vt.2S	ST4 {Vt.2S - Vt4.2S},[Xn]	void -> result	v7/A32/A64
void vst4q_f32(float32_t * ptr, float32x4x4_t val)	ptr -> Xn val.val[3] -> Vt4.4S val.val[2] -> Vt3.4S val.val[1] -> Vt2.4S val.val[0] -> Vt.4S	ST4 {Vt.4S - Vt4.4S},[Xn]	void -> result	v7/A32/A64
void vst4_p8(poly8_t * ptr, poly8x8x4_t val)	ptr -> Xn val.val[3] -> Vt4.8B val.val[2] -> Vt3.8B val.val[1] -> Vt2.8B val.val[0] -> Vt.8B	ST4 {Vt.8B - Vt4.8B},[Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst4q_p8(poly8_t * ptr, poly8x16x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.16B val.val[2] -> Vt3.16B val.val[1] -> Vt2.16B val.val[0] -> Vt.16B	ST4 {Vt.16B - Vt4.16B},[Xn]	void -> result	v7/A32/A64
<code>void vst4_p16(poly16_t * ptr, poly16x4x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST4 {Vt.4H - Vt4.4H},[Xn]	void -> result	v7/A32/A64
<code>void vst4q_p16(poly16_t * ptr, poly16x8x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST4 {Vt.8H - Vt4.8H},[Xn]	void -> result	v7/A32/A64
<code>void vst4_s64(int64_t * ptr, int64x1x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.1D val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D	ST1 {Vt.1D - Vt4.1D},[Xn]	void -> result	v7/A32/A64
<code>void vst4_u64(uint64_t * ptr, uint64x1x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.1D val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D	ST1 {Vt.1D - Vt4.1D},[Xn]	void -> result	v7/A32/A64
<code>void vst4_p64(poly64_t * ptr, poly64x1x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.1D val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D	ST1 {Vt.1D - Vt4.1D},[Xn]	void -> result	A32/A64
<code>void vst4q_s64(int64_t * ptr, int64x2x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.2D val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D	ST4 {Vt.2D - Vt4.2D},[Xn]	void -> result	A64
<code>void vst4q_u64(uint64_t * ptr, uint64x2x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.2D val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D	ST4 {Vt.2D - Vt4.2D},[Xn]	void -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void vst4q_p64(poly64_t * ptr, poly64x2x4_t val)	ptr -> Xn val.val[3] -> Vt4.2D val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D	ST4 {Vt.2D - Vt4.2D},[Xn]	void -> result	A64
void vst4_f64(float64_t * ptr, float64x1x4_t val)	ptr -> Xn val.val[3] -> Vt4.1D val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D	ST1 {Vt.1D - Vt4.1D},[Xn]	void -> result	A64
void vst4q_f64(float64_t * ptr, float64x2x4_t val)	ptr -> Xn val.val[3] -> Vt4.2D val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D	ST4 {Vt.2D - Vt4.2D},[Xn]	void -> result	A64
int16x4x2_t vld2_lane_s16(int16_t const * ptr, int16x4x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD2 {Vt.h - Vt2.h}[lane],[Xn]	Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x2_t vld2q_lane_s16(int16_t const * ptr, int16x8x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD2 {Vt.h - Vt2.h}[lane],[Xn]	Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int32x2x2_t vld2_lane_s32(int32_t const * ptr, int32x2x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.2S src.val[0] -> Vt.2S 0 <= lane <= 1	LD2 {Vt.s - Vt2.s}[lane],[Xn]	Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
int32x4x2_t vld2q_lane_s32(int32_t const * ptr, int32x4x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.4S src.val[0] -> Vt.4S 0 <= lane <= 3	LD2 {Vt.s - Vt2.s}[lane],[Xn]	Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint16x4x2_t vld2_lane_u16(uint16_t const * ptr, uint16x4x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD2 {Vt.h - Vt2.h}[lane],[Xn]	Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
uint16x8x2_t vld2q_lane_u16(uint16_t const * ptr, uint16x8x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD2 {Vt.h - Vt2.h}[lane],[Xn]	Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x2x2_t_vld2_lane_u32(uint32_t const * ptr, uint32x2x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt.2S src.val[0] -> Vt.2S 0 <= lane <= 1	LD2 {Vt.s - Vt2.s}[lane],[Xn]	Vt.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
uint32x4x2_t_vld2_lane_u32(uint32_t const * ptr, uint32x4x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt.2.4S src.val[0] -> Vt.4S 0 <= lane <= 3	LD2 {Vt.s - Vt2.s}[lane],[Xn]	Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
float16x4x2_t_vld2_lane_f16(float16_t const * ptr, float16x4x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt.2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD2 {Vt.h - Vt2.h}[lane],[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
float16x8x2_t_vld2q_lane_f16(float16_t const * ptr, float16x8x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt.2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD2 {Vt.h - Vt2.h}[lane],[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
float32x2x2_t_vld2_lane_f32(float32_t const * ptr, float32x2x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt.2.2S src.val[0] -> Vt.2S 0 <= lane <= 1	LD2 {Vt.s - Vt2.s}[lane],[Xn]	Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
float32x4x2_t_vld2q_lane_f32(float32_t const * ptr, float32x4x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt.2.4S src.val[0] -> Vt.4S 0 <= lane <= 3	LD2 {Vt.s - Vt2.s}[lane],[Xn]	Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
poly16x4x2_t_vld2_lane_p16(poly16_t const * ptr, poly16x4x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt.2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD2 {Vt.h - Vt2.h}[lane],[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
poly16x8x2_t_vld2q_lane_p16(poly16_t const * ptr, poly16x8x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt.2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD2 {Vt.h - Vt2.h}[lane],[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int8x8x2_t_vld2_lane_s8(int8_t const * ptr, int8x8x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt.2.8B src.val[0] -> Vt.8B 0 <= lane <= 7	LD2 {Vt.b - Vt2.b}[lane],[Xn]	Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
uint8x8x2_t_vld2_lane_u8(uint8_t const * ptr, uint8x8x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt.2.8B src.val[0] -> Vt.8B 0 <= lane <= 7	LD2 {Vt.b - Vt2.b}[lane],[Xn]	Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly8x8x2_t_vld2_lane_p8(poly8_t const * ptr, poly8x8x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.8B src.val[0] -> Vt.8B 0 <= lane <= 7	LD2 {Vt.b - Vt2.b}[lane], [Xn]	Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
int8x16x2_t_vld2q_lane_s8(int8_t const * ptr, int8x16x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.16B src.val[0] -> Vt.16B 0 <= lane <= 15	LD2 {Vt.b - Vt2.b}[lane], [Xn]	Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	A64
uint8x16x2_t_vld2q_lane_u8(uint8_t const * ptr, uint8x16x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.16B src.val[0] -> Vt.16B 0 <= lane <= 15	LD2 {Vt.b - Vt2.b}[lane], [Xn]	Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	A64
poly8x16x2_t_vld2q_lane_p8(poly8_t const * ptr, poly8x16x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.16B src.val[0] -> Vt.16B 0 <= lane <= 15	LD2 {Vt.b - Vt2.b}[lane], [Xn]	Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	A64
int64x1x2_t_vld2_lane_s64(int64_t const * ptr, int64x1x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD2 {Vt.d - Vt2.d}[lane], [Xn]	ptr -> Xn Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
int64x2x2_t_vld2q_lane_s64(int64_t const * ptr, int64x2x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD2 {Vt.d - Vt2.d}[lane], [Xn]	ptr -> Xn Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
uint64x1x2_t_vld2_lane_u64(uint64_t const * ptr, uint64x1x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD2 {Vt.d - Vt2.d}[lane], [Xn]	Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
uint64x2x2_t_vld2q_lane_u64(uint64_t const * ptr, uint64x2x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD2 {Vt.d - Vt2.d}[lane], [Xn]	Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
poly64x1x2_t_vld2_lane_p64(poly64_t const * ptr, poly64x1x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD2 {Vt.d - Vt2.d}[lane], [Xn]	Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
poly64x2x2_t_vld2q_lane_p64(poly64_t const * ptr, poly64x2x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD2 {Vt.d - Vt2.d}[lane], [Xn]	Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
float64x1x2_t_vld2_lane_f64(float64_t const * ptr, float64x1x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD2 {Vt.d - Vt2.d}[lane], [Xn]	Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64x2x2_t_vld2q_lane_f64(float64_t const * ptr, float64x2x2_t src, const int lane)	ptr -> Xn src.val[1] -> Vt.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD2 {Vt.d - Vt.d}[lane],[Xn]	Vt.2D -> result.val[1] Vt.2D -> result.val[0]	A64
int16x4x3_t_vld3_lane_s16(int16_t const * ptr, int16x4x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.4H src.val[1] -> Vt.2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD3 {Vt.h - Vt.h}[lane],[Xn]	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x3_t_vld3q_lane_s16(int16_t const * ptr, int16x8x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.8H src.val[1] -> Vt.2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD3 {Vt.h - Vt.h}[lane],[Xn]	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int32x2x3_t_vld3_lane_s32(int32_t const * ptr, int32x2x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.2S src.val[1] -> Vt.2.2S src.val[0] -> Vt.2S 0 <= lane <= 1	LD3 {Vt.s - Vt.s}[lane],[Xn]	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
int32x4x3_t_vld3q_lane_s32(int32_t const * ptr, int32x4x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.4S src.val[1] -> Vt.2.4S src.val[0] -> Vt.4S 0 <= lane <= 3	LD3 {Vt.s - Vt.s}[lane],[Xn]	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint16x4x3_t_vld3_lane_u16(uint16_t const * ptr, uint16x4x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.4H src.val[1] -> Vt.2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD3 {Vt.h - Vt.h}[lane],[Xn]	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
uint16x8x3_t_vld3q_lane_u16(uint16_t const * ptr, uint16x8x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.8H src.val[1] -> Vt.2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD3 {Vt.h - Vt.h}[lane],[Xn]	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
uint32x2x3_t_vld3_lane_u32(uint32_t const * ptr, uint32x2x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.2S src.val[1] -> Vt.2.2S src.val[0] -> Vt.2S 0 <= lane <= 1	LD3 {Vt.s - Vt.s}[lane],[Xn]	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4x3_t_vld3q_lane_u32(uint32_t const * ptr, uint32x4x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.4S src.val[1] -> Vt.2.4S src.val[0] -> Vt.4S 0 <= lane <= 3	LD3 {Vt.s - Vt.3.s}[lane],[Xn]	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
float16x4x3_t_vld3_lane_f16(float16_t const * ptr, float16x4x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.4H src.val[1] -> Vt.2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD3 {Vt.h - Vt.3.h}[lane],[Xn]	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
float16x8x3_t_vld3q_lane_f16(float16_t const * ptr, float16x8x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.8H src.val[1] -> Vt.2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD3 {Vt.h - Vt.3.h}[lane],[Xn]	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
float32x2x3_t_vld3_lane_f32(float32_t const * ptr, float32x2x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.2S src.val[1] -> Vt.2.2S src.val[0] -> Vt.2S 0 <= lane <= 1	LD3 {Vt.s - Vt.3.s}[lane],[Xn]	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
float32x4x3_t_vld3q_lane_f32(float32_t const * ptr, float32x4x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.4S src.val[1] -> Vt.2.4S src.val[0] -> Vt.4S 0 <= lane <= 3	LD3 {Vt.s - Vt.3.s}[lane],[Xn]	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
poly16x4x3_t_vld3_lane_p16(poly16_t const * ptr, poly16x4x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.4H src.val[1] -> Vt.2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD3 {Vt.h - Vt.3.h}[lane],[Xn]	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
poly16x8x3_t_vld3q_lane_p16(poly16_t const * ptr, poly16x8x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.8H src.val[1] -> Vt.2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD3 {Vt.h - Vt.3.h}[lane],[Xn]	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int8x8x3_t_vld3_lane_s8(int8_t const * ptr, int8x8x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt.3.8B src.val[1] -> Vt.2.8B src.val[0] -> Vt.8B 0 <= lane <= 7	LD3 {Vt.b - Vt.3.b}[lane],[Xn]	Vt.3.8B -> result.val[2] Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x8x3_t_vld3_lane_u8(uint8_t const * ptr, uint8x8x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.8B src.val[1] -> Vt2.8B src.val[0] -> Vt.8B 0 <= lane <= 7	LD3 {Vt.b - Vt3.b}[lane], [Xn]	Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
poly8x8x3_t_vld3_lane_p8(poly8_t const * ptr, poly8x8x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.8B src.val[1] -> Vt2.8B src.val[0] -> Vt.8B 0 <= lane <= 7	LD3 {Vt.b - Vt3.b}[lane], [Xn]	Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
int8x16x3_t_vld3q_lane_s8(int8_t const * ptr, int8x16x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.16B src.val[1] -> Vt2.16B src.val[0] -> Vt.16B 0 <= lane <= 15	LD3 {Vt.b - Vt3.b}[lane], [Xn]	Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	A64
uint8x16x3_t_vld3q_lane_u8(uint8_t const * ptr, uint8x16x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.16B src.val[1] -> Vt2.16B src.val[0] -> Vt.16B 0 <= lane <= 15	LD3 {Vt.b - Vt3.b}[lane], [Xn]	Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	A64
poly8x16x3_t_vld3q_lane_p8(poly8_t const * ptr, poly8x16x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.16B src.val[1] -> Vt2.16B src.val[0] -> Vt.16B 0 <= lane <= 15	LD3 {Vt.b - Vt3.b}[lane], [Xn]	Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	A64
int64x1x3_t_vld3_lane_s64(int64_t const * ptr, int64x1x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.1D src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD3 {Vt.d - Vt3.d}[lane], [Xn]	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
int64x2x3_t_vld3q_lane_s64(int64_t const * ptr, int64x2x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.2D src.val[1] -> Vt2.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD3 {Vt.d - Vt3.d}[lane], [Xn]	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
uint64x1x3_t_vld3_lane_u64(uint64_t const * ptr, uint64x1x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.1D src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD3 {Vt.d - Vt3.d}[lane], [Xn]	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64x2x3_t vld3q_lane_u64(uint64_t const * ptr, uint64x2x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.2D src.val[1] -> Vt2.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD3 {Vt.d - Vt3.d}[lane], [Xn]	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
poly64x1x3_t vld3_lane_p64(poly64_t const * ptr, poly64x1x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.1D src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD3 {Vt.d - Vt3.d}[lane], [Xn]	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
poly64x2x3_t vld3q_lane_p64(poly64_t const * ptr, poly64x2x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.2D src.val[1] -> Vt2.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD3 {Vt.d - Vt3.d}[lane], [Xn]	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
float64x1x3_t vld3_lane_f64(float64_t const * ptr, float64x1x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.1D src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD3 {Vt.d - Vt3.d}[lane], [Xn]	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
float64x2x3_t vld3q_lane_f64(float64_t const * ptr, float64x2x3_t src, const int lane)	ptr -> Xn src.val[2] -> Vt3.2D src.val[1] -> Vt2.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD3 {Vt.d - Vt3.d}[lane], [Xn]	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
int16x4x4_t vld4_lane_s16(int16_t const * ptr, int16x4x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.4H src.val[2] -> Vt3.4H src.val[1] -> Vt2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD4 {Vt.h - Vt4.h}[lane], [Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x4_t vld4q_lane_s16(int16_t const * ptr, int16x8x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.8H src.val[2] -> Vt3.8H src.val[1] -> Vt2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD4 {Vt.h - Vt4.h}[lane], [Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x2x4_t_vld4_lane_s32(int32_t const * ptr, int32x2x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.2S src.val[2] -> Vt3.2S src.val[1] -> Vt2.2S src.val[0] -> Vt.2S 0 <= lane <= 1	LD4 {Vt.s - Vt4.s}[lane],[Xn]	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
int32x4x4_t_vld4q_lane_s32(int32_t const * ptr, int32x4x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.4S src.val[2] -> Vt3.4S src.val[1] -> Vt2.4S src.val[0] -> Vt.4S 0 <= lane <= 3	LD4 {Vt.s - Vt4.s}[lane],[Xn]	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint16x4x4_t_vld4_lane_u16(uint16_t const * ptr, uint16x4x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.4H src.val[2] -> Vt3.4H src.val[1] -> Vt2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD4 {Vt.h - Vt4.h}[lane],[Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
uint16x8x4_t_vld4q_lane_u16(uint16_t const * ptr, uint16x8x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.8H src.val[2] -> Vt3.8H src.val[1] -> Vt2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD4 {Vt.h - Vt4.h}[lane],[Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
uint32x2x4_t_vld4_lane_u32(uint32_t const * ptr, uint32x2x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.2S src.val[2] -> Vt3.2S src.val[1] -> Vt2.2S src.val[0] -> Vt.2S 0 <= lane <= 1	LD4 {Vt.s - Vt4.s}[lane],[Xn]	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
uint32x4x4_t_vld4q_lane_u32(uint32_t const * ptr, uint32x4x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.4S src.val[2] -> Vt3.4S src.val[1] -> Vt2.4S src.val[0] -> Vt.4S 0 <= lane <= 3	LD4 {Vt.s - Vt4.s}[lane],[Xn]	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x4x4_t_vld4_lane_f16(float16_t const * ptr, float16x4x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.4H src.val[2] -> Vt3.4H src.val[1] -> Vt2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD4 {Vt.h - Vt4.h}[lane], [Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
float16x8x4_t_vld4q_lane_f16(float16_t const * ptr, float16x8x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.8H src.val[2] -> Vt3.8H src.val[1] -> Vt2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD4 {Vt.h - Vt4.h}[lane], [Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
float32x2x4_t_vld4_lane_f32(float32_t const * ptr, float32x2x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.2S src.val[2] -> Vt3.2S src.val[1] -> Vt2.2S src.val[0] -> Vt.2S 0 <= lane <= 1	LD4 {Vt.s - Vt4.s}[lane], [Xn]	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
float32x4x4_t_vld4q_lane_f32(float32_t const * ptr, float32x4x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.4S src.val[2] -> Vt3.4S src.val[1] -> Vt2.4S src.val[0] -> Vt.4S 0 <= lane <= 3	LD4 {Vt.s - Vt4.s}[lane], [Xn]	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
poly16x4x4_t_vld4_lane_p16(poly16_t const * ptr, poly16x4x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.4H src.val[2] -> Vt3.4H src.val[1] -> Vt2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD4 {Vt.h - Vt4.h}[lane], [Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
poly16x8x4_t_vld4q_lane_p16(poly16_t const * ptr, poly16x8x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.8H src.val[2] -> Vt3.8H src.val[1] -> Vt2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD4 {Vt.h - Vt4.h}[lane], [Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x8x4_t vld4_lane_s8(int8_t const * ptr, int8x8x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.8B src.val[2] -> Vt3.8B src.val[1] -> Vt2.8B src.val[0] -> Vt.8B 0 <= lane <= 7	LD4 {Vt.b - Vt4.b}[lane], [Xn]	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
uint8x8x4_t vld4_lane_u8(uint8_t const * ptr, uint8x8x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.8B src.val[2] -> Vt3.8B src.val[1] -> Vt2.8B src.val[0] -> Vt.8B 0 <= lane <= 7	LD4 {Vt.b - Vt4.b}[lane], [Xn]	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
poly8x8x4_t vld4_lane_p8(poly8_t const * ptr, poly8x8x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.8B src.val[2] -> Vt3.8B src.val[1] -> Vt2.8B src.val[0] -> Vt.8B 0 <= lane <= 7	LD4 {Vt.b - Vt4.b}[lane], [Xn]	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
int8x16x4_t vld4q_lane_s8(int8_t const * ptr, int8x16x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.16B src.val[2] -> Vt3.16B src.val[1] -> Vt2.16B src.val[0] -> Vt.16B 0 <= lane <= 15	LD4 {Vt.b - Vt4.b}[lane], [Xn]	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	A64
uint8x16x4_t vld4q_lane_u8(uint8_t const * ptr, uint8x16x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.16B src.val[2] -> Vt3.16B src.val[1] -> Vt2.16B src.val[0] -> Vt.16B 0 <= lane <= 15	LD4 {Vt.b - Vt4.b}[lane], [Xn]	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	A64
poly8x16x4_t vld4q_lane_p8(poly8_t const * ptr, poly8x16x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.16B src.val[2] -> Vt3.16B src.val[1] -> Vt2.16B src.val[0] -> Vt.16B 0 <= lane <= 15	LD4 {Vt.b - Vt4.b}[lane], [Xn]	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64x1x4_t_vld4_lane_s64(int64_t const * ptr, int64x1x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.1D src.val[2] -> Vt3.1D src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD4 {Vt.d - Vt4.d}[lane], [Xn]	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
int64x2x4_t_vld4q_lane_s64(int64_t const * ptr, int64x2x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.2D src.val[2] -> Vt3.2D src.val[1] -> Vt2.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD4 {Vt.d - Vt4.d}[lane], [Xn]	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
uint64x1x4_t_vld4_lane_u64(uint64_t const * ptr, uint64x1x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.1D src.val[2] -> Vt3.1D src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD4 {Vt.d - Vt4.d}[lane], [Xn]	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
uint64x2x4_t_vld4q_lane_u64(uint64_t const * ptr, uint64x2x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.2D src.val[2] -> Vt3.2D src.val[1] -> Vt2.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD4 {Vt.d - Vt4.d}[lane], [Xn]	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
poly64x1x4_t_vld4_lane_p64(poly64_t const * ptr, poly64x1x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.1D src.val[2] -> Vt3.1D src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD4 {Vt.d - Vt4.d}[lane], [Xn]	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
poly64x2x4_t_vld4q_lane_p64(poly64_t const * ptr, poly64x2x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.2D src.val[2] -> Vt3.2D src.val[1] -> Vt2.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD4 {Vt.d - Vt4.d}[lane], [Xn]	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
float64x1x4_t_vld4_lane_f64(float64_t const * ptr, float64x1x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.1D src.val[2] -> Vt3.1D src.val[1] -> Vt2.1D src.val[0] -> Vt.1D lane == 0	LD4 {Vt.d - Vt4.d}[lane], [Xn]	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64x2x4_t vld4q_lane_f64(float64_t const * ptr, float64x2x4_t src, const int lane)	ptr -> Xn src.val[3] -> Vt4.2D src.val[2] -> Vt3.2D src.val[1] -> Vt2.2D src.val[0] -> Vt.2D 0 <= lane <= 1	LD4 {Vt.d - Vt4.d}[lane], [Xn]	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
void vst2_lane_s8(int8_t * ptr, int8x8x2_t val, const int lane)	ptr -> Xn val.val[1] -> Vt2.8B val.val[0] -> Vt.8B 0 <= lane <= 7	ST2 {Vt.b - Vt2.b}[lane], [Xn]	void -> result	v7/A32/A64
void vst2_lane_u8(uint8_t * ptr, uint8x8x2_t val, const int lane)	ptr -> Xn val.val[1] -> Vt2.8B val.val[0] -> Vt.8B 0 <= lane <= 7	ST2 {Vt.b - Vt2.b}[lane], [Xn]	void -> result	v7/A32/A64
void vst2_lane_p8(poly8_t * ptr, poly8x8x2_t val, const int lane)	ptr -> Xn val.val[1] -> Vt2.8B val.val[0] -> Vt.8B 0 <= lane <= 7	ST2 {Vt.b - Vt2.b}[lane], [Xn]	void -> result	v7/A32/A64
void vst3_lane_s8(int8_t * ptr, int8x8x3_t val, const int lane)	ptr -> Xn val.val[2] -> Vt3.8B val.val[1] -> Vt2.8B val.val[0] -> Vt.8B 0 <= lane <= 7	ST3 {Vt.b - Vt3.b}[lane], [Xn]	void -> result	v7/A32/A64
void vst3_lane_u8(uint8_t * ptr, uint8x8x3_t val, const int lane)	ptr -> Xn val.val[2] -> Vt3.8B val.val[1] -> Vt2.8B val.val[0] -> Vt.8B 0 <= lane <= 7	ST3 {Vt.b - Vt3.b}[lane], [Xn]	void -> result	v7/A32/A64
void vst3_lane_p8(poly8_t * ptr, poly8x8x3_t val, const int lane)	ptr -> Xn val.val[2] -> Vt3.8B val.val[1] -> Vt2.8B val.val[0] -> Vt.8B 0 <= lane <= 7	ST3 {Vt.b - Vt3.b}[lane], [Xn]	void -> result	v7/A32/A64
void vst4_lane_s8(int8_t * ptr, int8x8x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.8B val.val[2] -> Vt3.8B val.val[1] -> Vt2.8B val.val[0] -> Vt.8B 0 <= lane <= 7	ST4 {Vt.b - Vt4.b}[lane], [Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst4_lane_u8(uint8_t * ptr, uint8x8x4_t val, const int lane)</code>	<code>ptr -> Xn val.val[3] -> Vt4.8B val.val[2] -> Vt3.8B val.val[1] -> Vt2.8B val.val[0] -> Vt.8B 0 <= lane <= 7</code>	<code>ST4 {Vt.b - Vt4.b}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst4_lane_p8(poly8_t * ptr, poly8x8x4_t val, const int lane)</code>	<code>ptr -> Xn val.val[3] -> Vt4.8B val.val[2] -> Vt3.8B val.val[1] -> Vt2.8B val.val[0] -> Vt.8B 0 <= lane <= 7</code>	<code>ST4 {Vt.b - Vt4.b}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2_lane_s16(int16_t * ptr, int16x4x2_t val, const int lane)</code>	<code>ptr -> Xn val.val[1] -> Vt2.4H val.val[0] -> Vt.4H 0 <= lane <= 3</code>	<code>ST2 {Vt.h - Vt2.h}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2q_lane_s16(int16_t * ptr, int16x8x2_t val, const int lane)</code>	<code>ptr -> Xn val.val[1] -> Vt2.8H val.val[0] -> Vt.8H 0 <= lane <= 7</code>	<code>ST2 {Vt.h - Vt2.h}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2_lane_s32(int32_t * ptr, int32x2x2_t val, const int lane)</code>	<code>ptr -> Xn val.val[1] -> Vt2.2S val.val[0] -> Vt.2S 0 <= lane <= 1</code>	<code>ST2 {Vt.s - Vt2.s}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2q_lane_s32(int32_t * ptr, int32x4x2_t val, const int lane)</code>	<code>ptr -> Xn val.val[1] -> Vt2.4S val.val[0] -> Vt.4S 0 <= lane <= 3</code>	<code>ST2 {Vt.s - Vt2.s}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2_lane_u16(uint16_t * ptr, uint16x4x2_t val, const int lane)</code>	<code>ptr -> Xn val.val[1] -> Vt2.4H val.val[0] -> Vt.4H 0 <= lane <= 3</code>	<code>ST2 {Vt.h - Vt2.h}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2q_lane_u16(uint16_t * ptr, uint16x8x2_t val, const int lane)</code>	<code>ptr -> Xn val.val[1] -> Vt2.8H val.val[0] -> Vt.8H 0 <= lane <= 7</code>	<code>ST2 {Vt.h - Vt2.h}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2_lane_u32(uint32_t * ptr, uint32x2x2_t val, const int lane)</code>	<code>ptr -> Xn val.val[1] -> Vt2.2S val.val[0] -> Vt.2S 0 <= lane <= 1</code>	<code>ST2 {Vt.s - Vt2.s}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst2q_lane_u32(uint32_t * ptr, uint32x4x2_t val, const int lane)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.4S</code> <code>val.val[0] -> Vt.4S</code> <code>0 <= lane <= 3</code>	<code>ST2 {Vt.s - Vt.2.s}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2_lane_f16(float16_t * ptr, float16x4x2_t val, const int lane)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.4H</code> <code>val.val[0] -> Vt.4H</code> <code>0 <= lane <= 3</code>	<code>ST2 {Vt.h - Vt.2.h}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2q_lane_f16(float16_t * ptr, float16x8x2_t val, const int lane)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.8H</code> <code>val.val[0] -> Vt.8H</code> <code>0 <= lane <= 7</code>	<code>ST2 {Vt.h - Vt.2.h}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2_lane_f32(float32_t * ptr, float32x2x2_t val, const int lane)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.2S</code> <code>val.val[0] -> Vt.2S</code> <code>0 <= lane <= 1</code>	<code>ST2 {Vt.s - Vt.2.s}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2q_lane_f32(float32_t * ptr, float32x4x2_t val, const int lane)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.4S</code> <code>val.val[0] -> Vt.4S</code> <code>0 <= lane <= 3</code>	<code>ST2 {Vt.s - Vt.2.s}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2_lane_p16(poly16_t * ptr, poly16x4x2_t val, const int lane)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.4H</code> <code>val.val[0] -> Vt.4H</code> <code>0 <= lane <= 3</code>	<code>ST2 {Vt.h - Vt.2.h}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2q_lane_p16(poly16_t * ptr, poly16x8x2_t val, const int lane)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.8H</code> <code>val.val[0] -> Vt.8H</code> <code>0 <= lane <= 7</code>	<code>ST2 {Vt.h - Vt.2.h}[lane], [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst2q_lane_s8(int8_t * ptr, int8x16x2_t val, const int lane)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.16B</code> <code>val.val[0] -> Vt.16B</code> <code>0 <= lane <= 15</code>	<code>ST2 {Vt.b - Vt.2.b}[lane], [Xn]</code>	<code>void -> result</code>	A64
<code>void vst2q_lane_u8(uint8_t * ptr, uint8x16x2_t val, const int lane)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.16B</code> <code>val.val[0] -> Vt.16B</code> <code>0 <= lane <= 15</code>	<code>ST2 {Vt.b - Vt.2.b}[lane], [Xn]</code>	<code>void -> result</code>	A64
<code>void vst2q_lane_p8(poly8_t * ptr, poly8x16x2_t val, const int lane)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.16B</code> <code>val.val[0] -> Vt.16B</code> <code>0 <= lane <= 15</code>	<code>ST2 {Vt.b - Vt.2.b}[lane], [Xn]</code>	<code>void -> result</code>	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void vst2_lane_s64(int64_t * ptr, int64x1x2_t val, const int lane)	ptr -> Xn val.val[1] -> Vt.1D val.val[0] -> Vt.1D lane == 0	ST2 {Vt.d - Vt.d}[lane], [Xn]	void -> result	A64
void vst2q_lane_s64(int64_t * ptr, int64x2x2_t val, const int lane)	ptr -> Xn val.val[1] -> Vt.2D val.val[0] -> Vt.2D 0 <= lane <= 1	ST2 {Vt.d - Vt.d}[lane], [Xn]	void -> result	A64
void vst2_lane_u64(uint64_t * ptr, uint64x1x2_t val, const int lane)	ptr -> Xn val.val[1] -> Vt.1D val.val[0] -> Vt.1D lane == 0	ST2 {Vt.d - Vt.d}[lane], [Xn]	void -> result	A64
void vst2q_lane_u64(uint64_t * ptr, uint64x2x2_t val, const int lane)	ptr -> Xn val.val[1] -> Vt.2D val.val[0] -> Vt.2D 0 <= lane <= 1	ST2 {Vt.d - Vt.d}[lane], [Xn]	void -> result	A64
void vst2_lane_p64(poly64_t * ptr, poly64x1x2_t val, const int lane)	ptr -> Xn val.val[1] -> Vt.1D val.val[0] -> Vt.1D lane == 0	ST2 {Vt.d - Vt.d}[lane], [Xn]	void -> result	A64
void vst2q_lane_p64(poly64_t * ptr, poly64x2x2_t val, const int lane)	ptr -> Xn val.val[1] -> Vt.2D val.val[0] -> Vt.2D 0 <= lane <= 1	ST2 {Vt.d - Vt.d}[lane], [Xn]	void -> result	A64
void vst2_lane_f64(float64_t * ptr, float64x1x2_t val, const int lane)	ptr -> Xn val.val[1] -> Vt.1D val.val[0] -> Vt.1D lane == 0	ST2 {Vt.d - Vt.d}[lane], [Xn]	void -> result	A64
void vst2q_lane_f64(float64_t * ptr, float64x2x2_t val, const int lane)	ptr -> Xn val.val[1] -> Vt.2D val.val[0] -> Vt.2D 0 <= lane <= 2	ST2 {Vt.d - Vt.d}[lane], [Xn]	void -> result	A64
void vst3_lane_s16(int16_t * ptr, int16x4x3_t val, const int lane)	ptr -> Xn val.val[2] -> Vt.3.4H val.val[1] -> Vt.2.4H val.val[0] -> Vt.4H 0 <= lane <= 3	ST3 {Vt.h - Vt3.h}[lane], [Xn]	void -> result	v7/A32/A64
void vst3q_lane_s16(int16_t * ptr, int16x8x3_t val, const int lane)	ptr -> Xn val.val[2] -> Vt.3.8H val.val[1] -> Vt.2.8H val.val[0] -> Vt.8H 0 <= lane <= 7	ST3 {Vt.h - Vt3.h}[lane], [Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst3_lane_s32(int32_t * ptr, int32x2x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.2S val.val[1] -> Vt.2.2S val.val[0] -> Vt.2S 0 <= lane <= 1</code>	<code>ST3 {Vt.s - Vt3.s}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_lane_s32(int32_t * ptr, int32x4x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.4S val.val[1] -> Vt.2.4S val.val[0] -> Vt.4S 0 <= lane <= 3</code>	<code>ST3 {Vt.s - Vt3.s}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_lane_u16(uint16_t * ptr, uint16x4x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.4H val.val[1] -> Vt.2.4H val.val[0] -> Vt.4H 0 <= lane <= 3</code>	<code>ST3 {Vt.h - Vt3.h}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_lane_u16(uint16_t * ptr, uint16x8x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.8H val.val[1] -> Vt.2.8H val.val[0] -> Vt.8H 0 <= lane <= 7</code>	<code>ST3 {Vt.h - Vt3.h}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_lane_u32(uint32_t * ptr, uint32x2x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.2S val.val[1] -> Vt.2.2S val.val[0] -> Vt.2S 0 <= lane <= 1</code>	<code>ST3 {Vt.s - Vt3.s}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_lane_u32(uint32_t * ptr, uint32x4x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.4S val.val[1] -> Vt.2.4S val.val[0] -> Vt.4S 0 <= lane <= 3</code>	<code>ST3 {Vt.s - Vt3.s}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_lane_f16(float16_t * ptr, float16x4x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.4H val.val[1] -> Vt.2.4H val.val[0] -> Vt.4H 0 <= lane <= 3</code>	<code>ST3 {Vt.h - Vt3.h}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_lane_f16(float16_t * ptr, float16x8x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.8H val.val[1] -> Vt.2.8H val.val[0] -> Vt.8H 0 <= lane <= 7</code>	<code>ST3 {Vt.h - Vt3.h}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst3_lane_f32(float32_t * ptr, float32x2x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.2S val.val[1] -> Vt.2.2S val.val[0] -> Vt.2S 0 <= lane <= 1</code>	<code>ST3 {Vt.s - Vt3.s}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_lane_f32(float32_t * ptr, float32x4x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.4S val.val[1] -> Vt.2.4S val.val[0] -> Vt.4S 0 <= lane <= 3</code>	<code>ST3 {Vt.s - Vt3.s}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_lane_p16(poly16_t * ptr, poly16x4x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.4H val.val[1] -> Vt.2.4H val.val[0] -> Vt.4H 0 <= lane <= 3</code>	<code>ST3 {Vt.h - Vt3.h}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_lane_p16(poly16_t * ptr, poly16x8x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.8H val.val[1] -> Vt.2.8H val.val[0] -> Vt.8H 0 <= lane <= 7</code>	<code>ST3 {Vt.h - Vt3.h}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_lane_s8(int8_t * ptr, int8x16x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.16B val.val[1] -> Vt.2.16B val.val[0] -> Vt.16B 0 <= lane <= 15</code>	<code>ST3 {Vt.b - Vt3.b}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_lane_u8(uint8_t * ptr, uint8x16x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.16B val.val[1] -> Vt.2.16B val.val[0] -> Vt.16B 0 <= lane <= 15</code>	<code>ST3 {Vt.b - Vt3.b}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3q_lane_p8(poly8_t * ptr, poly8x16x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.16B val.val[1] -> Vt.2.16B val.val[0] -> Vt.16B 0 <= lane <= 15</code>	<code>ST3 {Vt.b - Vt3.b}[lane],[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst3_lane_s64(int64_t * ptr, int64x1x3_t val, const int lane)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.1D val.val[1] -> Vt.2.1D val.val[0] -> Vt.1D lane == 0</code>	<code>ST3 {Vt.d - Vt3.d}[lane],[Xn]</code>	<code>void -> result</code>	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst3q_lane_s64(int64_t * ptr, int64x2x3_t val, const int lane)</code>	ptr -> Xn val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D 0 <= lane <= 1	ST3{Vt.d - Vt3.d}[lane],[Xn]	void -> result	A64
<code>void vst3_lane_u64(uint64_t * ptr, uint64x1x3_t val, const int lane)</code>	ptr -> Xn val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D lane == 0	ST3{Vt.d - Vt3.d}[lane],[Xn]	void -> result	A64
<code>void vst3q_lane_u64(uint64_t * ptr, uint64x2x3_t val, const int lane)</code>	ptr -> Xn val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D 0 <= lane <= 1	ST3{Vt.d - Vt3.d}[lane],[Xn]	void -> result	A64
<code>void vst3_lane_p64(poly64_t * ptr, poly64x1x3_t val, const int lane)</code>	ptr -> Xn val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D lane == 0	ST3{Vt.d - Vt3.d}[lane],[Xn]	void -> result	A64
<code>void vst3q_lane_p64(poly64_t * ptr, poly64x2x3_t val, const int lane)</code>	ptr -> Xn val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D 0 <= lane <= 1	ST3{Vt.d - Vt3.d}[lane],[Xn]	void -> result	A64
<code>void vst3_lane_f64(float64_t * ptr, float64x1x3_t val, const int lane)</code>	ptr -> Xn val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D lane == 0	ST3{Vt.d - Vt3.d}[lane],[Xn]	void -> result	A64
<code>void vst3q_lane_f64(float64_t * ptr, float64x2x3_t val, const int lane)</code>	ptr -> Xn val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D 0 <= lane <= 1	ST3{Vt.d - Vt3.d}[lane],[Xn]	void -> result	A64
<code>void vst4_lane_s16(int16_t * ptr, int16x4x4_t val, const int lane)</code>	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H 0 <= lane <= 3	ST4{Vt.h - Vt4.h}[lane],[Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void vst4q_lane_s16(int16_t * ptr, int16x8x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H 0 <= lane <= 7	ST4 {Vt.h - Vt4.h}[lane],[Xn]	void -> result	v7/A32/A64
void vst4_lane_s32(int32_t * ptr, int32x2x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.2S val.val[2] -> Vt3.2S val.val[1] -> Vt2.2S val.val[0] -> Vt.2S 0 <= lane <= 1	ST4 {Vt.s - Vt4.s}[lane],[Xn]	void -> result	v7/A32/A64
void vst4q_lane_s32(int32_t * ptr, int32x4x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.4S val.val[2] -> Vt3.4S val.val[1] -> Vt2.4S val.val[0] -> Vt.4S 0 <= lane <= 3	ST4 {Vt.s - Vt4.s}[lane],[Xn]	void -> result	v7/A32/A64
void vst4_lane_u16(uint16_t * ptr, uint16x4x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H 0 <= lane <= 3	ST4 {Vt.h - Vt4.h}[lane],[Xn]	void -> result	v7/A32/A64
void vst4q_lane_u16(uint16_t * ptr, uint16x8x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H 0 <= lane <= 7	ST4 {Vt.h - Vt4.h}[lane],[Xn]	void -> result	v7/A32/A64
void vst4_lane_u32(uint32_t * ptr, uint32x2x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.2S val.val[2] -> Vt3.2S val.val[1] -> Vt2.2S val.val[0] -> Vt.2S 0 <= lane <= 1	ST4 {Vt.s - Vt4.s}[lane],[Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void vst4q_lane_u32(uint32_t * ptr, uint32x4x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.4S val.val[2] -> Vt3.4S val.val[1] -> Vt2.4S val.val[0] -> Vt.4S 0 <= lane <= 3	ST4 {Vt.s - Vt4.s}[lane],[Xn]	void -> result	v7/A32/A64
void vst4_lane_f16(float16_t * ptr, float16x4x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H 0 <= lane <= 3	ST4 {Vt.h - Vt4.h}[lane],[Xn]	void -> result	v7/A32/A64
void vst4q_lane_f16(float16_t * ptr, float16x8x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H 0 <= lane <= 7	ST4 {Vt.h - Vt4.h}[lane],[Xn]	void -> result	v7/A32/A64
void vst4_lane_f32(float32_t * ptr, float32x2x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.2S val.val[2] -> Vt3.2S val.val[1] -> Vt2.2S val.val[0] -> Vt.2S 0 <= lane <= 1	ST4 {Vt.s - Vt4.s}[lane],[Xn]	void -> result	v7/A32/A64
void vst4q_lane_f32(float32_t * ptr, float32x4x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.4S val.val[2] -> Vt3.4S val.val[1] -> Vt2.4S val.val[0] -> Vt.4S 0 <= lane <= 3	ST4 {Vt.s - Vt4.s}[lane],[Xn]	void -> result	v7/A32/A64
void vst4_lane_p16(poly16_t * ptr, poly16x4x4_t val, const int lane)	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H 0 <= lane <= 3	ST4 {Vt.h - Vt4.h}[lane],[Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst4q_lane_p16(poly16_t * ptr, poly16x8x4_t val, const int lane)</code>	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H 0 <= lane <= 7	ST4 {Vt.h - Vt4.h}[lane], [Xn]	void -> result	v7/A32/A64
<code>void vst4q_lane_s8(int8_t * ptr, int8x16x4_t val, const int lane)</code>	ptr -> Xn val.val[3] -> Vt4.16B val.val[2] -> Vt3.16B val.val[1] -> Vt2.16B val.val[0] -> Vt.16B 0 <= lane <= 15	ST4 {Vt.b - Vt4.b}[lane], [Xn]	void -> result	A64
<code>void vst4q_lane_u8(uint8_t * ptr, uint8x16x4_t val, const int lane)</code>	ptr -> Xn val.val[3] -> Vt4.16B val.val[2] -> Vt3.16B val.val[1] -> Vt2.16B val.val[0] -> Vt.16B 0 <= lane <= 15	ST4 {Vt.b - Vt4.b}[lane], [Xn]	void -> result	A64
<code>void vst4q_lane_p8(poly8_t * ptr, poly8x16x4_t val, const int lane)</code>	ptr -> Xn val.val[3] -> Vt4.16B val.val[2] -> Vt3.16B val.val[1] -> Vt2.16B val.val[0] -> Vt.16B 0 <= lane <= 15	ST4 {Vt.b - Vt4.b}[lane], [Xn]	void -> result	A64
<code>void vst4_lane_s64(int64_t * ptr, int64x1x4_t val, const int lane)</code>	ptr -> Xn val.val[3] -> Vt4.1D val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D lane == 0	ST4 {Vt.d - Vt4.d}[lane], [Xn]	void -> result	A64
<code>void vst4_lane_s64(int64_t * ptr, int64x2x4_t val, const int lane)</code>	ptr -> Xn val.val[3] -> Vt4.2D val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D 0 <= lane <= 1	ST4 {Vt.d - Vt4.d}[lane], [Xn]	void -> result	A64
<code>void vst4_lane_u64(uint64_t * ptr, uint64x1x4_t val, const int lane)</code>	ptr -> Xn val.val[3] -> Vt4.1D val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D lane == 0	ST4 {Vt.d - Vt4.d}[lane], [Xn]	void -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst4q_lane_u64(uint64_t * ptr, uint64x2x4_t val, const int lane)</code>	<code>ptr -> Xn val.val[3] -> Vt4.2D val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D 0 <= lane <= 1</code>	<code>ST4 {Vt.d - Vt4.d}[lane], [Xn]</code>	<code>void -> result</code>	A64
<code>void vst4_lane_p64(poly64_t * ptr, poly64x1x4_t val, const int lane)</code>	<code>ptr -> Xn val.val[3] -> Vt4.1D val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D lane == 0</code>	<code>ST4 {Vt.d - Vt4.d}[lane], [Xn]</code>	<code>void -> result</code>	A64
<code>void vst4q_lane_p64(poly64_t * ptr, poly64x2x4_t val, const int lane)</code>	<code>ptr -> Xn val.val[3] -> Vt4.2D val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D 0 <= lane <= 1</code>	<code>ST4 {Vt.d - Vt4.d}[lane], [Xn]</code>	<code>void -> result</code>	A64
<code>void vst4_lane_f64(float64_t * ptr, float64x1x4_t val, const int lane)</code>	<code>ptr -> Xn val.val[3] -> Vt4.1D val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D lane == 0</code>	<code>ST4 {Vt.d - Vt4.d}[lane], [Xn]</code>	<code>void -> result</code>	A64
<code>void vst4q_lane_f64(float64_t * ptr, float64x2x4_t val, const int lane)</code>	<code>ptr -> Xn val.val[3] -> Vt4.2D val.val[2] -> Vt3.2D val.val[1] -> Vt2.2D val.val[0] -> Vt.2D 0 <= lane <= 1</code>	<code>ST4 {Vt.d - Vt4.d}[lane], [Xn]</code>	<code>void -> result</code>	A64
<code>void vst1_s8_x2(int8_t * ptr, int8x8x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt2.8B val.val[0] -> Vt.8B</code>	<code>ST1 {Vt.8B - Vt2.8B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_s8_x2(int8_t * ptr, int8x16x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt2.16B val.val[0] -> Vt.16B</code>	<code>ST1 {Vt.16B - Vt2.16B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_s16_x2(int16_t * ptr, int16x4x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt2.4H val.val[0] -> Vt.4H</code>	<code>ST1 {Vt.4H - Vt2.4H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_s16_x2(int16_t * ptr, int16x8x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt2.8H val.val[0] -> Vt.8H</code>	<code>ST1 {Vt.8H - Vt2.8H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst1_s32_x2(int32_t * ptr, int32x2x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2S val.val[0] -> Vt.2S</code>	<code>ST1 {Vt.2S - Vt.2S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_s32_x2(int32_t * ptr, int32x4x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.4S val.val[0] -> Vt.4S</code>	<code>ST1 {Vt.4S - Vt.4S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_u8_x2(uint8_t * ptr, uint8x8x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.8B val.val[0] -> Vt.8B</code>	<code>ST1 {Vt.8B - Vt.8B},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u8_x2(uint8_t * ptr, uint8x16x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.16B val.val[0] -> Vt.16B</code>	<code>ST1 {Vt.16B - Vt.16B},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_u16_x2(uint16_t * ptr, uint16x4x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.4H val.val[0] -> Vt.4H</code>	<code>ST1 {Vt.4H - Vt.4H},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u16_x2(uint16_t * ptr, uint16x8x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.8H val.val[0] -> Vt.8H</code>	<code>ST1 {Vt.8H - Vt.8H},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_u32_x2(uint32_t * ptr, uint32x2x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2S val.val[0] -> Vt.2S</code>	<code>ST1 {Vt.2S - Vt.2S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u32_x2(uint32_t * ptr, uint32x4x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.4S val.val[0] -> Vt.4S</code>	<code>ST1 {Vt.4S - Vt.4S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_f16_x2(float16_t * ptr, float16x4x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.4H val.val[0] -> Vt.4H</code>	<code>ST1 {Vt.4H - Vt.4H},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_f16_x2(float16_t * ptr, float16x8x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.8H val.val[0] -> Vt.8H</code>	<code>ST1 {Vt.8H - Vt.8H},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_f32_x2(float32_t * ptr, float32x2x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2S val.val[0] -> Vt.2S</code>	<code>ST1 {Vt.2S - Vt.2S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_f32_x2(float32_t * ptr, float32x4x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.4S val.val[0] -> Vt.4S</code>	<code>ST1 {Vt.4S - Vt.4S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_p8_x2(poly8_t * ptr, poly8x8x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.8B val.val[0] -> Vt.8B</code>	<code>ST1 {Vt.8B - Vt.8B},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_p8_x2(poly8_t * ptr, poly8x16x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.16B val.val[0] -> Vt.16B</code>	<code>ST1 {Vt.16B - Vt.16B},[Xn]</code>	<code>void -> result</code>	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst1_p16_x2(poly16_t * ptr, poly16x4x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.4H val.val[0] -> Vt.4H</code>	<code>ST1 {Vt.4H - Vt.2.4H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_p16_x2(poly16_t * ptr, poly16x8x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.8H val.val[0] -> Vt.8H</code>	<code>ST1 {Vt.8H - Vt.2.8H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_s64_x2(int64_t * ptr, int64x1x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.1D val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt.2.1D}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_u64_x2(uint64_t * ptr, uint64x1x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.1D val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt.2.1D}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_p64_x2(poly64_t * ptr, poly64x1x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.1D val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt.2.1D}, [Xn]</code>	<code>void -> result</code>	A32/A64
<code>void vst1q_s64_x2(int64_t * ptr, int64x2x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.2D val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt.2.2D}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u64_x2(uint64_t * ptr, uint64x2x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.2D val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt.2.2D}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_p64_x2(poly64_t * ptr, poly64x2x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.2D val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt.2.2D}, [Xn]</code>	<code>void -> result</code>	A32/A64
<code>void vst1_f64_x2(float64_t * ptr, float64x1x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.1D val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt.2.1D}, [Xn]</code>	<code>void -> result</code>	A64
<code>void vst1q_f64_x2(float64_t * ptr, float64x2x2_t val)</code>	<code>ptr -> Xn val.val[1] -> Vt.2.2D val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt.2.2D}, [Xn]</code>	<code>void -> result</code>	A64
<code>void vst1_s8_x3(int8_t * ptr, int8x8x3_t val)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.8B val.val[1] -> Vt.2.8B val.val[0] -> Vt.8B</code>	<code>ST1 {Vt.8B - Vt.3.8B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_s8_x3(int8_t * ptr, int8x16x3_t val)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.16B val.val[1] -> Vt.2.16B val.val[0] -> Vt.16B</code>	<code>ST1 {Vt.16B - Vt.3.16B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_s16_x3(int16_t * ptr, int16x4x3_t val)</code>	<code>ptr -> Xn val.val[2] -> Vt.3.4H val.val[1] -> Vt.2.4H val.val[0] -> Vt.4H</code>	<code>ST1 {Vt.4H - Vt.3.4H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst1q_s16_x3(int16_t * ptr, int16x8x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.8H</code> <code>val.val[1] -> Vt2.8H</code> <code>val.val[0] -> Vt.8H</code>	<code>ST1 {Vt.8H - Vt3.8H},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_s32_x3(int32_t * ptr, int32x2x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.2S</code> <code>val.val[1] -> Vt2.2S</code> <code>val.val[0] -> Vt.2S</code>	<code>ST1 {Vt.2S - Vt3.2S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_s32_x3(int32_t * ptr, int32x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.4S</code> <code>val.val[1] -> Vt2.4S</code> <code>val.val[0] -> Vt.4S</code>	<code>ST1 {Vt.4S - Vt3.4S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_u8_x3(uint8_t * ptr, uint8x8x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.8B</code> <code>val.val[1] -> Vt2.8B</code> <code>val.val[0] -> Vt.8B</code>	<code>ST1 {Vt.8B - Vt3.8B},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u8_x3(uint8_t * ptr, uint8x16x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.16B</code> <code>val.val[1] -> Vt2.16B</code> <code>val.val[0] -> Vt.16B</code>	<code>ST1 {Vt.16B - Vt3.16B},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_u16_x3(uint16_t * ptr, uint16x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.4H</code> <code>val.val[1] -> Vt2.4H</code> <code>val.val[0] -> Vt.4H</code>	<code>ST1 {Vt.4H - Vt3.4H},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u16_x3(uint16_t * ptr, uint16x8x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.8H</code> <code>val.val[1] -> Vt2.8H</code> <code>val.val[0] -> Vt.8H</code>	<code>ST1 {Vt.8H - Vt3.8H},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_u32_x3(uint32_t * ptr, uint32x2x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.2S</code> <code>val.val[1] -> Vt2.2S</code> <code>val.val[0] -> Vt.2S</code>	<code>ST1 {Vt.2S - Vt3.2S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u32_x3(uint32_t * ptr, uint32x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.4S</code> <code>val.val[1] -> Vt2.4S</code> <code>val.val[0] -> Vt.4S</code>	<code>ST1 {Vt.4S - Vt3.4S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_f16_x3(float16_t * ptr, float16x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.4H</code> <code>val.val[1] -> Vt2.4H</code> <code>val.val[0] -> Vt.4H</code>	<code>ST1 {Vt.4H - Vt3.4H},[Xn]</code>	<code>void -> result</code>	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void vst1q_f16_x3(float16_t * ptr, float16x8x3_t val)	ptr -> Xn val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST1 {Vt.8H - Vt3.8H},[Xn]	void -> result	v7/A32/A64
void vst1_f32_x3(float32_t * ptr, float32x2x3_t val)	ptr -> Xn val.val[2] -> Vt3.2S val.val[1] -> Vt2.2S val.val[0] -> Vt.2S	ST1 {Vt.2S - Vt3.2S},[Xn]	void -> result	v7/A32/A64
void vst1q_f32_x3(float32_t * ptr, float32x4x3_t val)	ptr -> Xn val.val[2] -> Vt3.4S val.val[1] -> Vt2.4S val.val[0] -> Vt.4S	ST1 {Vt.4S - Vt3.4S},[Xn]	void -> result	v7/A32/A64
void vst1_p8_x3(poly8_t * ptr, poly8x8x3_t val)	ptr -> Xn val.val[2] -> Vt3.8B val.val[1] -> Vt2.8B val.val[0] -> Vt.8B	ST1 {Vt.8B - Vt3.8B},[Xn]	void -> result	v7/A32/A64
void vst1q_p8_x3(poly8_t * ptr, poly8x16x3_t val)	ptr -> Xn val.val[2] -> Vt3.16B val.val[1] -> Vt2.16B val.val[0] -> Vt.16B	ST1 {Vt.16B - Vt3.16B},[Xn]	void -> result	v7/A32/A64
void vst1_p16_x3(poly16_t * ptr, poly16x4x3_t val)	ptr -> Xn val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST1 {Vt.4H - Vt3.4H},[Xn]	void -> result	v7/A32/A64
void vst1q_p16_x3(poly16_t * ptr, poly16x8x3_t val)	ptr -> Xn val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST1 {Vt.8H - Vt3.8H},[Xn]	void -> result	v7/A32/A64
void vst1_s64_x3(int64_t * ptr, int64x1x3_t val)	ptr -> Xn val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D	ST1 {Vt.1D - Vt3.1D},[Xn]	void -> result	v7/A32/A64
void vst1_u64_x3(uint64_t * ptr, uint64x1x3_t val)	ptr -> Xn val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D	ST1 {Vt.1D - Vt3.1D},[Xn]	void -> result	v7/A32/A64
void vst1_p64_x3(poly64_t * ptr, poly64x1x3_t val)	ptr -> Xn val.val[2] -> Vt3.1D val.val[1] -> Vt2.1D val.val[0] -> Vt.1D	ST1 {Vt.1D - Vt3.1D},[Xn]	void -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst1q_s64_x3(int64_t * ptr, int64x2x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.2D</code> <code>val.val[1] -> Vt2.2D</code> <code>val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt3.2D}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u64_x3(uint64_t * ptr, uint64x2x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.2D</code> <code>val.val[1] -> Vt2.2D</code> <code>val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt3.2D}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_p64_x3(poly64_t * ptr, poly64x2x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.2D</code> <code>val.val[1] -> Vt2.2D</code> <code>val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt3.2D}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_f64_x3(float64_t * ptr, float64x1x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.1D</code> <code>val.val[1] -> Vt2.1D</code> <code>val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt3.1D}, [Xn]</code>	<code>void -> result</code>	A64
<code>void vst1q_f64_x3(float64_t * ptr, float64x2x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt3.2D</code> <code>val.val[1] -> Vt2.2D</code> <code>val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt3.2D}, [Xn]</code>	<code>void -> result</code>	A64
<code>void vst1_s8_x4(int8_t * ptr, int8x8x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.8B</code> <code>val.val[2] -> Vt3.8B</code> <code>val.val[1] -> Vt2.8B</code> <code>val.val[0] -> Vt.8B</code>	<code>ST1 {Vt.8B - Vt4.8B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_s8_x4(int8_t * ptr, int8x16x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.16B</code> <code>val.val[2] -> Vt3.16B</code> <code>val.val[1] -> Vt2.16B</code> <code>val.val[0] -> Vt.16B</code>	<code>ST1 {Vt.16B - Vt4.16B}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_s16_x4(int16_t * ptr, int16x4x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.4H</code> <code>val.val[2] -> Vt3.4H</code> <code>val.val[1] -> Vt2.4H</code> <code>val.val[0] -> Vt.4H</code>	<code>ST1 {Vt.4H - Vt4.4H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_s16_x4(int16_t * ptr, int16x8x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.8H</code> <code>val.val[2] -> Vt3.8H</code> <code>val.val[1] -> Vt2.8H</code> <code>val.val[0] -> Vt.8H</code>	<code>ST1 {Vt.8H - Vt4.8H}, [Xn]</code>	<code>void -> result</code>	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst1_s32_x4(int32_t * ptr, int32x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.2S</code> <code>val.val[2] -> Vt3.2S</code> <code>val.val[1] -> Vt2.2S</code> <code>val.val[0] -> Vt.2S</code>	<code>ST1 {Vt.2S - Vt4.2S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_s32_x4(int32_t * ptr, int32x4x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.4S</code> <code>val.val[2] -> Vt3.4S</code> <code>val.val[1] -> Vt2.4S</code> <code>val.val[0] -> Vt.4S</code>	<code>ST1 {Vt.4S - Vt4.4S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_u8_x4(uint8_t * ptr, uint8x8x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.8B</code> <code>val.val[2] -> Vt3.8B</code> <code>val.val[1] -> Vt2.8B</code> <code>val.val[0] -> Vt.8B</code>	<code>ST1 {Vt.8B - Vt4.8B},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u8_x4(uint8_t * ptr, uint8x16x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.16B</code> <code>val.val[2] -> Vt3.16B</code> <code>val.val[1] -> Vt2.16B</code> <code>val.val[0] -> Vt.16B</code>	<code>ST1 {Vt.16B - Vt4.16B},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_u16_x4(uint16_t * ptr, uint16x4x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.4H</code> <code>val.val[2] -> Vt3.4H</code> <code>val.val[1] -> Vt2.4H</code> <code>val.val[0] -> Vt.4H</code>	<code>ST1 {Vt.4H - Vt4.4H},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u16_x4(uint16_t * ptr, uint16x8x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.8H</code> <code>val.val[2] -> Vt3.8H</code> <code>val.val[1] -> Vt2.8H</code> <code>val.val[0] -> Vt.8H</code>	<code>ST1 {Vt.8H - Vt4.8H},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_u32_x4(uint32_t * ptr, uint32x2x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.2S</code> <code>val.val[2] -> Vt3.2S</code> <code>val.val[1] -> Vt2.2S</code> <code>val.val[0] -> Vt.2S</code>	<code>ST1 {Vt.2S - Vt4.2S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u32_x4(uint32_t * ptr, uint32x4x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.4S</code> <code>val.val[2] -> Vt3.4S</code> <code>val.val[1] -> Vt2.4S</code> <code>val.val[0] -> Vt.4S</code>	<code>ST1 {Vt.4S - Vt4.4S},[Xn]</code>	<code>void -> result</code>	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst1_f16_x4(float16_t * ptr, float16x4x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST1 {Vt.4H - Vt4.4H},[Xn]	void -> result	v7/A32/A64
<code>void vst1q_f16_x4(float16_t * ptr, float16x8x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST1 {Vt.8H - Vt4.8H},[Xn]	void -> result	v7/A32/A64
<code>void vst1_f32_x4(float32_t * ptr, float32x2x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.2S val.val[2] -> Vt3.2S val.val[1] -> Vt2.2S val.val[0] -> Vt.2S	ST1 {Vt.2S - Vt4.2S},[Xn]	void -> result	v7/A32/A64
<code>void vst1q_f32_x4(float32_t * ptr, float32x4x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.4S val.val[2] -> Vt3.4S val.val[1] -> Vt2.4S val.val[0] -> Vt.4S	ST1 {Vt.4S - Vt4.4S},[Xn]	void -> result	v7/A32/A64
<code>void vst1_p8_x4(poly8_t * ptr, poly8x8x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.8B val.val[2] -> Vt3.8B val.val[1] -> Vt2.8B val.val[0] -> Vt.8B	ST1 {Vt.8B - Vt4.8B},[Xn]	void -> result	v7/A32/A64
<code>void vst1q_p8_x4(poly8_t * ptr, poly8x16x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.16B val.val[2] -> Vt3.16B val.val[1] -> Vt2.16B val.val[0] -> Vt.16B	ST1 {Vt.16B - Vt4.16B},[Xn]	void -> result	v7/A32/A64
<code>void vst1_p16_x4(poly16_t * ptr, poly16x4x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST1 {Vt.4H - Vt4.4H},[Xn]	void -> result	v7/A32/A64
<code>void vst1q_p16_x4(poly16_t * ptr, poly16x8x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST1 {Vt.8H - Vt4.8H},[Xn]	void -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst1_s64_x4(int64_t * ptr, int64x1x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.1D</code> <code>val.val[2] -> Vt3.1D</code> <code>val.val[1] -> Vt2.1D</code> <code>val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt4.1D},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_u64_x4(uint64_t * ptr, uint64x1x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.1D</code> <code>val.val[2] -> Vt3.1D</code> <code>val.val[1] -> Vt2.1D</code> <code>val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt4.1D},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1_p64_x4(poly64_t * ptr, poly64x1x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.1D</code> <code>val.val[2] -> Vt3.1D</code> <code>val.val[1] -> Vt2.1D</code> <code>val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt4.1D},[Xn]</code>	<code>void -> result</code>	A32/A64
<code>void vst1q_s64_x4(int64_t * ptr, int64x2x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.2D</code> <code>val.val[2] -> Vt3.2D</code> <code>val.val[1] -> Vt2.2D</code> <code>val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt4.2D},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_u64_x4(uint64_t * ptr, uint64x2x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.2D</code> <code>val.val[2] -> Vt3.2D</code> <code>val.val[1] -> Vt2.2D</code> <code>val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt4.2D},[Xn]</code>	<code>void -> result</code>	v7/A32/A64
<code>void vst1q_p64_x4(poly64_t * ptr, poly64x2x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.2D</code> <code>val.val[2] -> Vt3.2D</code> <code>val.val[1] -> Vt2.2D</code> <code>val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt4.2D},[Xn]</code>	<code>void -> result</code>	A32/A64
<code>void vst1_f64_x4(float64_t * ptr, float64x1x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.1D</code> <code>val.val[2] -> Vt3.1D</code> <code>val.val[1] -> Vt2.1D</code> <code>val.val[0] -> Vt.1D</code>	<code>ST1 {Vt.1D - Vt4.1D},[Xn]</code>	<code>void -> result</code>	A64
<code>void vst1q_f64_x4(float64_t * ptr, float64x2x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt4.2D</code> <code>val.val[2] -> Vt3.2D</code> <code>val.val[1] -> Vt2.2D</code> <code>val.val[0] -> Vt.2D</code>	<code>ST1 {Vt.2D - Vt4.2D},[Xn]</code>	<code>void -> result</code>	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x8x2_t vld1_s8_x2(int8_t const * ptr)	ptr -> Xn	LD1 {Vt.8B - Vt.2.8B},[Xn]	Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
int8x16x2_t vld1q_s8_x2(int8_t const * ptr)	ptr -> Xn	LD1 {Vt.16B - Vt.2.16B},[Xn]	Vt.2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
int16x4x2_t vld1_s16_x2(int16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt.2.4H},[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x2_t vld1q_s16_x2(int16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt.2.8H},[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int32x2x2_t vld1_s32_x2(int32_t const * ptr)	ptr -> Xn	LD1 {Vt.2S - Vt.2.2S},[Xn]	Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
int32x4x2_t vld1q_s32_x2(int32_t const * ptr)	ptr -> Xn	LD1 {Vt.4S - Vt.2.4S},[Xn]	Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint8x8x2_t vld1_u8_x2(uint8_t const * ptr)	ptr -> Xn	LD1 {Vt.8B - Vt.2.8B},[Xn]	Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
uint8x16x2_t vld1q_u8_x2(uint8_t const * ptr)	ptr -> Xn	LD1 {Vt.16B - Vt.2.16B},[Xn]	Vt.2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
uint16x4x2_t vld1_u16_x2(uint16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt.2.4H},[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
uint16x8x2_t vld1q_u16_x2(uint16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt.2.8H},[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
uint32x2x2_t vld1_u32_x2(uint32_t const * ptr)	ptr -> Xn	LD1 {Vt.2S - Vt.2.2S},[Xn]	Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
uint32x4x2_t vld1q_u32_x2(uint32_t const * ptr)	ptr -> Xn	LD1 {Vt.4S - Vt.2.4S},[Xn]	Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
float16x4x2_t vld1_f16_x2(float16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt.2.4H},[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
float16x8x2_t vld1q_f16_x2(float16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt.2.8H},[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
float32x2x2_t vld1_f32_x2(float32_t const * ptr)	ptr -> Xn	LD1 {Vt.2S - Vt.2.2S},[Xn]	Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
float32x4x2_t vld1q_f32_x2(float32_t const * ptr)	ptr -> Xn	LD1 {Vt.4S - Vt.2.4S},[Xn]	Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
poly8x8x2_t vld1_p8_x2(poly8_t const * ptr)	ptr -> Xn	LD1 {Vt.8B - Vt.2.8B},[Xn]	Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
poly8x16x2_t vld1q_p8_x2(poly8_t const * ptr)	ptr -> Xn	LD1 {Vt.16B - Vt.2.16B},[Xn]	Vt.2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly16x4x2_t_vld1_p16_x2(poly16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt.2.4H},{Xn}	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
poly16x8x2_t_vld1q_p16_x2(poly16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt.2.8H},{Xn}	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int64x1x2_t_vld1_s64_x2(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.2.1D},{Xn}	Vt.2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
uint64x1x2_t_vld1_u64_x2(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.2.1D},{Xn}	Vt.2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
poly64x1x2_t_vld1_p64_x2(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.2.1D},{Xn}	Vt.2.1D -> result.val[1] Vt.1D -> result.val[0]	A32/A64
int64x2x2_t_vld1q_s64_x2(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt.2.2D},{Xn}	Vt.2.2D -> result.val[1] Vt.2D -> result.val[0]	v7/A32/A64
uint64x2x2_t_vld1q_u64_x2(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt.2.2D},{Xn}	Vt.2.2D -> result.val[1] Vt.2D -> result.val[0]	v7/A32/A64
poly64x2x2_t_vld1q_p64_x2(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt.2.2D},{Xn}	Vt.2.2D -> result.val[1] Vt.2D -> result.val[0]	A32/A64
float64x1x2_t_vld1_f64_x2(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.2.1D},{Xn}	Vt.2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
float64x2x2_t_vld1q_f64_x2(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt.2.2D},{Xn}	Vt.2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
int8x8x3_t_vld1_s8_x3(int8_t const * ptr)	ptr -> Xn	LD1 {Vt.8B - Vt.3.8B},{Xn}	Vt.3.8B -> result.val[2] Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
int8x16x3_t_vld1q_s8_x3(int8_t const * ptr)	ptr -> Xn	LD1 {Vt.16B - Vt.3.16B},{Xn}	Vt.3.16B -> result.val[2] Vt.2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
int16x4x3_t_vld1_s16_x3(int16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt.3.4H},{Xn}	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x3_t_vld1q_s16_x3(int16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt.3.8H},{Xn}	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int32x2x3_t_vld1_s32_x3(int32_t const * ptr)	ptr -> Xn	LD1 {Vt.2S - Vt.3.2S},{Xn}	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4x3_t_vld1q_s32_x3(int32_t const * ptr)	ptr -> Xn	LD1 {Vt.4S - Vt.3.4S},[Xn]	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint8x8x3_t_vld1_u8_x3(uint8_t const * ptr)	ptr -> Xn	LD1 {Vt.8B - Vt.3.8B},[Xn]	Vt.3.8B -> result.val[2] Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
uint8x16x3_t_vld1q_u8_x3(uint8_t const * ptr)	ptr -> Xn	LD1 {Vt.16B - Vt.3.16B},[Xn]	Vt.3.16B -> result.val[2] Vt.2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
uint16x4x3_t_vld1_u16_x3(uint16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt.3.4H},[Xn]	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
uint16x8x3_t_vld1q_u16_x3(uint16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt.3.8H},[Xn]	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
uint32x2x3_t_vld1_u32_x3(uint32_t const * ptr)	ptr -> Xn	LD1 {Vt.2S - Vt.3.2S},[Xn]	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
uint32x4x3_t_vld1q_u32_x3(uint32_t const * ptr)	ptr -> Xn	LD1 {Vt.4S - Vt.3.4S},[Xn]	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
float16x4x3_t_vld1_f16_x3(float16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt.3.4H},[Xn]	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
float16x8x3_t_vld1q_f16_x3(float16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt.3.8H},[Xn]	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
float32x2x3_t_vld1_f32_x3(float32_t const * ptr)	ptr -> Xn	LD1 {Vt.2S - Vt.3.2S},[Xn]	Vt.3.2S -> result.val[2] Vt.2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
float32x4x3_t_vld1q_f32_x3(float32_t const * ptr)	ptr -> Xn	LD1 {Vt.4S - Vt.3.4S},[Xn]	Vt.3.4S -> result.val[2] Vt.2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
poly8x8x3_t_vld1_p8_x3(poly8_t const * ptr)	ptr -> Xn	LD1 {Vt.8B - Vt.3.8B},[Xn]	Vt.3.8B -> result.val[2] Vt.2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly8x16x3_t_vld1q_p8_x3(poly8_t const * ptr)	ptr -> Xn	LD1 {Vt.16B - Vt3.16B},{Xn}	Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
poly16x4x3_t_vld1_p16_x3(poly16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt3.4H},{Xn}	Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
poly16x8x3_t_vld1q_p16_x3(poly16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt3.8H},{Xn}	Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int64x1x3_t_vld1_s64_x3(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt3.1D},{Xn}	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
uint64x1x3_t_vld1_u64_x3(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt3.1D},{Xn}	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
poly64x1x3_t_vld1_p64_x3(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt3.1D},{Xn}	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A32/A64
int64x2x3_t_vld1q_s64_x3(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt3.2D},{Xn}	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	v7/A32/A64
uint64x2x3_t_vld1q_u64_x3(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt3.2D},{Xn}	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	v7/A32/A64
poly64x2x3_t_vld1q_p64_x3(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt3.2D},{Xn}	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A32/A64
float64x1x3_t_vld1_f64_x3(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt3.1D},{Xn}	Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
float64x2x3_t_vld1q_f64_x3(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt3.2D},{Xn}	Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
int8x8x4_t_vld1_s8_x4(int8_t const * ptr)	ptr -> Xn	LD1 {Vt.8B - Vt4.8B},{Xn}	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16x4_t_vld1q_s8_x4(int8_t const * ptr)	ptr -> Xn	LD1 {Vt.16B - Vt4.16B},[Xn]	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
int16x4x4_t_vld1_s16_x4(int16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt4.4H},[Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
int16x8x4_t_vld1q_s16_x4(int16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt4.8H},[Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int32x2x4_t_vld1_s32_x4(int32_t const * ptr)	ptr -> Xn	LD1 {Vt.2S - Vt4.2S},[Xn]	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
int32x4x4_t_vld1q_s32_x4(int32_t const * ptr)	ptr -> Xn	LD1 {Vt.4S - Vt4.4S},[Xn]	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
uint8x8x4_t_vld1_u8_x4(uint8_t const * ptr)	ptr -> Xn	LD1 {Vt.8B - Vt4.8B},[Xn]	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
uint8x16x4_t_vld1q_u8_x4(uint8_t const * ptr)	ptr -> Xn	LD1 {Vt.16B - Vt4.16B},[Xn]	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
uint16x4x4_t_vld1_u16_x4(uint16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt4.4H},[Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
uint16x8x4_t_vld1q_u16_x4(uint16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt4.8H},[Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>uint32x2x4_t vld1_u32_x4(uint32_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD1 {Vt.2S - Vt.4.2S},[Xn]</code>	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
<code>uint32x4x4_t vld1q_u32_x4(uint32_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD1 {Vt.4S - Vt.4.4S},[Xn]</code>	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
<code>float16x4x4_t vld1_f16_x4(float16_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD1 {Vt.4H - Vt.4.4H},[Xn]</code>	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
<code>float16x8x4_t vld1q_f16_x4(float16_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD1 {Vt.8H - Vt.4.8H},[Xn]</code>	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
<code>float32x2x4_t vld1_f32_x4(float32_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD1 {Vt.2S - Vt.4.2S},[Xn]</code>	Vt4.2S -> result.val[3] Vt3.2S -> result.val[2] Vt2.2S -> result.val[1] Vt.2S -> result.val[0]	v7/A32/A64
<code>float32x4x4_t vld1q_f32_x4(float32_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD1 {Vt.4S - Vt.4.4S},[Xn]</code>	Vt4.4S -> result.val[3] Vt3.4S -> result.val[2] Vt2.4S -> result.val[1] Vt.4S -> result.val[0]	v7/A32/A64
<code>poly8x8x4_t vld1_p8_x4(poly8_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD1 {Vt.8B - Vt.4.8B},[Xn]</code>	Vt4.8B -> result.val[3] Vt3.8B -> result.val[2] Vt2.8B -> result.val[1] Vt.8B -> result.val[0]	v7/A32/A64
<code>poly8x16x4_t vld1q_p8_x4(poly8_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD1 {Vt.16B - Vt.4.16B},[Xn]</code>	Vt4.16B -> result.val[3] Vt3.16B -> result.val[2] Vt2.16B -> result.val[1] Vt.16B -> result.val[0]	v7/A32/A64
<code>poly16x4x4_t vld1_p16_x4(poly16_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD1 {Vt.4H - Vt.4.4H},[Xn]</code>	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly16x8x4_t vld1q_p16_x4(poly16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt.4.8H},{Xn}	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
int64x1x4_t vld1_s64_x4(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.4.1D},{Xn}	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
uint64x1x4_t vld1_u64_x4(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.4.1D},{Xn}	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	v7/A32/A64
poly64x1x4_t vld1_p64_x4(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.4.1D},{Xn}	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A32/A64
int64x2x4_t vld1q_s64_x4(int64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt.4.2D},{Xn}	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	v7/A32/A64
uint64x2x4_t vld1q_u64_x4(uint64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt.4.2D},{Xn}	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	v7/A32/A64
poly64x2x4_t vld1q_p64_x4(poly64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt.4.2D},{Xn}	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A32/A64
float64x1x4_t vld1_f64_x4(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.1D - Vt.4.1D},{Xn}	Vt4.1D -> result.val[3] Vt3.1D -> result.val[2] Vt2.1D -> result.val[1] Vt.1D -> result.val[0]	A64
float64x2x4_t vld1q_f64_x4(float64_t const * ptr)	ptr -> Xn	LD1 {Vt.2D - Vt.4.2D},{Xn}	Vt4.2D -> result.val[3] Vt3.2D -> result.val[2] Vt2.2D -> result.val[1] Vt.2D -> result.val[0]	A64
int8x8_t vpadd_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	ADDP Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x4_t vpadd_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	ADDP Vd.4H, Vn.4H, Vm.4H	Vd.4H -> result	v7/A32/A64
int32x2_t vpadd_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	ADDP Vd.2S, Vn.2S, Vm.2S	Vd.2S -> result	v7/A32/A64
uint8x8_t vpadd_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	ADDP Vd.8B, Vn.8B, Vm.8B	Vd.8B -> result	v7/A32/A64
uint16x4_t vpadd_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	ADDP Vd.4H, Vn.4H, Vm.4H	Vd.4H -> result	v7/A32/A64
uint32x2_t vpadd_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	ADDP Vd.2S, Vn.2S, Vm.2S	Vd.2S -> result	v7/A32/A64
float32x2_t vpadd_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FADDP Vd.2S, Vn.2S, Vm.2S	Vd.2S -> result	v7/A32/A64
int8x16_t vpaddq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	ADDP Vd.16B, Vn.16B, Vm.16B	Vd.16B -> result	A64
int16x8_t vpaddq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	ADDP Vd.8H, Vn.8H, Vm.8H	Vd.8H -> result	A64
int32x4_t vpaddq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	ADDP Vd.4S, Vn.4S, Vm.4S	Vd.4S -> result	A64
int64x2_t vpaddq_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	ADDP Vd.2D, Vn.2D, Vm.2D	Vd.2D -> result	A64
uint8x16_t vpaddq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	ADDP Vd.16B, Vn.16B, Vm.16B	Vd.16B -> result	A64
uint16x8_t vpaddq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	ADDP Vd.8H, Vn.8H, Vm.8H	Vd.8H -> result	A64
uint32x4_t vpaddq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	ADDP Vd.4S, Vn.4S, Vm.4S	Vd.4S -> result	A64
uint64x2_t vpaddq_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	ADDP Vd.2D, Vn.2D, Vm.2D	Vd.2D -> result	A64
float32x4_t vpaddq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FADDP Vd.4S, Vn.4S, Vm.4S	Vd.4S -> result	A64
float64x2_t vpaddq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FADDP Vd.2D, Vn.2D, Vm.2D	Vd.2D -> result	A64
int16x4_t vpaddl_s8(int8x8_t a)	a -> Vn.8B	SADDLP Vd.4H, Vn.8B	Vd.4H -> result	v7/A32/A64
int16x8_t vpaddlq_s8(int8x16_t a)	a -> Vn.16B	SADDLP Vd.8H, Vn.16B	Vd.8H -> result	v7/A32/A64
int32x2_t vpaddl_s16(int16x4_t a)	a -> Vn.4H	SADDLP Vd.2S, Vn.4H	Vd.2S -> result	v7/A32/A64
int32x4_t vpaddlq_s16(int16x8_t a)	a -> Vn.8H	SADDLP Vd.4S, Vn.8H	Vd.4S -> result	v7/A32/A64
int64x1_t vpaddl_s32(int32x2_t a)	a -> Vn.2S	SADDLP Vd.1D, Vn.2S	Vd.1D -> result	v7/A32/A64
int64x2_t vpaddlq_s32(int32x4_t a)	a -> Vn.4S	SADDLP Vd.2D, Vn.4S	Vd.2D -> result	v7/A32/A64
uint16x4_t vpaddl_u8(uint8x8_t a)	a -> Vn.8B	UADDLP Vd.4H, Vn.8B	Vd.4H -> result	v7/A32/A64
uint16x8_t vpaddlq_u8(uint8x16_t a)	a -> Vn.16B	UADDLP Vd.8H, Vn.16B	Vd.8H -> result	v7/A32/A64
uint32x2_t vpaddl_u16(uint16x4_t a)	a -> Vn.4H	UADDLP Vd.2S, Vn.4H	Vd.2S -> result	v7/A32/A64
uint32x4_t vpaddlq_u16(uint16x8_t a)	a -> Vn.8H	UADDLP Vd.4S, Vn.8H	Vd.4S -> result	v7/A32/A64
uint64x1_t vpaddl_u32(uint32x2_t a)	a -> Vn.2S	UADDLP Vd.1D, Vn.2S	Vd.1D -> result	v7/A32/A64
uint64x2_t vpaddlq_u32(uint32x4_t a)	a -> Vn.4S	UADDLP Vd.2D, Vn.4S	Vd.2D -> result	v7/A32/A64
int16x4_t vpadal_s8(int16x4_t a, int8x8_t b)	a -> Vd.4H b -> Vn.8B	SADALP Vd.4H, Vn.8B	Vd.4H -> result	v7/A32/A64
int16x8_t vpadalq_s8(int16x8_t a, int8x16_t b)	a -> Vd.8H b -> Vn.16B	SADALP Vd.8H, Vn.16B	Vd.8H -> result	v7/A32/A64
int32x2_t vpadal_s16(int32x2_t a, int16x4_t b)	a -> Vd.2S b -> Vn.4H	SADALP Vd.2S, Vn.4H	Vd.2S -> result	v7/A32/A64
int32x4_t vpadalq_s16(int32x4_t a, int16x8_t b)	a -> Vd.4S b -> Vn.8H	SADALP Vd.4S, Vn.8H	Vd.4S -> result	v7/A32/A64
int64x1_t vpadal_s32(int64x1_t a, int32x2_t b)	a -> Vd.1D b -> Vn.2S	SADALP Vd.1D, Vn.2S	Vd.1D -> result	v7/A32/A64
int64x2_t vpadalq_s32(int64x2_t a, int32x4_t b)	a -> Vd.2D b -> Vn.4S	SADALP Vd.2D, Vn.4S	Vd.2D -> result	v7/A32/A64
uint16x4_t vpadal_u8(uint16x4_t a, uint8x8_t b)	a -> Vd.4H b -> Vn.8B	UADALP Vd.4H, Vn.8B	Vd.4H -> result	v7/A32/A64
uint16x8_t vpadalq_u8(uint16x8_t a, uint8x16_t b)	a -> Vd.8H b -> Vn.16B	UADALP Vd.8H, Vn.16B	Vd.8H -> result	v7/A32/A64
uint32x2_t vpadal_u16(uint32x2_t a, uint16x4_t b)	a -> Vd.2S b -> Vn.4H	UADALP Vd.2S, Vn.4H	Vd.2S -> result	v7/A32/A64
uint32x4_t vpadalq_u16(uint32x4_t a, uint16x8_t b)	a -> Vd.4S b -> Vn.8H	UADALP Vd.4S, Vn.8H	Vd.4S -> result	v7/A32/A64
uint64x1_t vpadal_u32(uint64x1_t a, uint32x2_t b)	a -> Vd.1D b -> Vn.2S	UADALP Vd.1D, Vn.2S	Vd.1D -> result	v7/A32/A64
uint64x2_t vpadalq_u32(uint64x2_t a, uint32x4_t b)	a -> Vd.2D b -> Vn.4S	UADALP Vd.2D, Vn.4S	Vd.2D -> result	v7/A32/A64
int8x8_t vpmask_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SMAXP Vd.8B, Vn.8B, Vm.8B	Vd.8B -> result	v7/A32/A64
int16x4_t vpmask_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SMAXP Vd.4H, Vn.4H, Vm.4H	Vd.4H -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x2_t vpmmax_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SMAXP Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint8x8_t vpmmax_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UMAXP Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint16x4_t vpmmax_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UMAXP Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint32x2_t vpmmax_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UMAXP Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
float32x2_t vpmmax_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FMAXP Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int8x16_t vpmmaxq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SMAXP Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
int16x8_t vpmmaxq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SMAXP Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int32x4_t vpmmaxq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SMAXP Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
uint8x16_t vpmmaxq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UMAXP Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
uint16x8_t vpmmaxq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UMAXP Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
uint32x4_t vpmmaxq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UMAXP Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float32x4_t vpmmaxq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FMAXP Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x2_t vpmmaxq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FMAXP Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
int8x8_t vpmmin_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	SMINP Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
int16x4_t vpmmin_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	SMINP Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
int32x2_t vpmmin_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	SMINP Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
uint8x8_t vpmmin_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UMINP Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
uint16x4_t vpmmin_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UMINP Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	v7/A32/A64
uint32x2_t vpmmin_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UMINP Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
float32x2_t vpmmin_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FMINP Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	v7/A32/A64
int8x16_t vpmminq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	SMINP Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
int16x8_t vpmminq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	SMINP Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int32x4_t vpmminq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	SMINP Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
uint8x16_t vpmminq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UMINP Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
uint16x8_t vpmminq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UMINP Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
uint32x4_t vpmminq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UMINP Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float32x4_t vpmminq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FMINP Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x2_t vpmminq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FMINP Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t vpmmaxnm_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FMAXNMP Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
float32x4_t vpmmaxnmq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FMAXNMP Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x2_t vpmmaxnmq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FMAXNMP Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t vpmminnm_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FMINNMP Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
float32x4_t vpmminnmq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FMINNMP Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x2_t vpmminnmq_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FMINNMP Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
int64_t vpaddd_s64(int64x2_t a)	a -> Vn.2D	ADDP Dd,Vn.2D	Dd -> result	A64
uint64_t vpaddd_u64(uint64x2_t a)	a -> Vn.2D	ADDP Dd,Vn.2D	Dd -> result	A64
float32_t vpaddd_s32(float32x2_t a)	a -> Vn.2S	FADDP Sd,Vn.2S	Sd -> result	A64
float64_t vpaddd_f64(float64x2_t a)	a -> Vn.2D	FADDP Dd,Vn.2D	Dd -> result	A64
float32_t vpmmaxs_f32(float32x2_t a)	a -> Vn.2S	FMAXP Sd,Vn.2S	Sd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64_t vpmxq f64(float64x2_t a)	a -> Vn.2D	FMAXP Dd,Vn.2D	Dd -> result	A64
float32_t vpmins f32(float32x2_t a)	a -> Vn.2S	FMINP Sd,Vn.2S	Sd -> result	A64
float64_t vpminq f64(float64x2_t a)	a -> Vn.2D	FMINP Dd,Vn.2D	Dd -> result	A64
float32_t vpmxnms f32(float32x2_t a)	a -> Vn.2S	FMAXNMP Sd,Vn.2S	Sd -> result	A64
float64_t vpmxnmq f64(float64x2_t a)	a -> Vn.2D	FMAXNMP Dd,Vn.2D	Dd -> result	A64
float32_t vpminnms f32(float32x2_t a)	a -> Vn.2S	FMINNMP Sd,Vn.2S	Sd -> result	A64
float64_t vpminnmq f64(float64x2_t a)	a -> Vn.2D	FMINNMP Dd,Vn.2D	Dd -> result	A64
int8_t vaddv_s8(int8x8_t a)	a -> Vn.8B	ADDV Bd,Vn.8B	Bd -> result	A64
int8_t vaddvq_s8(int8x16_t a)	a -> Vn.16B	ADDV Bd,Vn.16B	Bd -> result	A64
int16_t vaddv_s16(int16x4_t a)	a -> Vn.4H	ADDV Hd,Vn.4H	Hd -> result	A64
int16_t vaddvq_s16(int16x8_t a)	a -> Vn.8H	ADDV Hd,Vn.8H	Hd -> result	A64
int32_t vaddv_s32(int32x2_t a)	a -> Vn.2S a -> Vm.2S	ADDP Vd.2S,Vn.2S,Vm.2S	Vd.S[0] -> result	A64
int32_t vaddvq_s32(int32x4_t a)	a -> Vn.4S	ADDV Sd,Vn.4S	Sd -> result	A64
int64_t vaddvq_s64(int64x2_t a)	a -> Vn.2D	ADDP Dd,Vn.2D	Dd -> result	A64
uint8_t vaddv_u8(uint8x8_t a)	a -> Vn.8B	ADDV Bd,Vn.8B	Bd -> result	A64
uint8_t vaddvq_u8(uint8x16_t a)	a -> Vn.16B	ADDV Bd,Vn.16B	Bd -> result	A64
uint16_t vaddv_u16(uint16x4_t a)	a -> Vn.4H	ADDV Hd,Vn.4H	Hd -> result	A64
uint16_t vaddvq_u16(uint16x8_t a)	a -> Vn.8H	ADDV Hd,Vn.8H	Hd -> result	A64
uint32_t vaddv_u32(uint32x2_t a)	a -> Vn.2S a -> Vm.2S	ADDP Vd.2S,Vn.2S,Vm.2S	Vd.S[0] -> result	A64
uint32_t vaddvq_u32(uint32x4_t a)	a -> Vn.4S	ADDV Sd,Vn.4S	Sd -> result	A64
uint64_t vaddvq_u64(uint64x2_t a)	a -> Vn.2D	ADDP Dd,Vn.2D	Dd -> result	A64
float32_t vaddv_f32(float32x2_t a)	a -> Vn.2S	FADDP Sd,Vn.2S	Sd -> result	A64
float32_t vaddvq_f32(float32x4_t a)	a -> Vn.4S a -> Vm.4S	FADDP Vt.4S,Vn.4S,Vm.4S FADDP Sd,Vt.2S	Sd -> result	A64
float64_t vaddvq_f64(float64x2_t a)	a -> Vn.2D	FADDP Dd,Vn.2D	Dd -> result	A64
int16_t vaddlv_s8(int8x8_t a)	a -> Vn.8B	SADDLV Hd,Vn.8B	Hd -> result	A64
int16_t vaddlvq_s8(int8x16_t a)	a -> Vn.16B	SADDLV Hd,Vn.16B	Hd -> result	A64
int32_t vaddlv_s16(int16x4_t a)	a -> Vn.4H	SADDLV Sd,Vn.4H	Sd -> result	A64
int32_t vaddlvq_s16(int16x8_t a)	a -> Vn.8H	SADDLV Sd,Vn.8H	Sd -> result	A64
int64_t vaddlv_s32(int32x2_t a)	a -> Vn.2S	SADDLP Vd.1D,Vn.2S	Dd -> result	A64
int64_t vaddlvq_s32(int32x4_t a)	a -> Vn.4S	SADDLV Dd,Vn.4S	Dd -> result	A64
uint16_t vaddlv_u8(uint8x8_t a)	a -> Vn.8B	UADDLV Hd,Vn.8B	Hd -> result	A64
uint16_t vaddlvq_u8(uint8x16_t a)	a -> Vn.16B	UADDLV Hd,Vn.16B	Hd -> result	A64
uint32_t vaddlv_u16(uint16x4_t a)	a -> Vn.4H	UADDLV Sd,Vn.4H	Sd -> result	A64
uint32_t vaddlvq_u16(uint16x8_t a)	a -> Vn.8H	UADDLV Sd,Vn.8H	Sd -> result	A64
uint64_t vaddlv_u32(uint32x2_t a)	a -> Vn.2S	UADDLP Vd.1D,Vn.2S	Dd -> result	A64
uint64_t vaddlvq_u32(uint32x4_t a)	a -> Vn.4S	UADDLV Dd,Vn.4S	Dd -> result	A64
int8_t vmaxv_s8(int8x8_t a)	a -> Vn.8B	SMAXV Bd,Vn.8B	Bd -> result	A64
int8_t vmaxvq_s8(int8x16_t a)	a -> Vn.16B	SMAXV Bd,Vn.16B	Bd -> result	A64
int16_t vmaxv_s16(int16x4_t a)	a -> Vn.4H	SMAXV Hd,Vn.4H	Hd -> result	A64
int16_t vmaxvq_s16(int16x8_t a)	a -> Vn.8H	SMAXV Hd,Vn.8H	Hd -> result	A64
int32_t vmaxv_s32(int32x2_t a)	a -> Vn.2S a -> Vm.2S	SMAXP Vd.2S,Vn.2S,Vm.2S	Vd.S[0] -> result	A64
int32_t vmaxvq_s32(int32x4_t a)	a -> Vn.4S	SMAXV Sd,Vn.4S	Sd -> result	A64
uint8_t vmaxv_u8(uint8x8_t a)	a -> Vn.8B	UMAXV Bd,Vn.8B	Bd -> result	A64
uint8_t vmaxvq_u8(uint8x16_t a)	a -> Vn.16B	UMAXV Bd,Vn.16B	Bd -> result	A64
uint16_t vmaxv_u16(uint16x4_t a)	a -> Vn.4H	UMAXV Hd,Vn.4H	Hd -> result	A64
uint16_t vmaxvq_u16(uint16x8_t a)	a -> Vn.8H	UMAXV Hd,Vn.8H	Hd -> result	A64
uint32_t vmaxv_u32(uint32x2_t a)	a -> Vn.2S a -> Vm.2S	UMAXP Vd.2S,Vn.2S,Vm.2S	Vd.S[0] -> result	A64
uint32_t vmaxvq_u32(uint32x4_t a)	a -> Vn.4S	UMAXV Sd,Vn.4S	Sd -> result	A64
float32_t vmaxv_f32(float32x2_t a)	a -> Vn.2S	FMAXP Sd,Vn.2S	Sd -> result	A64
float32_t vmaxvq_f32(float32x4_t a)	a -> Vn.4S	FMAXV Sd,Vn.4S	Sd -> result	A64
float64_t vmaxvq_f64(float64x2_t a)	a -> Vn.2D	FMAXP Dd,Vn.2D	Dd -> result	A64
int8_t vminv_s8(int8x8_t a)	a -> Vn.8B	SMINV Bd,Vn.8B	Bd -> result	A64
int8_t vminvq_s8(int8x16_t a)	a -> Vn.16B	SMINV Bd,Vn.16B	Bd -> result	A64
int16_t vminv_s16(int16x4_t a)	a -> Vn.4H	SMINV Hd,Vn.4H	Hd -> result	A64
int16_t vminvq_s16(int16x8_t a)	a -> Vn.8H	SMINV Hd,Vn.8H	Hd -> result	A64
int32_t vminv_s32(int32x2_t a)	a -> Vn.2S a -> Vm.2S	SMINP Vd.2S,Vn.2S,Vm.2S	Vd.S[0] -> result	A64
int32_t vminvq_s32(int32x4_t a)	a -> Vn.4S	SMINV Sd,Vn.4S	Sd -> result	A64
uint8_t vminv_u8(uint8x8_t a)	a -> Vn.8B	UMINV Bd,Vn.8B	Bd -> result	A64
uint8_t vminvq_u8(uint8x16_t a)	a -> Vn.16B	UMINV Bd,Vn.16B	Bd -> result	A64
uint16_t vminv_u16(uint16x4_t a)	a -> Vn.4H	UMINV Hd,Vn.4H	Hd -> result	A64
uint16_t vminvq_u16(uint16x8_t a)	a -> Vn.8H	UMINV Hd,Vn.8H	Hd -> result	A64
uint32_t vminv_u32(uint32x2_t a)	a -> Vn.2S a -> Vm.2S	UMINP Vd.2S,Vn.2S,Vm.2S	Vd.S[0] -> result	A64
uint32_t vminvq_u32(uint32x4_t a)	a -> Vn.4S	UMINV Sd,Vn.4S	Sd -> result	A64
float32_t vminv_f32(float32x2_t a)	a -> Vn.2S	FMINP Sd,Vn.2S	Sd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32_t vminvq_f32(float32x4_t a)	a -> Vn.4S	FMINV Sd,Vn.4S	Sd -> result	A64
float64_t vminvq_f64(float64x2_t a)	a -> Vn.2D	FMINP Dd,Vn.2D	Dd -> result	A64
float32_t vmaxnmv_f32(float32x2_t a)	a -> Vn.2S	FMAXNMP Sd,Vn.2S	Sd -> result	A64
float32_t vmaxnmvq_f32(float32x4_t a)	a -> Vn.4S	FMAXNMV Sd,Vn.4S	Sd -> result	A64
float64_t vmaxnmvq_f64(float64x2_t a)	a -> Vn.2D	FMAXNMP Dd,Vn.2D	Dd -> result	A64
float32_t vminnmv_f32(float32x2_t a)	a -> Vn.2S	FMINNMP Sd,Vn.2S	Sd -> result	A64
float32_t vminnmvq_f32(float32x4_t a)	a -> Vn.4S	FMINNMV Sd,Vn.4S	Sd -> result	A64
float64_t vminnmvq_f64(float64x2_t a)	a -> Vn.2D	FMINNMP Dd,Vn.2D	Dd -> result	A64
int8x8_t vext_s8(int8x8_t a, int8x8_t b, const int n)	a -> Vn.8B b -> Vm.8B 0 <= n <= 7	EXT Vd.8B,Vn.8B,Vm.8B,#n	Vd.8B -> result	v7/A32/A64
int8x16_t vextq_s8(int8x16_t a, int8x16_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 15	EXT Vd.16B,Vn.16B,Vm.16B,#n	Vd.16B -> result	v7/A32/A64
int16x4_t vext_s16(int16x4_t a, int16x4_t b, const int n)	a -> Vn.8B b -> Vm.8B 0 <= n <= 3	EXT Vd.8B,Vn.8B,Vm.8B,#(n<<1)	Vd.8B -> result	v7/A32/A64
int16x8_t vextq_s16(int16x8_t a, int16x8_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 7	EXT Vd.16B,Vn.16B,Vm.16B,#(n<<1)	Vd.16B -> result	v7/A32/A64
int32x2_t vext_s32(int32x2_t a, int32x2_t b, const int n)	a -> Vn.8B b -> Vm.8B 0 <= n <= 1	EXT Vd.8B,Vn.8B,Vm.8B,#(n<<2)	Vd.8B -> result	v7/A32/A64
int32x4_t vextq_s32(int32x4_t a, int32x4_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 3	EXT Vd.16B,Vn.16B,Vm.16B,#(n<<2)	Vd.16B -> result	v7/A32/A64
int64x1_t vext_s64(int64x1_t a, int64x1_t b, const int n)	a -> Vn.8B b -> Vm.8B n == 0	EXT Vd.8B,Vn.8B,Vm.8B,#(n<<3)	Vd.8B -> result	v7/A32/A64
int64x2_t vextq_s64(int64x2_t a, int64x2_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 1	EXT Vd.16B,Vn.16B,Vm.16B,#(n<<3)	Vd.16B -> result	v7/A32/A64
uint8x8_t vext_u8(uint8x8_t a, uint8x8_t b, const int n)	a -> Vn.8B b -> Vm.8B 0 <= n <= 7	EXT Vd.8B,Vn.8B,Vm.8B,#n	Vd.8B -> result	v7/A32/A64
uint8x16_t vextq_u8(uint8x16_t a, uint8x16_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 15	EXT Vd.16B,Vn.16B,Vm.16B,#n	Vd.16B -> result	v7/A32/A64
uint16x4_t vext_u16(uint16x4_t a, uint16x4_t b, const int n)	a -> Vn.8B b -> Vm.8B 0 <= n <= 3	EXT Vd.8B,Vn.8B,Vm.8B,#(n<<1)	Vd.8B -> result	v7/A32/A64
uint16x8_t vextq_u16(uint16x8_t a, uint16x8_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 7	EXT Vd.16B,Vn.16B,Vm.16B,#(n<<1)	Vd.16B -> result	v7/A32/A64
uint32x2_t vext_u32(uint32x2_t a, uint32x2_t b, const int n)	a -> Vn.8B b -> Vm.8B 0 <= n <= 1	EXT Vd.8B,Vn.8B,Vm.8B,#(n<<2)	Vd.8B -> result	v7/A32/A64
uint32x4_t vextq_u32(uint32x4_t a, uint32x4_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 3	EXT Vd.16B,Vn.16B,Vm.16B,#(n<<2)	Vd.16B -> result	v7/A32/A64
uint64x1_t vext_u64(uint64x1_t a, uint64x1_t b, const int n)	a -> Vn.8B b -> Vm.8B n == 0	EXT Vd.8B,Vn.8B,Vm.8B,#(n<<3)	Vd.8B -> result	v7/A32/A64
uint64x2_t vextq_u64(uint64x2_t a, uint64x2_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 1	EXT Vd.16B,Vn.16B,Vm.16B,#(n<<3)	Vd.16B -> result	v7/A32/A64
poly64x1_t vext_p64(poly64x1_t a, poly64x1_t b, const int n)	a -> Vn.8B b -> Vm.8B n == 0	EXT Vd.8B,Vn.8B,Vm.8B,#(n<<3)	Vd.8B -> result	A32/A64
poly64x2_t vextq_p64(poly64x2_t a, poly64x2_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 1	EXT Vd.16B,Vn.16B,Vm.16B,#(n<<3)	Vd.16B -> result	A32/A64
float32x2_t vext_f32(float32x2_t a, float32x2_t b, const int n)	a -> Vn.8B b -> Vm.8B 0 <= n <= 1	EXT Vd.8B,Vn.8B,Vm.8B,#(n<<2)	Vd.8B -> result	v7/A32/A64
float32x4_t vextq_f32(float32x4_t a, float32x4_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 3	EXT Vd.16B,Vn.16B,Vm.16B,#(n<<2)	Vd.16B -> result	v7/A32/A64
float64x1_t vext_f64(float64x1_t a, float64x1_t b, const int n)	a -> Vn.8B b -> Vm.8B n == 0	EXT Vd.8B,Vn.8B,Vm.8B,#(n<<3)	Vd.8B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64x2_t vextq_f64(float64x2_t a, float64x2_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 1	EXT Vd.16B, Vn.16B, Vm.16B, #n (< 3)	Vd.16B -> result	A64
poly8x8_t vext_p8(poly8x8_t a, poly8x8_t b, const int n)	a -> Vn.8B b -> Vm.8B 0 <= n <= 7	EXT Vd.8B, Vn.8B, Vm.8B, #n	Vd.8B -> result	v7/A32/A64
poly8x16_t vextq_p8(poly8x16_t a, poly8x16_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 15	EXT Vd.16B, Vn.16B, Vm.16B, #n	Vd.16B -> result	v7/A32/A64
poly16x4_t vext_p16(poly16x4_t a, poly16x4_t b, const int n)	a -> Vn.8B b -> Vm.8B 0 <= n <= 3	EXT Vd.8B, Vn.8B, Vm.8B, #n (< 4)	Vd.8B -> result	v7/A32/A64
poly16x8_t vextq_p16(poly16x8_t a, poly16x8_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 7	EXT Vd.16B, Vn.16B, Vm.16B, #n (< 8)	Vd.16B -> result	v7/A32/A64
int8x8_t vrev64_s8(int8x8_t vec)	vec -> Vn.8B	REV64 Vd.8B, Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vrev64q_s8(int8x16_t vec)	vec -> Vn.16B	REV64 Vd.16B, Vn.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vrev64_s16(int16x4_t vec)	vec -> Vn.4H	REV64 Vd.4H, Vn.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vrev64q_s16(int16x8_t vec)	vec -> Vn.8H	REV64 Vd.8H, Vn.8H	Vd.8H -> result	v7/A32/A64
int32x2_t vrev64_s32(int32x2_t vec)	vec -> Vn.2S	REV64 Vd.2S, Vn.2S	Vd.2S -> result	v7/A32/A64
int32x4_t vrev64q_s32(int32x4_t vec)	vec -> Vn.4S	REV64 Vd.4S, Vn.4S	Vd.4S -> result	v7/A32/A64
uint8x8_t vrev64_u8(uint8x8_t vec)	vec -> Vn.8B	REV64 Vd.8B, Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vrev64q_u8(uint8x16_t vec)	vec -> Vn.16B	REV64 Vd.16B, Vn.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vrev64_u16(uint16x4_t vec)	vec -> Vn.4H	REV64 Vd.4H, Vn.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vrev64q_u16(uint16x8_t vec)	vec -> Vn.8H	REV64 Vd.8H, Vn.8H	Vd.8H -> result	v7/A32/A64
uint32x2_t vrev64_u32(uint32x2_t vec)	vec -> Vn.2S	REV64 Vd.2S, Vn.2S	Vd.2S -> result	v7/A32/A64
uint32x4_t vrev64q_u32(uint32x4_t vec)	vec -> Vn.4S	REV64 Vd.4S, Vn.4S	Vd.4S -> result	v7/A32/A64
float32x2_t vrev64_f32(float32x2_t vec)	vec -> Vn.2S	REV64 Vd.2S, Vn.2S	Vd.2S -> result	v7/A32/A64
float32x4_t vrev64q_f32(float32x4_t vec)	vec -> Vn.4S	REV64 Vd.4S, Vn.4S	Vd.4S -> result	v7/A32/A64
poly8x8_t vrev64_p8(poly8x8_t vec)	vec -> Vn.8B	REV64 Vd.8B, Vn.8B	Vd.8B -> result	v7/A32/A64
poly8x16_t vrev64q_p8(poly8x16_t vec)	vec -> Vn.16B	REV64 Vd.16B, Vn.16B	Vd.16B -> result	v7/A32/A64
poly16x4_t vrev64_p16(poly16x4_t vec)	vec -> Vn.4H	REV64 Vd.4H, Vn.4H	Vd.4H -> result	v7/A32/A64
poly16x8_t vrev64q_p16(poly16x8_t vec)	vec -> Vn.8H	REV64 Vd.8H, Vn.8H	Vd.8H -> result	v7/A32/A64
int8x8_t vrev32_s8(int8x8_t vec)	vec -> Vn.8B	REV32 Vd.8B, Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vrev32q_s8(int8x16_t vec)	vec -> Vn.16B	REV32 Vd.16B, Vn.16B	Vd.16B -> result	v7/A32/A64
int16x4_t vrev32_s16(int16x4_t vec)	vec -> Vn.4H	REV32 Vd.4H, Vn.4H	Vd.4H -> result	v7/A32/A64
int16x8_t vrev32q_s16(int16x8_t vec)	vec -> Vn.8H	REV32 Vd.8H, Vn.8H	Vd.8H -> result	v7/A32/A64
uint8x8_t vrev32_u8(uint8x8_t vec)	vec -> Vn.8B	REV32 Vd.8B, Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vrev32q_u8(uint8x16_t vec)	vec -> Vn.16B	REV32 Vd.16B, Vn.16B	Vd.16B -> result	v7/A32/A64
uint16x4_t vrev32_u16(uint16x4_t vec)	vec -> Vn.4H	REV32 Vd.4H, Vn.4H	Vd.4H -> result	v7/A32/A64
uint16x8_t vrev32q_u16(uint16x8_t vec)	vec -> Vn.8H	REV32 Vd.8H, Vn.8H	Vd.8H -> result	v7/A32/A64
poly8x8_t vrev32_p8(poly8x8_t vec)	vec -> Vn.8B	REV32 Vd.8B, Vn.8B	Vd.8B -> result	v7/A32/A64
poly8x16_t vrev32q_p8(poly8x16_t vec)	vec -> Vn.16B	REV32 Vd.16B, Vn.16B	Vd.16B -> result	v7/A32/A64
poly16x4_t vrev32_p16(poly16x4_t vec)	vec -> Vn.4H	REV32 Vd.4H, Vn.4H	Vd.4H -> result	v7/A32/A64
poly16x8_t vrev32q_p16(poly16x8_t vec)	vec -> Vn.8H	REV32 Vd.8H, Vn.8H	Vd.8H -> result	v7/A32/A64
int8x8_t vrev16_s8(int8x8_t vec)	vec -> Vn.8B	REV16 Vd.8B, Vn.8B	Vd.8B -> result	v7/A32/A64
int8x16_t vrev16q_s8(int8x16_t vec)	vec -> Vn.16B	REV16 Vd.16B, Vn.16B	Vd.16B -> result	v7/A32/A64
uint8x8_t vrev16_u8(uint8x8_t vec)	vec -> Vn.8B	REV16 Vd.8B, Vn.8B	Vd.8B -> result	v7/A32/A64
uint8x16_t vrev16q_u8(uint8x16_t vec)	vec -> Vn.16B	REV16 Vd.16B, Vn.16B	Vd.16B -> result	v7/A32/A64
poly8x8_t vrev16_p8(poly8x8_t vec)	vec -> Vn.8B	REV16 Vd.8B, Vn.8B	Vd.8B -> result	v7/A32/A64
poly8x16_t vrev16q_p8(poly8x16_t vec)	vec -> Vn.16B	REV16 Vd.16B, Vn.16B	Vd.16B -> result	v7/A32/A64
int8x8_t vzip1_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	ZIP1 Vd.8B, Vn.8B, Vm.8B	Vd.8B -> result	A64
int8x16_t vzip1q_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	ZIP1 Vd.16B, Vn.16B, Vm.16B	Vd.16B -> result	A64
int16x4_t vzip1_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP1 Vd.4H, Vn.4H, Vm.4H	Vd.4H -> result	A64
int16x8_t vzip1q_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP1 Vd.8H, Vn.8H, Vm.8H	Vd.8H -> result	A64
int32x2_t vzip1_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	ZIP1 Vd.2S, Vn.2S, Vm.2S	Vd.2S -> result	A64
int32x4_t vzip1q_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	ZIP1 Vd.4S, Vn.4S, Vm.4S	Vd.4S -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64x2_t vzip1q_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	ZIP1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint8x8_t vzip1_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	ZIP1 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
uint8x16_t vzip1q_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	ZIP1 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
uint16x4_t vzip1_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP1 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
uint16x8_t vzip1q_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP1 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
uint32x2_t vzip1_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	ZIP1 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
uint32x4_t vzip1q_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	ZIP1 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
uint64x2_t vzip1q_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	ZIP1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly64x2_t vzip1q_p64(poly64x2_t a, poly64x2_t b)	a -> Vn.2D b -> Vm.2D	ZIP1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t vzip1_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	ZIP1 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
float32x4_t vzip1q_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	ZIP1 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x2_t vzip1q_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	ZIP1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly8x8_t vzip1_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	ZIP1 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
poly8x16_t vzip1q_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	ZIP1 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
poly16x4_t vzip1_p16(poly16x4_t a, poly16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP1 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
poly16x8_t vzip1q_p16(poly16x8_t a, poly16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP1 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int8x8_t vzip2_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	ZIP2 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
int8x16_t vzip2q_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	ZIP2 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
int16x4_t vzip2_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
int16x8_t vzip2q_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int32x2_t vzip2_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	ZIP2 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
int32x4_t vzip2q_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	ZIP2 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
int64x2_t vzip2q_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	ZIP2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint8x8_t vzip2_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	ZIP2 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
uint8x16_t vzip2q_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	ZIP2 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
uint16x4_t vzip2_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
uint16x8_t vzip2q_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
uint32x2_t vzip2_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	ZIP2 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
uint32x4_t vzip2q_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	ZIP2 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
uint64x2_t vzip2q_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	ZIP2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly64x2_t vzip2q_p64(poly64x2_t a, poly64x2_t b)	a -> Vn.2D b -> Vm.2D	ZIP2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t vzip2_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	ZIP2 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
float32x4_t vzip2q_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	ZIP2 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x2_t vzip2q_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	ZIP2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly8x8_t vzip2_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	ZIP2 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
poly8x16_t vzip2q_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	ZIP2 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly16x4_t_vzip2_p16(poly16x4_t a, poly16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
poly16x8_t_vzip2q_p16(poly16x8_t a, poly16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int8x8_t_vuzp1_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	UZP1 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
int8x16_t_vuzp1q_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	UZP1 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
int16x4_t_vuzp1_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP1 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
int16x8_t_vuzp1q_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP1 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int32x2_t_vuzp1_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	UZP1 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
int32x4_t_vuzp1q_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	UZP1 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
int64x2_t_vuzp1q_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	UZP1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint8x8_t_vuzp1_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UZP1 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
uint8x16_t_vuzp1q_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UZP1 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
uint16x4_t_vuzp1_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP1 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
uint16x8_t_vuzp1q_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP1 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
uint32x2_t_vuzp1_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UZP1 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
uint32x4_t_vuzp1q_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UZP1 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
uint64x2_t_vuzp1q_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	UZP1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly64x2_t_vuzp1q_p64(poly64x2_t a, poly64x2_t b)	a -> Vn.2D b -> Vm.2D	UZP1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t_vuzp1_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	UZP1 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
float32x4_t_vuzp1q_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	UZP1 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x2_t_vuzp1q_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	UZP1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly8x8_t_vuzp1_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	UZP1 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
poly8x16_t_vuzp1q_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	UZP1 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
poly16x4_t_vuzp1_p16(poly16x4_t a, poly16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP1 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
poly16x8_t_vuzp1q_p16(poly16x8_t a, poly16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP1 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int8x8_t_vuzp2_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	UZP2 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
int8x16_t_vuzp2q_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	UZP2 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
int16x4_t_vuzp2_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
int16x8_t_vuzp2q_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int32x2_t_vuzp2_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	UZP2 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
int32x4_t_vuzp2q_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	UZP2 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
int64x2_t_vuzp2q_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	UZP2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint8x8_t_vuzp2_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UZP2 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
uint8x16_t_vuzp2q_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UZP2 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
uint16x4_t_vuzp2_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
uint16x8_t_vuzp2q_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
uint32x2_t_vuzp2_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UZP2 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t vuzp2q_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UZP2 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
uint64x2_t vuzp2q_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	UZP2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly64x2_t vuzp2q_p64(poly64x2_t a, poly64x2_t b)	a -> Vn.2D b -> Vm.2D	UZP2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t vuzp2_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	UZP2 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
float32x4_t vuzp2q_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	UZP2 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x2_t vuzp2q_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	UZP2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly8x8_t vuzp2_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	UZP2 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
poly8x16_t vuzp2q_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	UZP2 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
poly16x4_t vuzp2_p16(poly16x4_t a, poly16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
poly16x8_t vuzp2q_p16(poly16x8_t a, poly16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int8x8_t vtrn1_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	TRN1 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
int8x16_t vtrn1q_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	TRN1 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
int16x4_t vtrn1_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN1 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
int16x8_t vtrn1q_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN1 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int32x2_t vtrn1_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	TRN1 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
int32x4_t vtrn1q_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	TRN1 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
int64x2_t vtrn1q_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	TRN1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint8x8_t vtrn1_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	TRN1 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
uint8x16_t vtrn1q_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	TRN1 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
uint16x4_t vtrn1_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN1 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
uint16x8_t vtrn1q_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN1 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
uint32x2_t vtrn1_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	TRN1 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
uint32x4_t vtrn1q_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	TRN1 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
uint64x2_t vtrn1q_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	TRN1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly64x2_t vtrn1q_p64(poly64x2_t a, poly64x2_t b)	a -> Vn.2D b -> Vm.2D	TRN1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t vtrn1_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	TRN1 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
float32x4_t vtrn1q_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	TRN1 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x2_t vtrn1q_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	TRN1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly8x8_t vtrn1_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	TRN1 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
poly8x16_t vtrn1q_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	TRN1 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
poly16x4_t vtrn1_p16(poly16x4_t a, poly16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN1 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
poly16x8_t vtrn1q_p16(poly16x8_t a, poly16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN1 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int8x8_t vtrn2_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	TRN2 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
int8x16_t vtrn2q_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	TRN2 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
int16x4_t vtrn2_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
int16x8_t vtrn2q_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x2_t_vtrn2_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	TRN2 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
int32x4_t_vtrn2q_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	TRN2 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
int64x2_t_vtrn2q_s64(int64x2_t a, int64x2_t b)	a -> Vn.2D b -> Vm.2D	TRN2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint8x8_t_vtrn2_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	TRN2 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
uint8x16_t_vtrn2q_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	TRN2 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
uint16x4_t_vtrn2_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
uint16x8_t_vtrn2q_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
uint32x2_t_vtrn2_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	TRN2 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
uint32x4_t_vtrn2q_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	TRN2 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
uint64x2_t_vtrn2q_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D b -> Vm.2D	TRN2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly64x2_t_vtrn2q_p64(poly64x2_t a, poly64x2_t b)	a -> Vn.2D b -> Vm.2D	TRN2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
float32x2_t_vtrn2_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	TRN2 Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
float32x4_t_vtrn2q_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	TRN2 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float64x2_t_vtrn2q_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	TRN2 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
poly8x8_t_vtrn2_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	TRN2 Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	A64
poly8x16_t_vtrn2q_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	TRN2 Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	A64
poly16x4_t_vtrn2_p16(poly16x4_t a, poly16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
poly16x8_t_vtrn2q_p16(poly16x8_t a, poly16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int8x8_t_vtbl1_s8(int8x8_t a, int8x8_t idx)	Zeros(64):a -> Vn.16B	TBL Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x8_t_vtbl1_u8(uint8x8_t a, uint8x8_t idx)	Zeros(64):a -> Vn.16B	TBL Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
poly8x8_t_vtbl1_p8(poly8x8_t a, uint8x8_t idx)	Zeros(64):a -> Vn.16B	TBL Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
int8x8_t_vtbx1_s8(int8x8_t a, int8x8_t b, int8x8_t idx)	Zeros(64):b -> Vn.16B	MOVI Vtmp.8B,#8 CMHS Vtmp.8B,Vm.8B,Vtmp.8B TBL Vtmp1.8B,{Vn.16B},Vm.8B BIF Vd.8B,Vtmp1.8B,Vtmp.8B	Vd.8B -> result	v7/A32/A64
uint8x8_t_vtbx1_u8(uint8x8_t a, uint8x8_t b, uint8x8_t idx)	Zeros(64):b -> Vn.16B	MOVI Vtmp.8B,#8 CMHS Vtmp.8B,Vm.8B,Vtmp.8B TBL Vtmp1.8B,{Vn.16B},Vm.8B BIF Vd.8B,Vtmp1.8B,Vtmp.8B	Vd.8B -> result	v7/A32/A64
poly8x8_t_vtbx1_p8(poly8x8_t a, poly8x8_t b, uint8x8_t idx)	Zeros(64):b -> Vn.16B	MOVI Vtmp.8B,#8 CMHS Vtmp.8B,Vm.8B,Vtmp.8B TBL Vtmp1.8B,{Vn.16B},Vm.8B BIF Vd.8B,Vtmp1.8B,Vtmp.8B	Vd.8B -> result	v7/A32/A64
int8x8_t_vtbl2_s8(int8x8x2_t a, int8x8_t idx)	a.val[1]:a.val[0] -> Vn.16B	TBL Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x8_t_vtbl2_u8(uint8x8x2_t a, uint8x8_t idx)	a.val[1]:a.val[0] -> Vn.16B	TBL Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
poly8x8_t_vtbl2_p8(poly8x8x2_t a, uint8x8_t idx)	a.val[1]:a.val[0] -> Vn.16B	TBL Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
int8x8_t_vtbl3_s8(int8x8x3_t a, int8x8_t idx)	a.val[1]:a.val[0] -> Vn.16B Zeros(64):a.val[2] -> Vn+1.16B	TBL Vd.8B,{Vn.16B,Vn+1.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x8_t_vtbl3_u8(uint8x8x3_t a, uint8x8_t idx)	a.val[1]:a.val[0] -> Vn.16B Zeros(64):a.val[2] -> Vn+1.16B	TBL Vd.8B,{Vn.16B,Vn+1.16B},Vm.8B	Vd.8B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly8x8_t_vtbl3_p8(poly8x8x3_t a, uint8x8_t idx)	a.val[1]:a.val[0] -> Vn.16B Zeros(64):a.val[2] -> Vn+1.16B	TBL Vd.8B,{Vn.16B,Vn+1.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
int8x8_t_vtbl4_s8(int8x8x4_t a, int8x8_t idx)	a.val[1]:a.val[0] -> Vn.16B a.val[3]:a.val[2] -> Vn+1.16B	TBL Vd.8B,{Vn.16B,Vn+1.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x8_t_vtbl4_u8(uint8x8x4_t a, uint8x8_t idx)	a.val[1]:a.val[0] -> Vn.16B a.val[3]:a.val[2] -> Vn+1.16B	TBL Vd.8B,{Vn.16B,Vn+1.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
poly8x8_t_vtbl4_p8(poly8x8x4_t a, uint8x8_t idx)	a.val[1]:a.val[0] -> Vn.16B a.val[3]:a.val[2] -> Vn+1.16B	TBL Vd.8B,{Vn.16B,Vn+1.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
int8x8_t_vtbx2_s8(int8x8_t a, int8x8x2_t b, int8x8_t idx)	b.val[1]:b.val[0] -> Vn.16B	TBX Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x8_t_vtbx2_u8(uint8x8_t a, uint8x8x2_t b, uint8x8_t idx)	b.val[1]:b.val[0] -> Vn.16B	TBX Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
poly8x8_t_vtbx2_p8(poly8x8_t a, poly8x8x2_t b, uint8x8_t idx)	b.val[1]:b.val[0] -> Vn.16B	TBX Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
int8x8_t_vtbx3_s8(int8x8_t a, int8x8x3_t b, int8x8_t idx)	b.val[1]:b.val[0] -> Vn.16B Zeros(64):b.val[2] -> Vn+1.16B	MOVI Vtmp.8B,#24 CMHS Vtmp.8B,Vm.8B,Vtmp.8B TBL Vtmp.18B,{Vn.16B,Vn+1.16B},Vm.8B BIF Vd.8B,Vtmp.18B,Vtmp.8B	Vd.8B -> result	v7/A32/A64
uint8x8_t_vtbx3_u8(uint8x8_t a, uint8x8x3_t b, uint8x8_t idx)	b.val[1]:b.val[0] -> Vn.16B Zeros(64):b.val[2] -> Vn+1.16B	MOVI Vtmp.8B,#24 CMHS Vtmp.8B,Vm.8B,Vtmp.8B TBL Vtmp.18B,{Vn.16B,Vn+1.16B},Vm.8B BIF Vd.8B,Vtmp.18B,Vtmp.8B	Vd.8B -> result	v7/A32/A64
poly8x8_t_vtbx3_p8(poly8x8_t a, poly8x8x3_t b, uint8x8_t idx)	b.val[1]:b.val[0] -> Vn.16B Zeros(64):b.val[2] -> Vn+1.16B	MOVI Vtmp.8B,#24 CMHS Vtmp.8B,Vm.8B,Vtmp.8B TBL Vtmp.18B,{Vn.16B,Vn+1.16B},Vm.8B BIF Vd.8B,Vtmp.18B,Vtmp.8B	Vd.8B -> result	v7/A32/A64
int8x8_t_vtbx4_s8(int8x8_t a, int8x8x4_t b, int8x8_t idx)	b.val[1]:b.val[0] -> Vn.16B b.val[3]:b.val[2] -> Vn+1.16B	TBX Vd.8B,{Vn.16B,Vn+1.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
uint8x8_t_vtbx4_u8(uint8x8_t a, uint8x8x4_t b, uint8x8_t idx)	b.val[1]:b.val[0] -> Vn.16B b.val[3]:b.val[2] -> Vn+1.16B	TBX Vd.8B,{Vn.16B,Vn+1.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
poly8x8_t_vtbx4_p8(poly8x8_t a, poly8x8x4_t b, uint8x8_t idx)	b.val[1]:b.val[0] -> Vn.16B b.val[3]:b.val[2] -> Vn+1.16B	TBX Vd.8B,{Vn.16B,Vn+1.16B},Vm.8B	Vd.8B -> result	v7/A32/A64
int8x8_t_vqtbl1_s8(int8x16_t t, uint8x8_t idx)	t -> Vn.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	A64
int8x16_t_vqtbl1q_s8(int8x16_t t, uint8x16_t idx)	t -> Vn.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B},Vm.16B	Vd.16B -> result	A64
uint8x8_t_vqtbl1_u8(uint8x16_t t, uint8x8_t idx)	t -> Vn.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t vqtbl1q_u8(uint8x16_t t, uint8x16_t idx)	t -> Vn.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B},Vm.16B	Vd.16B -> result	A64
poly8x8_t vqtbl1_p8(poly8x16_t t, uint8x8_t idx)	t -> Vn.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	A64
poly8x16_t vqtbl1q_p8(poly8x16_t t, uint8x16_t idx)	t -> Vn.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B},Vm.16B	Vd.16B -> result	A64
int8x8_t vqtbx1_s8(int8x8_t a, int8x16_t t, uint8x8_t idx)	a -> Vd.8B t -> Vn.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	A64
int8x16_t vqtbx1q_s8(int8x16_t a, int8x16_t t, uint8x16_t idx)	a -> Vd.16B t -> Vn.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B},Vm.16B	Vd.16B -> result	A64
uint8x8_t vqtbx1_u8(uint8x8_t a, uint8x16_t t, uint8x8_t idx)	a -> Vd.8B t -> Vn.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	A64
uint8x16_t vqtbx1q_u8(uint8x16_t a, uint8x16_t t, uint8x16_t idx)	a -> Vd.16B t -> Vn.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B},Vm.16B	Vd.16B -> result	A64
poly8x8_t vqtbx1_p8(poly8x8_t a, poly8x16_t t, uint8x8_t idx)	a -> Vd.8B t -> Vn.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B},Vm.8B	Vd.8B -> result	A64
poly8x16_t vqtbx1q_p8(poly8x16_t a, poly8x16_t t, uint8x16_t idx)	a -> Vd.16B t -> Vn.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B},Vm.16B	Vd.16B -> result	A64
int8x8_t vqtbl2_s8(int8x16x2_t t, uint8x8_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B - Vn+1.16B},Vm.8B	Vd.8B -> result	A64
int8x16_t vqtbl2q_s8(int8x16x2_t t, uint8x16_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B - Vn+1.16B},Vm.16B	Vd.16B -> result	A64
uint8x8_t vqtbl2_u8(uint8x16x2_t t, uint8x8_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B - Vn+1.16B},Vm.8B	Vd.8B -> result	A64
uint8x16_t vqtbl2q_u8(uint8x16x2_t t, uint8x16_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B - Vn+1.16B},Vm.16B	Vd.16B -> result	A64
poly8x8_t vqtbl2_p8(poly8x16x2_t t, uint8x8_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B - Vn+1.16B},Vm.8B	Vd.8B -> result	A64
poly8x16_t vqtbl2q_p8(poly8x16x2_t t, uint8x16_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B - Vn+1.16B},Vm.16B	Vd.16B -> result	A64
int8x8_t vqtbl3_s8(int8x16x3_t t, uint8x8_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B - Vn+2.16B},Vm.8B	Vd.8B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t vqtbl3q_s8(int8x16x3_t t, uint8x16_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B - Vn+2.16B},Vm.16B	Vd.16B -> result	A64
uint8x8_t vqtbl3_u8(uint8x16x3_t t, uint8x8_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B - Vn+2.16B},Vm.8B	Vd.8B -> result	A64
uint8x16_t vqtbl3q_u8(uint8x16x3_t t, uint8x16_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B - Vn+2.16B},Vm.16B	Vd.16B -> result	A64
poly8x8_t vqtbl3_p8(poly8x16x3_t t, uint8x8_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B - Vn+2.16B},Vm.8B	Vd.8B -> result	A64
poly8x16_t vqtbl3q_p8(poly8x16x3_t t, uint8x16_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B - Vn+2.16B},Vm.16B	Vd.16B -> result	A64
int8x8_t vqtbl4_s8(int8x16x4_t t, uint8x8_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B - Vn+3.16B},Vm.8B	Vd.8B -> result	A64
int8x16_t vqtbl4q_s8(int8x16x4_t t, uint8x16_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B - Vn+3.16B},Vm.16B	Vd.16B -> result	A64
uint8x8_t vqtbl4_u8(uint8x16x4_t t, uint8x8_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B - Vn+3.16B},Vm.8B	Vd.8B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t vqtbl4q_u8(uint8x16x4_t t, uint8x16_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B - Vn+3.16B},Vm.16B	Vd.16B -> result	A64
poly8x8_t vqtbl4_p8(poly8x16x4_t t, uint8x8_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.8B	TBL Vd.8B,{Vn.16B - Vn+3.16B},Vm.8B	Vd.8B -> result	A64
poly8x16_t vqtbl4q_p8(poly8x16x4_t t, uint8x16_t idx)	t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.16B	TBL Vd.16B,{Vn.16B - Vn+3.16B},Vm.16B	Vd.16B -> result	A64
int8x8_t vqtbx2_s8(int8x8_t a, int8x16x2_t t, uint8x8_t idx)	a -> Vd.8B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B - Vn+1.16B},Vm.8B	Vd.8B -> result	A64
int8x16_t vqtbx2q_s8(int8x16_t a, int8x16x2_t t, uint8x16_t idx)	a -> Vd.16B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B - Vn+1.16B},Vm.16B	Vd.16B -> result	A64
uint8x8_t vqtbx2_u8(uint8x8_t a, uint8x16x2_t t, uint8x8_t idx)	a -> Vd.8B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B - Vn+1.16B},Vm.8B	Vd.8B -> result	A64
uint8x16_t vqtbx2q_u8(uint8x16_t a, uint8x16x2_t t, uint8x16_t idx)	a -> Vd.16B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B - Vn+1.16B},Vm.16B	Vd.16B -> result	A64
poly8x8_t vqtbx2_p8(poly8x8_t a, poly8x16x2_t t, uint8x8_t idx)	a -> Vd.8B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B - Vn+1.16B},Vm.8B	Vd.8B -> result	A64
poly8x16_t vqtbx2q_p8(poly8x16_t a, poly8x16x2_t t, uint8x16_t idx)	a -> Vd.16B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B - Vn+1.16B},Vm.16B	Vd.16B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x8_t vqtbx3_s8(int8x8_t a, int8x16x3_t t, uint8x8_t idx)	a -> Vd.8B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B - Vn+2.16B},Vm.8B	Vd.8B -> result	A64
int8x16_t vqtbx3q_s8(int8x16_t a, int8x16x3_t t, uint8x16_t idx)	a -> Vd.16B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B - Vn+2.16B},Vm.16B	Vd.16B -> result	A64
uint8x8_t vqtbx3_u8(uint8x8_t a, uint8x16x3_t t, uint8x8_t idx)	a -> Vd.8B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B - Vn+2.16B},Vm.8B	Vd.8B -> result	A64
uint8x16_t vqtbx3q_u8(uint8x16_t a, uint8x16x3_t t, uint8x16_t idx)	a -> Vd.16B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B - Vn+2.16B},Vm.16B	Vd.16B -> result	A64
poly8x8_t vqtbx3_p8(poly8x8_t a, poly8x16x3_t t, uint8x8_t idx)	a -> Vd.8B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B - Vn+2.16B},Vm.8B	Vd.8B -> result	A64
poly8x16_t vqtbx3q_p8(poly8x16_t a, poly8x16x3_t t, uint8x16_t idx)	a -> Vd.16B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B - Vn+2.16B},Vm.16B	Vd.16B -> result	A64
int8x8_t vqtbx4_s8(int8x8_t a, int8x16x4_t t, uint8x8_t idx)	a -> Vd.8B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B - Vn+3.16B},Vm.8B	Vd.8B -> result	A64
int8x16_t vqtbx4q_s8(int8x16_t a, int8x16x4_t t, uint8x16_t idx)	a -> Vd.16B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B - Vn+3.16B},Vm.16B	Vd.16B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x8_t vqtbx4_u8(uint8x8_t a, uint8x16x4_t t, uint8x8_t idx)	a -> Vd.8B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B - Vn+3.16B},Vm.8B	Vd.8B -> result	A64
uint8x16_t vqtbx4q_u8(uint8x16_t a, uint8x16x4_t t, uint8x16_t idx)	a -> Vd.16B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B - Vn+3.16B},Vm.16B	Vd.16B -> result	A64
poly8x8_t vqtbx4_p8(poly8x8_t a, poly8x16x4_t t, uint8x8_t idx)	a -> Vd.8B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.8B	TBX Vd.8B,{Vn.16B - Vn+3.16B},Vm.8B	Vd.8B -> result	A64
poly8x16_t vqtbx4q_p8(poly8x16_t a, poly8x16x4_t t, uint8x16_t idx)	a -> Vd.16B t.val[0] -> Vn.16B t.val[1] -> Vn+1.16B t.val[2] -> Vn+2.16B t.val[3] -> Vn+3.16B idx -> Vm.16B	TBX Vd.16B,{Vn.16B - Vn+3.16B},Vm.16B	Vd.16B -> result	A64
uint8_t vget_lane_u8(uint8x8_t v, const int lane)	v -> Vn.8B 0 <= lane <= 7	UMOV Rd,Vn.B[lane]	Rd -> result	v7/A32/A64
uint16_t vget_lane_u16(uint16x4_t v, const int lane)	v -> Vn.4H 0 <= lane <= 3	UMOV Rd,Vn.H[lane]	Rd -> result	v7/A32/A64
uint32_t vget_lane_u32(uint32x2_t v, const int lane)	v -> Vn.2S 0 <= lane <= 1	UMOV Rd,Vn.S[lane]	Rd -> result	v7/A32/A64
uint64_t vget_lane_u64(uint64x1_t v, const int lane)	v -> Vn.1D lane == 0	UMOV Rd,Vn.D[lane]	Rd -> result	v7/A32/A64
poly64_t vget_lane_p64(poly64x1_t v, const int lane)	v -> Vn.1D lane == 0	UMOV Rd,Vn.D[lane]	Rd -> result	A32/A64
int8_t vget_lane_s8(int8x8_t v, const int lane)	v -> Vn.8B 0 <= lane <= 7	SMOV Rd,Vn.B[lane]	Rd -> result	v7/A32/A64
int16_t vget_lane_s16(int16x4_t v, const int lane)	v -> Vn.4H 0 <= lane <= 3	SMOV Rd,Vn.H[lane]	Rd -> result	v7/A32/A64
int32_t vget_lane_s32(int32x2_t v, const int lane)	v -> Vn.2S 0 <= lane <= 1	SMOV Rd,Vn.S[lane]	Rd -> result	v7/A32/A64
int64_t vget_lane_s64(int64x1_t v, const int lane)	v -> Vn.1D lane == 0	UMOV Rd,Vn.D[lane]	Rd -> result	v7/A32/A64
poly8_t vget_lane_p8(poly8x8_t v, const int lane)	v -> Vn.8B 0 <= lane <= 7	UMOV Rd,Vn.B[lane]	Rd -> result	v7/A32/A64
poly16_t vget_lane_p16(poly16x4_t v, const int lane)	v -> Vn.4H 0 <= lane <= 3	UMOV Rd,Vn.H[lane]	Rd -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32_t vget_lane_f32(float32x2_t v, const int lane)	v -> Vn.2S 0 <= lane <= 1	DUP Sd,Vn.S[lane]	Sd -> result	v7/A32/A64
float64_t vget_lane_f64(float64x1_t v, const int lane)	v -> Vn.1D lane == 0	DUP Dd,Vn.D[lane]	Dd -> result	A64
uint8_t vgetq_lane_u8(uint8x16_t v, const int lane)	v -> Vn.16B 0 <= lane <= 15	UMOV Rd,Vn.B[lane]	Rd -> result	v7/A32/A64
uint16_t vgetq_lane_u16(uint16x8_t v, const int lane)	v -> Vn.8H 0 <= lane <= 7	UMOV Rd,Vn.H[lane]	Rd -> result	v7/A32/A64
uint32_t vgetq_lane_u32(uint32x4_t v, const int lane)	v -> Vn.4S 0 <= lane <= 3	UMOV Rd,Vn.S[lane]	Rd -> result	v7/A32/A64
uint64_t vgetq_lane_u64(uint64x2_t v, const int lane)	v -> Vn.2D 0 <= lane <= 1	UMOV Rd,Vn.D[lane]	Rd -> result	v7/A32/A64
poly64_t vgetq_lane_p64(poly64x2_t v, const int lane)	v -> Vn.2D 0 <= lane <= 1	UMOV Rd,Vn.D[lane]	Rd -> result	A32/A64
int8_t vgetq_lane_s8(int8x16_t v, const int lane)	v -> Vn.16B 0 <= lane <= 15	SMOV Rd,Vn.B[lane]	Rd -> result	v7/A32/A64
int16_t vgetq_lane_s16(int16x8_t v, const int lane)	v -> Vn.8H 0 <= lane <= 7	SMOV Rd,Vn.H[lane]	Rd -> result	v7/A32/A64
int32_t vgetq_lane_s32(int32x4_t v, const int lane)	v -> Vn.4S 0 <= lane <= 3	SMOV Rd,Vn.S[lane]	Rd -> result	v7/A32/A64
int64_t vgetq_lane_s64(int64x2_t v, const int lane)	v -> Vn.2D 0 <= lane <= 1	UMOV Rd,Vn.D[lane]	Rd -> result	v7/A32/A64
poly8_t vgetq_lane_p8(poly8x16_t v, const int lane)	v -> Vn.16B 0 <= lane <= 15	UMOV Rd,Vn.B[lane]	Rd -> result	v7/A32/A64
poly16_t vgetq_lane_p16(poly16x8_t v, const int lane)	v -> Vn.8H 0 <= lane <= 7	UMOV Rd,Vn.H[lane]	Rd -> result	v7/A32/A64
float16_t vget_lane_f16(float16x4_t v, const int lane)	v -> Vn.4H 0 <= lane <= 3	DUP Hd,Vn.H[lane]	Hd -> result	v7/A32/A64
float16_t vgetq_lane_f16(float16x8_t v, const int lane)	v -> Vn.8H 0 <= lane <= 7	DUP Hd,Vn.H[lane]	Hd -> result	v7/A32/A64
float32_t vgetq_lane_f32(float32x4_t v, const int lane)	v -> Vn.4S 0 <= lane <= 3	DUP Sd,Vn.S[lane]	Sd -> result	v7/A32/A64
float64_t vgetq_lane_f64(float64x2_t v, const int lane)	v -> Vn.2D 0 <= lane <= 1	DUP Dd,Vn.D[lane]	Dd -> result	A64
uint8x8_t vset_lane_u8(uint8_t a, uint8x8_t v, const int lane)	a -> Rn v -> Vd.8B 0 <= lane <= 7	INS Vd.B[lane],Rn	Vd.8B -> result	v7/A32/A64
uint16x4_t vset_lane_u16(uint16_t a, uint16x4_t v, const int lane)	a -> Rn v -> Vd.4H 0 <= lane <= 3	INS Vd.H[lane],Rn	Vd.4H -> result	v7/A32/A64
uint32x2_t vset_lane_u32(uint32_t a, uint32x2_t v, const int lane)	a -> Rn v -> Vd.2S 0 <= lane <= 1	INS Vd.S[lane],Rn	Vd.2S -> result	v7/A32/A64
uint64x1_t vset_lane_u64(uint64_t a, uint64x1_t v, const int lane)	a -> Rn v -> Vd.1D lane == 0	INS Vd.D[lane],Rn	Vd.1D -> result	v7/A32/A64
poly64x1_t vset_lane_p64(poly64_t a, poly64x1_t v, const int lane)	a -> Rn v -> Vd.1D lane == 0	INS Vd.D[lane],Rn	Vd.1D -> result	A32/A64
int8x8_t vset_lane_s8(int8_t a, int8x8_t v, const int lane)	a -> Rn v -> Vd.8B 0 <= lane <= 7	INS Vd.B[lane],Rn	Vd.8B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x4_t vset_lane_s16(int16_t a, int16x4_t v, const int lane)	a -> Rn v -> Vd.4H 0 <= lane <= 3	INS Vd.H[lane],Rn	Vd.4H -> result	v7/A32/A64
int32x2_t vset_lane_s32(int32_t a, int32x2_t v, const int lane)	a -> Rn v -> Vd.2S 0 <= lane <= 1	INS Vd.S[lane],Rn	Vd.2S -> result	v7/A32/A64
int64x1_t vset_lane_s64(int64_t a, int64x1_t v, const int lane)	a -> Rn v -> Vd.1D lane == 0	INS Vd.D[lane],Rn	Vd.1D -> result	v7/A32/A64
poly8x8_t vset_lane_p8(poly8_t a, poly8x8_t v, const int lane)	a -> Rn v -> Vd.8B 0 <= lane <= 7	INS Vd.B[lane],Rn	Vd.8B -> result	v7/A32/A64
poly16x4_t vset_lane_p16(poly16_t a, poly16x4_t v, const int lane)	a -> Rn v -> Vd.4H 0 <= lane <= 3	INS Vd.H[lane],Rn	Vd.4H -> result	v7/A32/A64
float16x4_t vset_lane_f16(float16_t a, float16x4_t v, const int lane)	a -> VnH v -> Vd.4H 0 <= lane <= 3	INS Vd.H[lane],Vn.H[0]	Vd.4H -> result	v7/A32/A64
float16x8_t vsetq_lane_f16(float16_t a, float16x8_t v, const int lane)	a -> VnH v -> Vd.8H 0 <= lane <= 7	INS Vd.H[lane],Vn.H[0]	Vd.8H -> result	v7/A32/A64
float32x2_t vset_lane_f32(float32_t a, float32x2_t v, const int lane)	a -> Rn v -> Vd.2S 0 <= lane <= 1	INS Vd.S[lane],Rn	Vd.2S -> result	v7/A32/A64
float64x1_t vset_lane_f64(float64_t a, float64x1_t v, const int lane)	a -> Rn v -> Vd.1D lane == 0	INS Vd.D[lane],Rn	Vd.1D -> result	A64
uint8x16_t vsetq_lane_u8(uint8_t a, uint8x16_t v, const int lane)	a -> Rn v -> Vd.16B 0 <= lane <= 15	INS Vd.B[lane],Rn	Vd.16B -> result	v7/A32/A64
uint16x8_t vsetq_lane_u16(uint16_t a, uint16x8_t v, const int lane)	a -> Rn v -> Vd.8H 0 <= lane <= 7	INS Vd.H[lane],Rn	Vd.8H -> result	v7/A32/A64
uint32x4_t vsetq_lane_u32(uint32_t a, uint32x4_t v, const int lane)	a -> Rn v -> Vd.4S 0 <= lane <= 3	INS Vd.S[lane],Rn	Vd.4S -> result	v7/A32/A64
uint64x2_t vsetq_lane_u64(uint64_t a, uint64x2_t v, const int lane)	a -> Rn v -> Vd.2D 0 <= lane <= 1	INS Vd.D[lane],Rn	Vd.2D -> result	v7/A32/A64
poly64x2_t vsetq_lane_p64(poly64_t a, poly64x2_t v, const int lane)	a -> Rn v -> Vd.2D 0 <= lane <= 1	INS Vd.D[lane],Rn	Vd.2D -> result	A32/A64
int8x16_t vsetq_lane_s8(int8_t a, int8x16_t v, const int lane)	a -> Rn v -> Vd.16B 0 <= lane <= 15	INS Vd.B[lane],Rn	Vd.16B -> result	v7/A32/A64
int16x8_t vsetq_lane_s16(int16_t a, int16x8_t v, const int lane)	a -> Rn v -> Vd.8H 0 <= lane <= 7	INS Vd.H[lane],Rn	Vd.8H -> result	v7/A32/A64
int32x4_t vsetq_lane_s32(int32_t a, int32x4_t v, const int lane)	a -> Rn v -> Vd.4S 0 <= lane <= 3	INS Vd.S[lane],Rn	Vd.4S -> result	v7/A32/A64
int64x2_t vsetq_lane_s64(int64_t a, int64x2_t v, const int lane)	a -> Rn v -> Vd.2D 0 <= lane <= 1	INS Vd.D[lane],Rn	Vd.2D -> result	v7/A32/A64
poly8x16_t vsetq_lane_p8(poly8_t a, poly8x16_t v, const int lane)	a -> Rn v -> Vd.16B 0 <= lane <= 15	INS Vd.B[lane],Rn	Vd.16B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly16x8_t vsetq_lane_p16(poly16_t a, poly16x8_t v, const int lane)	a -> Rn v -> Vd.8H 0 <= lane <= 7	INS Vd.H[lane],Rn	Vd.8H -> result	v7/A32/A64
float32x4_t vsetq_lane_f32(float32_t a, float32x4_t v, const int lane)	a -> Rn v -> Vd.4S 0 <= lane <= 3	INS Vd.S[lane],Rn	Vd.4S -> result	v7/A32/A64
float64x2_t vsetq_lane_f64(float64_t a, float64x2_t v, const int lane)	a -> Rn v -> Vd.2D 0 <= lane <= 1	INS Vd.D[lane],Rn	Vd.2D -> result	A64
float32_t vrecpxs_f32(float32_t a)	a -> Sn	FRECPX Sd,Sn	Sd -> result	A64
float64_t vrecpxd_f64(float64_t a)	a -> Dn	FRECPX Dd,Dn	Dd -> result	A64
float32x2_t vfma_n_f32(float32x2_t a, float32x2_t b, float32_t n)	a -> Vd.2S b -> Vn.2S n -> Vm.S[0]	FMLA Vd.2S,Vn.2S,Vm.S[0]	Vd.2S -> result	v7/A32/A64
float32x4_t vfmaq_n_f32(float32x4_t a, float32x4_t b, float32_t n)	a -> Vd.4S b -> Vn.4S n -> Vm.S[0]	FMLA Vd.4S,Vn.4S,Vm.S[0]	Vd.4S -> result	v7/A32/A64
float32x2_t vfms_n_f32(float32x2_t a, float32x2_t b, float32_t n)	a -> Vd.2S b -> Vn.2S n -> Vm.S[0]	FMLS Vd.2S,Vn.2S,Vm.S[0]	Vd.2S -> result	A64
float32x4_t vfmsq_n_f32(float32x4_t a, float32x4_t b, float32_t n)	a -> Vd.4S b -> Vn.4S n -> Vm.S[0]	FMLS Vd.4S,Vn.4S,Vm.S[0]	Vd.4S -> result	A64
float64x1_t vfma_n_f64(float64x1_t a, float64x1_t b, float64_t n)	a -> Da b -> Dn n -> Dm	FMADD Dd,Dn,Dm,Da	Dd -> result	A64
float64x2_t vfmaq_n_f64(float64x2_t a, float64x2_t b, float64_t n)	a -> Vd.2D b -> Vn.2D n -> Vm.D[0]	FMLA Vd.2D,Vn.2D,Vm.D[0]	Vd.2D -> result	A64
float64x1_t vfms_n_f64(float64x1_t a, float64x1_t b, float64_t n)	a -> Da b -> Dn n -> Dm	FMSUB Dd,Dn,Dm,Da	Dd -> result	A64
float64x2_t vfmsq_n_f64(float64x2_t a, float64x2_t b, float64_t n)	a -> Vd.2D b -> Vn.2D n -> Vm.D[0]	FMLS Vd.2D,Vn.2D,Vm.D[0]	Vd.2D -> result	A64
int8x8x2_t vtrn_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	TRN1 Vd1.8B,Vn.8B,Vm.8B TRN2 Vd2.8B,Vn.8B,Vm.8B	Vd1.8B -> result.val[0] Vd2.8B -> result.val[1]	v7/A32/A64
int16x4x2_t vtrn_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN1 Vd1.4H,Vn.4H,Vm.4H TRN2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64
uint8x8x2_t vtrn_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	TRN1 Vd1.8B,Vn.8B,Vm.8B TRN2 Vd2.8B,Vn.8B,Vm.8B	Vd1.8B -> result.val[0] Vd2.8B -> result.val[1]	v7/A32/A64
uint16x4x2_t vtrn_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN1 Vd1.4H,Vn.4H,Vm.4H TRN2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64
poly8x8x2_t vtrn_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	TRN1 Vd1.8B,Vn.8B,Vm.8B TRN2 Vd2.8B,Vn.8B,Vm.8B	Vd1.8B -> result.val[0] Vd2.8B -> result.val[1]	v7/A32/A64
poly16x4x2_t vtrn_p16(poly16x4_t a, poly16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN1 Vd1.4H,Vn.4H,Vm.4H TRN2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64
int32x2x2_t vtrn_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	TRN1 Vd1.2S,Vn.2S,Vm.2S TRN2 Vd2.2S,Vn.2S,Vm.2S	Vd1.2S -> result.val[0] Vd2.2S -> result.val[1]	v7/A32/A64
float32x2x2_t vtrn_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	TRN1 Vd1.2S,Vn.2S,Vm.2S TRN2 Vd2.2S,Vn.2S,Vm.2S	Vd1.2S -> result.val[0] Vd2.2S -> result.val[1]	v7/A32/A64
uint32x2x2_t vtrn_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	TRN1 Vd1.2S,Vn.2S,Vm.2S TRN2 Vd2.2S,Vn.2S,Vm.2S	Vd1.2S -> result.val[0] Vd2.2S -> result.val[1]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16x2_t_vtrnq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	TRN1 Vd1.16B,Vn.16B,Vm.16B TRN2 Vd2.16B,Vn.16B,Vm.16B	Vd1.16B -> result.val[0] Vd2.16B -> result.val[1]	v7/A32/A64
int16x8x2_t_vtrnq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN1 Vd1.8H,Vn.8H,Vm.8H TRN2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
int32x4x2_t_vtrnq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	TRN1 Vd1.4S,Vn.4S,Vm.4S TRN2 Vd2.4S,Vn.4S,Vm.4S	Vd1.4S -> result.val[0] Vd2.4S -> result.val[1]	v7/A32/A64
float32x4x2_t_vtrnq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	TRN1 Vd1.4S,Vn.4S,Vm.4S TRN2 Vd2.4S,Vn.4S,Vm.4S	Vd1.4S -> result.val[0] Vd2.4S -> result.val[1]	v7/A32/A64
uint8x16x2_t_vtrnq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	TRN1 Vd1.16B,Vn.16B,Vm.16B TRN2 Vd2.16B,Vn.16B,Vm.16B	Vd1.16B -> result.val[0] Vd2.16B -> result.val[1]	v7/A32/A64
uint16x8x2_t_vtrnq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN1 Vd1.8H,Vn.8H,Vm.8H TRN2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
uint32x4x2_t_vtrnq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	TRN1 Vd1.4S,Vn.4S,Vm.4S TRN2 Vd2.4S,Vn.4S,Vm.4S	Vd1.4S -> result.val[0] Vd2.4S -> result.val[1]	v7/A32/A64
poly8x16x2_t_vtrnq_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	TRN1 Vd1.16B,Vn.16B,Vm.16B TRN2 Vd2.16B,Vn.16B,Vm.16B	Vd1.16B -> result.val[0] Vd2.16B -> result.val[1]	v7/A32/A64
poly16x8x2_t_vtrnq_p16(poly16x8_t a, poly16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN1 Vd1.8H,Vn.8H,Vm.8H TRN2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
int8x8x2_t_vzip_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	ZIP1 Vd1.8B,Vn.8B,Vm.8B ZIP2 Vd2.8B,Vn.8B,Vm.8B	Vd1.8B -> result.val[0] Vd2.8B -> result.val[1]	v7/A32/A64
int16x4x2_t_vzip_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP1 Vd1.4H,Vn.4H,Vm.4H ZIP2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64
uint8x8x2_t_vzip_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	ZIP1 Vd1.8B,Vn.8B,Vm.8B ZIP2 Vd2.8B,Vn.8B,Vm.8B	Vd1.8B -> result.val[0] Vd2.8B -> result.val[1]	v7/A32/A64
uint16x4x2_t_vzip_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP1 Vd1.4H,Vn.4H,Vm.4H ZIP2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64
poly8x8x2_t_vzip_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	ZIP1 Vd1.8B,Vn.8B,Vm.8B ZIP2 Vd2.8B,Vn.8B,Vm.8B	Vd1.8B -> result.val[0] Vd2.8B -> result.val[1]	v7/A32/A64
poly16x4x2_t_vzip_p16(poly16x4_t a, poly16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP1 Vd1.4H,Vn.4H,Vm.4H ZIP2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64
int32x2x2_t_vzip_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	ZIP1 Vd1.2S,Vn.2S,Vm.2S ZIP2 Vd2.2S,Vn.2S,Vm.2S	Vd1.2S -> result.val[0] Vd2.2S -> result.val[1]	v7/A32/A64
float32x2x2_t_vzip_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	ZIP1 Vd1.2S,Vn.2S,Vm.2S ZIP2 Vd2.2S,Vn.2S,Vm.2S	Vd1.2S -> result.val[0] Vd2.2S -> result.val[1]	v7/A32/A64
uint32x2x2_t_vzip_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	ZIP1 Vd1.2S,Vn.2S,Vm.2S ZIP2 Vd2.2S,Vn.2S,Vm.2S	Vd1.2S -> result.val[0] Vd2.2S -> result.val[1]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16x2_t_vzipq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	ZIP1 Vd1.16B,Vn.16B,Vm.16B ZIP2 Vd2.16B,Vn.16B,Vm.16B	Vd1.16B -> result.val[0] Vd2.16B -> result.val[1]	v7/A32/A64
int16x8x2_t_vzipq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP1 Vd1.8H,Vn.8H,Vm.8H ZIP2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
int32x4x2_t_vzipq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	ZIP1 Vd1.4S,Vn.4S,Vm.4S ZIP2 Vd2.4S,Vn.4S,Vm.4S	Vd1.4S -> result.val[0] Vd2.4S -> result.val[1]	v7/A32/A64
float32x4x2_t_vzipq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	ZIP1 Vd1.4S,Vn.4S,Vm.4S ZIP2 Vd2.4S,Vn.4S,Vm.4S	Vd1.4S -> result.val[0] Vd2.4S -> result.val[1]	v7/A32/A64
uint8x16x2_t_vzipq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	ZIP1 Vd1.16B,Vn.16B,Vm.16B ZIP2 Vd2.16B,Vn.16B,Vm.16B	Vd1.16B -> result.val[0] Vd2.16B -> result.val[1]	v7/A32/A64
uint16x8x2_t_vzipq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP1 Vd1.8H,Vn.8H,Vm.8H ZIP2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
uint32x4x2_t_vzipq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	ZIP1 Vd1.4S,Vn.4S,Vm.4S ZIP2 Vd2.4S,Vn.4S,Vm.4S	Vd1.4S -> result.val[0] Vd2.4S -> result.val[1]	v7/A32/A64
poly8x16x2_t_vzipq_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	ZIP1 Vd1.16B,Vn.16B,Vm.16B ZIP2 Vd2.16B,Vn.16B,Vm.16B	Vd1.16B -> result.val[0] Vd2.16B -> result.val[1]	v7/A32/A64
poly16x8x2_t_vzipq_p16(poly16x8_t a, poly16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP1 Vd1.8H,Vn.8H,Vm.8H ZIP2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
int8x8x2_t_vuzp_s8(int8x8_t a, int8x8_t b)	a -> Vn.8B b -> Vm.8B	UZP1 Vd1.8B,Vn.8B,Vm.8B UZP2 Vd2.8B,Vn.8B,Vm.8B	Vd1.8B -> result.val[0] Vd2.8B -> result.val[1]	v7/A32/A64
int16x4x2_t_vuzp_s16(int16x4_t a, int16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP1 Vd1.4H,Vn.4H,Vm.4H UZP2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64
int32x2x2_t_vuzp_s32(int32x2_t a, int32x2_t b)	a -> Vn.2S b -> Vm.2S	UZP1 Vd1.2S,Vn.2S,Vm.2S UZP2 Vd2.2S,Vn.2S,Vm.2S	Vd1.2S -> result.val[0] Vd2.2S -> result.val[1]	v7/A32/A64
float32x2x2_t_vuzp_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	UZP1 Vd1.2S,Vn.2S,Vm.2S UZP2 Vd2.2S,Vn.2S,Vm.2S	Vd1.2S -> result.val[0] Vd2.2S -> result.val[1]	v7/A32/A64
uint8x8x2_t_vuzp_u8(uint8x8_t a, uint8x8_t b)	a -> Vn.8B b -> Vm.8B	UZP1 Vd1.8B,Vn.8B,Vm.8B UZP2 Vd2.8B,Vn.8B,Vm.8B	Vd1.8B -> result.val[0] Vd2.8B -> result.val[1]	v7/A32/A64
uint16x4x2_t_vuzp_u16(uint16x4_t a, uint16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP1 Vd1.4H,Vn.4H,Vm.4H UZP2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64
uint32x2x2_t_vuzp_u32(uint32x2_t a, uint32x2_t b)	a -> Vn.2S b -> Vm.2S	UZP1 Vd1.2S,Vn.2S,Vm.2S UZP2 Vd2.2S,Vn.2S,Vm.2S	Vd1.2S -> result.val[0] Vd2.2S -> result.val[1]	v7/A32/A64
poly8x8x2_t_vuzp_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	UZP1 Vd1.8B,Vn.8B,Vm.8B UZP2 Vd2.8B,Vn.8B,Vm.8B	Vd1.8B -> result.val[0] Vd2.8B -> result.val[1]	v7/A32/A64
poly16x4x2_t_vuzp_p16(poly16x4_t a, poly16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP1 Vd1.4H,Vn.4H,Vm.4H UZP2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16x2_t_vuzpq_s8(int8x16_t a, int8x16_t b)	a -> Vn.16B b -> Vm.16B	UZP1 Vd1.16B,Vn.16B,Vm.16B UZP2 Vd2.16B,Vn.16B,Vm.16B	Vd1.16B -> result.val[0] Vd2.16B -> result.val[1]	v7/A32/A64
int16x8x2_t_vuzpq_s16(int16x8_t a, int16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP1 Vd1.8H,Vn.8H,Vm.8H UZP2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
int32x4x2_t_vuzpq_s32(int32x4_t a, int32x4_t b)	a -> Vn.4S b -> Vm.4S	UZP1 Vd1.4S,Vn.4S,Vm.4S UZP2 Vd2.4S,Vn.4S,Vm.4S	Vd1.4S -> result.val[0] Vd2.4S -> result.val[1]	v7/A32/A64
float32x4x2_t_vuzpq_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	UZP1 Vd1.4S,Vn.4S,Vm.4S UZP2 Vd2.4S,Vn.4S,Vm.4S	Vd1.4S -> result.val[0] Vd2.4S -> result.val[1]	v7/A32/A64
uint8x16x2_t_vuzpq_u8(uint8x16_t a, uint8x16_t b)	a -> Vn.16B b -> Vm.16B	UZP1 Vd1.16B,Vn.16B,Vm.16B UZP2 Vd2.16B,Vn.16B,Vm.16B	Vd1.16B -> result.val[0] Vd2.16B -> result.val[1]	v7/A32/A64
uint16x8x2_t_vuzpq_u16(uint16x8_t a, uint16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP1 Vd1.8H,Vn.8H,Vm.8H UZP2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
uint32x4x2_t_vuzpq_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S b -> Vm.4S	UZP1 Vd1.4S,Vn.4S,Vm.4S UZP2 Vd2.4S,Vn.4S,Vm.4S	Vd1.4S -> result.val[0] Vd2.4S -> result.val[1]	v7/A32/A64
poly8x16x2_t_vuzpq_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	UZP1 Vd1.16B,Vn.16B,Vm.16B UZP2 Vd2.16B,Vn.16B,Vm.16B	Vd1.16B -> result.val[0] Vd2.16B -> result.val[1]	v7/A32/A64
poly16x8x2_t_vuzpq_p16(poly16x8_t a, poly16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP1 Vd1.8H,Vn.8H,Vm.8H UZP2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
int16x4_t_vreinterpret_s16_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
int32x2_t_vreinterpret_s32_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.2S -> result	v7/A32/A64
float32x2_t_vreinterpret_f32_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.2S -> result	v7/A32/A64
uint8x8_t_vreinterpret_u8_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.8B -> result	v7/A32/A64
uint16x4_t_vreinterpret_u16_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
uint32x2_t_vreinterpret_u32_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.2S -> result	v7/A32/A64
poly8x8_t_vreinterpret_p8_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.8B -> result	v7/A32/A64
poly16x4_t_vreinterpret_p16_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
uint64x1_t_vreinterpret_u64_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.1D -> result	v7/A32/A64
int64x1_t_vreinterpret_s64_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.1D -> result	v7/A32/A64
float64x1_t_vreinterpret_f64_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.1D -> result	A64
poly64x1_t_vreinterpret_p64_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.1D -> result	A32/A64
float16x4_t_vreinterpret_f16_s8(int8x8_t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
int8x8_t_vreinterpret_s8_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
int32x2_t_vreinterpret_s32_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
float32x2_t_vreinterpret_f32_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
uint8x8_t_vreinterpret_u8_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
uint16x4_t_vreinterpret_u16_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64
uint32x2_t_vreinterpret_u32_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
poly8x8_t_vreinterpret_p8_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
poly16x4_t_vreinterpret_p16_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64
uint64x1_t_vreinterpret_u64_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	v7/A32/A64
int64x1_t_vreinterpret_s64_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	v7/A32/A64
float64x1_t_vreinterpret_f64_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	A64
poly64x1_t_vreinterpret_p64_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	A32/A64
float16x4_t_vreinterpret_f16_s16(int16x4_t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64
int8x8_t_vreinterpret_s8_s32(int32x2_t a)	a -> Vd.2S	NOP	Vd.8B -> result	v7/A32/A64
int16x4_t_vreinterpret_s16_s32(int32x2_t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
float32x2_t_vreinterpret_f32_s32(int32x2_t a)	a -> Vd.2S	NOP	Vd.2S -> result	v7/A32/A64
uint8x8_t_vreinterpret_u8_s32(int32x2_t a)	a -> Vd.2S	NOP	Vd.8B -> result	v7/A32/A64
uint16x4_t_vreinterpret_u16_s32(int32x2_t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
uint32x2_t_vreinterpret_u32_s32(int32x2_t a)	a -> Vd.2S	NOP	Vd.2S -> result	v7/A32/A64
poly8x8_t_vreinterpret_p8_s32(int32x2_t a)	a -> Vd.2S	NOP	Vd.8B -> result	v7/A32/A64
poly16x4_t_vreinterpret_p16_s32(int32x2_t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
uint64x1_t_vreinterpret_u64_s32(int32x2_t a)	a -> Vd.2S	NOP	Vd.1D -> result	v7/A32/A64
int64x1_t_vreinterpret_s64_s32(int32x2_t a)	a -> Vd.2S	NOP	Vd.1D -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float64x1 tvreinterpret_f64_s32(int32x2 t a)	a -> Vd.2S	NOP	Vd.1D -> result	A64
poly64x1 tvreinterpret_p64_s32(int32x2 t a)	a -> Vd.2S	NOP	Vd.1D -> result	A32/A64
float16x4 tvreinterpret_f16_s32(int32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
int8x8 tvreinterpret_s8_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.8B -> result	v7/A32/A64
int16x4 tvreinterpret_s16_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
int32x2 tvreinterpret_s32_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.2S -> result	v7/A32/A64
uint8x8 tvreinterpret_u8_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.8B -> result	v7/A32/A64
uint16x4 tvreinterpret_u16_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
uint32x2 tvreinterpret_u32_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.2S -> result	v7/A32/A64
poly8x8 tvreinterpret_p8_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.8B -> result	v7/A32/A64
poly16x4 tvreinterpret_p16_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
uint64x1 tvreinterpret_u64_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.1D -> result	v7/A32/A64
int64x1 tvreinterpret_s64_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.1D -> result	v7/A32/A64
float64x1 tvreinterpret_f64_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.1D -> result	A64
poly64x1 tvreinterpret_p64_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.1D -> result	A32/A64
poly64x1 tvreinterpret_p64_f64(float64x1 t a)	a -> Vd.1D	NOP	Vd.1D -> result	A64
float16x4 tvreinterpret_f16_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
int8x8 tvreinterpret_s8_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.8B -> result	v7/A32/A64
int16x4 tvreinterpret_s16_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
int32x2 tvreinterpret_s32_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.2S -> result	v7/A32/A64
float32x2 tvreinterpret_f32_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.2S -> result	v7/A32/A64
uint16x4 tvreinterpret_u16_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
uint32x2 tvreinterpret_u32_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.2S -> result	v7/A32/A64
poly8x8 tvreinterpret_p8_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.8B -> result	v7/A32/A64
poly16x4 tvreinterpret_p16_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
uint64x1 tvreinterpret_u64_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.1D -> result	v7/A32/A64
int64x1 tvreinterpret_s64_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.1D -> result	v7/A32/A64
float64x1 tvreinterpret_f64_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.1D -> result	A64
poly64x1 tvreinterpret_p64_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.1D -> result	A32/A64
float16x4 tvreinterpret_f16_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
int8x8 tvreinterpret_s8_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
int16x4 tvreinterpret_s16_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64
int32x2 tvreinterpret_s32_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
float32x2 tvreinterpret_f32_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
uint8x8 tvreinterpret_u8_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
uint32x2 tvreinterpret_u32_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
poly8x8 tvreinterpret_p8_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
poly16x4 tvreinterpret_p16_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64
uint64x1 tvreinterpret_u64_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.1D -> result	v7/A32/A64
int64x1 tvreinterpret_s64_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.1D -> result	v7/A32/A64
float64x1 tvreinterpret_f64_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.1D -> result	A64
poly64x1 tvreinterpret_p64_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.1D -> result	A32/A64
float16x4 tvreinterpret_f16_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64
int8x8 tvreinterpret_s8_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.8B -> result	v7/A32/A64
int16x4 tvreinterpret_s16_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
int32x2 tvreinterpret_s32_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.2S -> result	v7/A32/A64
float32x2 tvreinterpret_f32_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.2S -> result	v7/A32/A64
uint8x8 tvreinterpret_u8_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.8B -> result	v7/A32/A64
uint16x4 tvreinterpret_u16_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
poly8x8 tvreinterpret_p8_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.8B -> result	v7/A32/A64
poly16x4 tvreinterpret_p16_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
uint64x1 tvreinterpret_u64_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.1D -> result	v7/A32/A64
int64x1 tvreinterpret_s64_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.1D -> result	v7/A32/A64
float64x1 tvreinterpret_f64_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.1D -> result	A64
poly64x1 tvreinterpret_p64_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.1D -> result	A32/A64
float16x4 tvreinterpret_f16_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	v7/A32/A64
int8x8 tvreinterpret_s8_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.8B -> result	v7/A32/A64
int16x4 tvreinterpret_s16_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
int32x2 tvreinterpret_s32_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.2S -> result	v7/A32/A64
float32x2 tvreinterpret_f32_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.2S -> result	v7/A32/A64
uint8x8 tvreinterpret_u8_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.8B -> result	v7/A32/A64
uint16x4 tvreinterpret_u16_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
uint32x2 tvreinterpret_u32_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.2S -> result	v7/A32/A64
poly16x4 tvreinterpret_p16_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
uint64x1 tvreinterpret_u64_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.1D -> result	v7/A32/A64
int64x1 tvreinterpret_s64_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.1D -> result	v7/A32/A64
float64x1 tvreinterpret_f64_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.1D -> result	A64
poly64x1 tvreinterpret_p64_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.1D -> result	A32/A64
float16x4 tvreinterpret_f16_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.4H -> result	v7/A32/A64
int8x8 tvreinterpret_s8_p16(poly16x4 t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
int16x4 tvreinterpret_s16_p16(poly16x4 t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x2_tvreinterpret_s32_p16(poly16x4_t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
float32x2_tvreinterpret_f32_p16(poly16x4_t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
uint8x8_tvreinterpret_u8_p16(poly16x4_t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
uint16x4_tvreinterpret_u16_p16(poly16x4_t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64
uint32x2_tvreinterpret_u32_p16(poly16x4_t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
poly8x8_tvreinterpret_p8_p16(poly16x4_t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
uint64x1_tvreinterpret_u64_p16(poly16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	v7/A32/A64
int64x1_tvreinterpret_s64_p16(poly16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	v7/A32/A64
float64x1_tvreinterpret_f64_p16(poly16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	A64
poly64x1_tvreinterpret_p64_p16(poly16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	A32/A64
float16x4_tvreinterpret_f16_p16(poly16x4_t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64
int8x8_tvreinterpret_s8_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	v7/A32/A64
int16x4_tvreinterpret_s16_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	v7/A32/A64
int32x2_tvreinterpret_s32_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.2S -> result	v7/A32/A64
float32x2_tvreinterpret_f32_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.2S -> result	v7/A32/A64
uint8x8_tvreinterpret_u8_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	v7/A32/A64
uint16x4_tvreinterpret_u16_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	v7/A32/A64
uint32x2_tvreinterpret_u32_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.2S -> result	v7/A32/A64
poly8x8_tvreinterpret_p8_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	v7/A32/A64
poly16x4_tvreinterpret_p16_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	v7/A32/A64
int64x1_tvreinterpret_s64_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.1D -> result	v7/A32/A64
float64x1_tvreinterpret_f64_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.1D -> result	A64
poly64x1_tvreinterpret_p64_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.1D -> result	A32/A64
float16x4_tvreinterpret_f16_u64(uint64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	v7/A32/A64
int8x8_tvreinterpret_s8_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	v7/A32/A64
int16x4_tvreinterpret_s16_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	v7/A32/A64
int32x2_tvreinterpret_s32_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.2S -> result	v7/A32/A64
float32x2_tvreinterpret_f32_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.2S -> result	v7/A32/A64
uint8x8_tvreinterpret_u8_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	v7/A32/A64
uint16x4_tvreinterpret_u16_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	v7/A32/A64
uint32x2_tvreinterpret_u32_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.2S -> result	v7/A32/A64
poly8x8_tvreinterpret_p8_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	v7/A32/A64
poly16x4_tvreinterpret_p16_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	v7/A32/A64
uint64x1_tvreinterpret_u64_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.1D -> result	v7/A32/A64
float64x1_tvreinterpret_f64_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.1D -> result	A64
uint64x1_tvreinterpret_u64_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.1D -> result	A32/A64
float16x4_tvreinterpret_f16_s64(int64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	v7/A32/A64
int8x8_tvreinterpret_s8_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
int16x4_tvreinterpret_s16_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64
int32x2_tvreinterpret_s32_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
float32x2_tvreinterpret_f32_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
uint8x8_tvreinterpret_u8_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
uint16x4_tvreinterpret_u16_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64
uint32x2_tvreinterpret_u32_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.2S -> result	v7/A32/A64
poly8x8_tvreinterpret_p8_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.8B -> result	v7/A32/A64
poly16x4_tvreinterpret_p16_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.4H -> result	v7/A32/A64
uint64x1_tvreinterpret_u64_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	v7/A32/A64
int64x1_tvreinterpret_s64_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	v7/A32/A64
float64x1_tvreinterpret_f64_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	A64
poly64x1_tvreinterpret_p64_f16(float16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	A32/A64
int16x8_tvreinterpretq_s16_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
int32x4_tvreinterpretq_s32_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.4S -> result	v7/A32/A64
float32x4_tvreinterpretq_f32_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.4S -> result	v7/A32/A64
uint8x16_tvreinterpretq_u8_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.16B -> result	v7/A32/A64
uint16x8_tvreinterpretq_u16_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
uint32x4_tvreinterpretq_u32_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.4S -> result	v7/A32/A64
poly8x16_tvreinterpretq_p8_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.16B -> result	v7/A32/A64
poly16x8_tvreinterpretq_p16_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
uint64x2_tvreinterpretq_u64_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	v7/A32/A64
int64x2_tvreinterpretq_s64_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	v7/A32/A64
float64x2_tvreinterpretq_f64_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	A64
poly64x2_tvreinterpretq_p64_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	A32/A64
poly128_tvreinterpretq_p128_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.1Q -> result	A32/A64
float16x8_tvreinterpretq_f16_s8(int8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
int8x16_tvreinterpretq_s8_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
int32x4_tvreinterpretq_s32_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64
float32x4_tvreinterpretq_f32_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t vreinterpretq_u8_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
uint16x8_t vreinterpretq_u16_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
uint32x4_t vreinterpretq_u32_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64
poly8x16_t vreinterpretq_p8_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
poly16x8_t vreinterpretq_p16_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
uint64x2_t vreinterpretq_u64_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	v7/A32/A64
int64x2_t vreinterpretq_s64_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	v7/A32/A64
float64x2_t vreinterpretq_f64_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A64
poly64x2_t vreinterpretq_p64_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A32/A64
poly128_t vreinterpretq_p128_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.1Q -> result	A32/A64
float16x8_t vreinterpretq_f16_s16(int16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
int8x16_t vreinterpretq_s8_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.16B -> result	v7/A32/A64
int16x8_t vreinterpretq_s16_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
float32x4_t vreinterpretq_f32_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.4S -> result	v7/A32/A64
uint8x16_t vreinterpretq_u8_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.16B -> result	v7/A32/A64
uint16x8_t vreinterpretq_u16_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
uint32x4_t vreinterpretq_u32_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.4S -> result	v7/A32/A64
poly8x16_t vreinterpretq_p8_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.16B -> result	v7/A32/A64
poly16x8_t vreinterpretq_p16_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
uint64x2_t vreinterpretq_u64_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	v7/A32/A64
int64x2_t vreinterpretq_s64_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	v7/A32/A64
float64x2_t vreinterpretq_f64_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	A64
poly64x2_t vreinterpretq_p64_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	A32/A64
poly128_t vreinterpretq_p128_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.1Q -> result	A32/A64
float16x8_t vreinterpretq_f16_s32(int32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
int8x16_t vreinterpretq_s8_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.16B -> result	v7/A32/A64
int16x8_t vreinterpretq_s16_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
int32x4_t vreinterpretq_s32_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.4S -> result	v7/A32/A64
uint8x16_t vreinterpretq_u8_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.16B -> result	v7/A32/A64
uint16x8_t vreinterpretq_u16_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
uint32x4_t vreinterpretq_u32_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.4S -> result	v7/A32/A64
poly8x16_t vreinterpretq_p8_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.16B -> result	v7/A32/A64
poly16x8_t vreinterpretq_p16_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
uint64x2_t vreinterpretq_u64_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	v7/A32/A64
int64x2_t vreinterpretq_s64_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	v7/A32/A64
float64x2_t vreinterpretq_f64_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	A64
poly64x2_t vreinterpretq_p64_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	A32/A64
poly128_t vreinterpretq_p128_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.1Q -> result	A32/A64
poly64x2_t vreinterpretq_p64_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	A64
poly128_t vreinterpretq_p128_f64(float64x2_t a)	a -> Vd.1Q	NOP	Vd.2D -> result	A64
float16x8_t vreinterpretq_f16_f32(float32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
int8x16_t vreinterpretq_s8_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.16B -> result	v7/A32/A64
int16x8_t vreinterpretq_s16_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
int32x4_t vreinterpretq_s32_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.4S -> result	v7/A32/A64
float32x4_t vreinterpretq_f32_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.4S -> result	v7/A32/A64
uint16x8_t vreinterpretq_u16_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
uint32x4_t vreinterpretq_u32_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.4S -> result	v7/A32/A64
poly8x16_t vreinterpretq_p8_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.16B -> result	v7/A32/A64
poly16x8_t vreinterpretq_p16_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
uint64x2_t vreinterpretq_u64_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	v7/A32/A64
int64x2_t vreinterpretq_s64_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	v7/A32/A64
float64x2_t vreinterpretq_f64_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	A64
poly64x2_t vreinterpretq_p64_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	A32/A64
poly128_t vreinterpretq_p128_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.1Q -> result	A32/A64
float16x8_t vreinterpretq_f16_u8(uint8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
int8x16_t vreinterpretq_s8_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
int16x8_t vreinterpretq_s16_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
int32x4_t vreinterpretq_s32_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64
float32x4_t vreinterpretq_f32_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t vreinterpretq_u8_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
uint32x4_t vreinterpretq_u32_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64
poly8x16_t vreinterpretq_p8_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
poly16x8_t vreinterpretq_p16_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
uint64x2_t vreinterpretq_u64_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	v7/A32/A64
int64x2_t vreinterpretq_s64_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	v7/A32/A64
float64x2_t vreinterpretq_f64_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A64
poly64x2_t vreinterpretq_p64_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A32/A64
poly128_t vreinterpretq_p128_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.1Q -> result	A32/A64
float16x8_t vreinterpretq_f16_u16(uint16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
int8x16_t vreinterpretq_s8_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.16B -> result	v7/A32/A64
int16x8_t vreinterpretq_s16_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
int32x4_t vreinterpretq_s32_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.4S -> result	v7/A32/A64
float32x4_t vreinterpretq_f32_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.4S -> result	v7/A32/A64
uint8x16_t vreinterpretq_u8_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.16B -> result	v7/A32/A64
uint16x8_t vreinterpretq_u16_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
poly8x16_t vreinterpretq_p8_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.16B -> result	v7/A32/A64
poly16x8_t vreinterpretq_p16_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
uint64x2_t vreinterpretq_u64_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	v7/A32/A64
int64x2_t vreinterpretq_s64_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	v7/A32/A64
float64x2_t vreinterpretq_f64_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	A64
poly64x2_t vreinterpretq_p64_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.2D -> result	A32/A64
poly128_t vreinterpretq_p128_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.1Q -> result	A32/A64
float16x8_t vreinterpretq_f16_u32(uint32x4_t a)	a -> Vd.4S	NOP	Vd.8H -> result	v7/A32/A64
int8x16_t vreinterpretq_s8_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.16B -> result	v7/A32/A64
int16x8_t vreinterpretq_s16_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
int32x4_t vreinterpretq_s32_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.4S -> result	v7/A32/A64
float32x4_t vreinterpretq_f32_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.4S -> result	v7/A32/A64
uint8x16_t vreinterpretq_u8_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.16B -> result	v7/A32/A64
uint16x8_t vreinterpretq_u16_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
uint32x4_t vreinterpretq_u32_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.4S -> result	v7/A32/A64
poly16x8_t vreinterpretq_p16_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
uint64x2_t vreinterpretq_u64_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	v7/A32/A64
int64x2_t vreinterpretq_s64_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	v7/A32/A64
float64x2_t vreinterpretq_f64_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	A64
poly64x2_t vreinterpretq_p64_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.2D -> result	A32/A64
poly128_t vreinterpretq_p128_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.1Q -> result	A32/A64
float16x8_t vreinterpretq_f16_p8(poly8x16_t a)	a -> Vd.16B	NOP	Vd.8H -> result	v7/A32/A64
int8x16_t vreinterpretq_s8_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
int16x8_t vreinterpretq_s16_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
int32x4_t vreinterpretq_s32_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64
float32x4_t vreinterpretq_f32_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64
uint8x16_t vreinterpretq_u8_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
uint16x8_t vreinterpretq_u16_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
uint32x4_t vreinterpretq_u32_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64
poly8x16_t vreinterpretq_p8_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
uint64x2_t vreinterpretq_u64_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	v7/A32/A64
int64x2_t vreinterpretq_s64_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	v7/A32/A64
float64x2_t vreinterpretq_f64_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A64
poly64x2_t vreinterpretq_p64_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A32/A64
poly128_t vreinterpretq_p128_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.1Q -> result	A32/A64
float16x8_t vreinterpretq_f16_p16(poly16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
int8x16_t vreinterpretq_s8_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	v7/A32/A64
int16x8_t vreinterpretq_s16_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	v7/A32/A64
int32x4_t vreinterpretq_s32_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.4S -> result	v7/A32/A64
float32x4_t vreinterpretq_f32_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.4S -> result	v7/A32/A64
uint8x16_t vreinterpretq_u8_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	v7/A32/A64
uint16x8_t vreinterpretq_u16_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	v7/A32/A64
uint32x4_t vreinterpretq_u32_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.4S -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
poly8x16_tvreinterpretq_p8_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	v7/A32/A64
poly16x8_tvreinterpretq_p16_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	v7/A32/A64
int64x2_tvreinterpretq_s64_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	v7/A32/A64
float64x2_tvreinterpretq_f64_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	v7/A32/A64
float64x2_tvreinterpretq_f64_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	A64
poly64x2_tvreinterpretq_p64_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	A32/A64
poly128_tvreinterpretq_p128_s64(int64x2_t a)	a -> Vd.1Q	NOP	Vd.2D -> result	A32/A64
poly64x2_tvreinterpretq_p64_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	A32/A64
poly128_tvreinterpretq_p128_u64(uint64x2_t a)	a -> Vd.1Q	NOP	Vd.2D -> result	A32/A64
float16x8_tvreinterpretq_f16_u64(uint64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	v7/A32/A64
int8x16_tvreinterpretq_s8_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	v7/A32/A64
int16x8_tvreinterpretq_s16_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	v7/A32/A64
int32x4_tvreinterpretq_s32_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.4S -> result	v7/A32/A64
float32x4_tvreinterpretq_f32_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.4S -> result	v7/A32/A64
uint8x16_tvreinterpretq_u8_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	v7/A32/A64
uint16x8_tvreinterpretq_u16_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	v7/A32/A64
uint32x4_tvreinterpretq_u32_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.4S -> result	v7/A32/A64
poly8x16_tvreinterpretq_p8_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	v7/A32/A64
poly16x8_tvreinterpretq_p16_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	v7/A32/A64
uint64x2_tvreinterpretq_u64_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	v7/A32/A64
uint64x2_tvreinterpretq_u64_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	A32/A64
float16x8_tvreinterpretq_f16_s64(int64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	v7/A32/A64
int8x16_tvreinterpretq_s8_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
int16x8_tvreinterpretq_s16_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
int32x4_tvreinterpretq_s32_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64
float32x4_tvreinterpretq_f32_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64
uint8x16_tvreinterpretq_u8_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
uint16x8_tvreinterpretq_u16_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
uint32x4_tvreinterpretq_u32_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	v7/A32/A64
poly8x16_tvreinterpretq_p8_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	v7/A32/A64
poly16x8_tvreinterpretq_p16_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	v7/A32/A64
uint64x2_tvreinterpretq_u64_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	v7/A32/A64
int64x2_tvreinterpretq_s64_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	v7/A32/A64
float64x2_tvreinterpretq_f64_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A64
poly64x2_tvreinterpretq_p64_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A32/A64
poly128_tvreinterpretq_p128_f16(float16x8_t a)	a -> Vd.8H	NOP	Vd.1Q -> result	A32/A64
int8x8_tvreinterpret_s8_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	A64
int16x4_tvreinterpret_s16_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	A64
int32x2_tvreinterpret_s32_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.2S -> result	A64
uint8x8_tvreinterpret_u8_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	A64
uint16x4_tvreinterpret_u16_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	A64
uint32x2_tvreinterpret_u32_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.2S -> result	A64
poly8x8_tvreinterpret_p8_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	A64
poly16x4_tvreinterpret_p16_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	A64
uint64x1_tvreinterpret_u64_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.1D -> result	A64
int64x1_tvreinterpret_s64_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.1D -> result	A64
float16x4_tvreinterpret_f16_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	A64
float32x2_tvreinterpret_f32_f64(float64x1_t a)	a -> Vd.1D	NOP	Vd.2S -> result	A64
int8x16_tvreinterpretq_s8_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	A64
int16x8_tvreinterpretq_s16_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	A64
int32x4_tvreinterpretq_s32_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.4S -> result	A64
uint8x16_tvreinterpretq_u8_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	A64
uint16x8_tvreinterpretq_u16_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	A64
uint32x4_tvreinterpretq_u32_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.4S -> result	A64
poly8x16_tvreinterpretq_p8_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	A64
poly16x8_tvreinterpretq_p16_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	A64
uint64x2_tvreinterpretq_u64_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	A64
int64x2_tvreinterpretq_s64_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	A64
float16x8_tvreinterpretq_f16_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	A64
float32x4_tvreinterpretq_f32_f64(float64x2_t a)	a -> Vd.2D	NOP	Vd.4S -> result	A64
int8x8_tvreinterpret_s8_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x4_t vreinterpret_s16_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	A32/A64
int32x2_t vreinterpret_s32_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.2S -> result	A32/A64
uint8x8_t vreinterpret_u8_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	A32/A64
uint16x4_t vreinterpret_u16_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	A32/A64
uint32x2_t vreinterpret_u32_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.2S -> result	A32/A64
poly8x8_t vreinterpret_p8_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.8B -> result	A32/A64
poly16x4_t vreinterpret_p16_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	A32/A64
int64x1_t vreinterpret_s64_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.1D -> result	A32/A64
float64x1_t vreinterpret_f64_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.1D -> result	A64
float16x4_t vreinterpret_f16_p64(poly64x1_t a)	a -> Vd.1D	NOP	Vd.4H -> result	A32/A64
int8x16_t vreinterpretq_s8_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	A32/A64
int16x8_t vreinterpretq_s16_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	A32/A64
int32x4_t vreinterpretq_s32_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.4S -> result	A32/A64
uint8x16_t vreinterpretq_u8_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	A32/A64
uint16x8_t vreinterpretq_u16_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	A32/A64
uint32x4_t vreinterpretq_u32_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.4S -> result	A32/A64
poly8x16_t vreinterpretq_p8_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.16B -> result	A32/A64
poly16x8_t vreinterpretq_p16_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	A32/A64
int64x2_t vreinterpretq_s64_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	A32/A64
float64x2_t vreinterpretq_f64_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.2D -> result	A64
float16x8_t vreinterpretq_f16_p64(poly64x2_t a)	a -> Vd.2D	NOP	Vd.8H -> result	A32/A64
int8x16_t vreinterpretq_s8_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.16B -> result	A32/A64
int16x8_t vreinterpretq_s16_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.8H -> result	A32/A64
int32x4_t vreinterpretq_s32_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.4S -> result	A32/A64
uint8x16_t vreinterpretq_u8_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.16B -> result	A32/A64
uint16x8_t vreinterpretq_u16_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.8H -> result	A32/A64
uint32x4_t vreinterpretq_u32_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.4S -> result	A32/A64
poly8x16_t vreinterpretq_p8_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.16B -> result	A32/A64
poly16x8_t vreinterpretq_p16_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.8H -> result	A32/A64
uint64x2_t vreinterpretq_u64_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.2D -> result	A32/A64
int64x2_t vreinterpretq_s64_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.2D -> result	A32/A64
float64x2_t vreinterpretq_f64_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.2D -> result	A64
float16x8_t vreinterpretq_f16_p128(poly128_t a)	a -> Vd.1Q	NOP	Vd.8H -> result	A32/A64
poly128_t vldrq_p128(poly128_t const * ptr)	ptr -> Xn	LDR Qd,[Xn]	Qd -> result	A32/A64
void vstrq_p128(poly128_t * ptr, poly128_t val)	ptr -> Xn val -> Qt	STR Qt,[Xn]	void -> result	A32/A64
uint8x16_t vaeseq_u8(uint8x16_t data, uint8x16_t key)	data -> Vd.16B key -> Vn.16B	AESE Vd.16B,Vn.16B	Vd.16B -> result	A32/A64
uint8x16_t vaesdq_u8(uint8x16_t data, uint8x16_t key)	data -> Vd.16B key -> Vn.16B	AESD Vd.16B,Vn.16B	Vd.16B -> result	A32/A64
uint8x16_t vaesmcq_u8(uint8x16_t data)	data -> Vn.16B	AESMC Vd.16B,Vn.16B	Vd.16B -> result	A32/A64
uint8x16_t vaesimcq_u8(uint8x16_t data)	data -> Vn.16B	AESIMC Vd.16B,Vn.16B	Vd.16B -> result	A32/A64
uint32x4_t vsha1cq_u32(uint32x4_t hash_abcd, uint32_t hash_e, uint32x4_t wk)	hash_abcd -> Qd hash_e -> Sn wk -> Vm.4S	SHA1C Qd,Sn,Vm.4S	Qd -> result	A32/A64
uint32x4_t vsha1pq_u32(uint32x4_t hash_abcd, uint32_t hash_e, uint32x4_t wk)	hash_abcd -> Qd hash_e -> Sn wk -> Vm.4S	SHA1P Qd,Sn,Vm.4S	Qd -> result	A32/A64
uint32x4_t vsha1mq_u32(uint32x4_t hash_abcd, uint32_t hash_e, uint32x4_t wk)	hash_abcd -> Qd hash_e -> Sn wk -> Vm.4S	SHA1M Qd,Sn,Vm.4S	Qd -> result	A32/A64
uint32_t vsha1h_u32(uint32_t hash_e)	hash_e -> Sn	SHA1H Sd,Sn	Sd -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t vsha1su0q_u32(uint32x4_t w0_3, uint32x4_t w4_7, uint32x4_t w8_11)	w0_3 -> Vd.4S w4_7 -> Vn.4S w8_11 -> Vm.4S	SHA1SU0 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A32/A64
uint32x4_t vsha1sq_u32(uint32x4_t tw0_3, uint32x4_t tw12_15)	tw0_3 -> Vd.4S tw12_15 -> Vn.4S	SHA1SU1 Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
uint32x4_t vsha256hq_u32(uint32x4_t hash_abcd, uint32x4_t hash_efgh, uint32x4_t wk)	hash_abcd -> Qd hash_efgh -> Qn wk -> Vm.4S	SHA256H Qd,Qn,Vm.4S	Qd -> result	A32/A64
uint32x4_t vsha256h2q_u32(uint32x4_t hash_efgh, uint32x4_t hash_abcd, uint32x4_t wk)	hash_efgh -> Qd hash_abcd -> Qn wk -> Vm.4S	SHA256H2 Qd,Qn,Vm.4S	Qd -> result	A32/A64
uint32x4_t vsha256su0q_u32(uint32x4_t w0_3, uint32x4_t w4_7)	w0_3 -> Vd.4S w4_7 -> Vn.4S	SHA256SU0 Vd.4S,Vn.4S	Vd.4S -> result	A32/A64
uint32x4_t vsha256su1q_u32(uint32x4_t tw0_3, uint32x4_t tw8_11, uint32x4_t tw12_15)	tw0_3 -> Vd.4S tw8_11 -> Vn.4S tw12_15 -> Vm.4S	SHA256SU1 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A32/A64
poly128_t vmull_p64(poly64_t a, poly64_t b)	a -> Vn.1D b -> Vm.1D	PMULL Vd.1Q,Vn.1D,Vm.1D	Vd.1Q -> result	A32/A64
poly128_t vmull_high_p64(poly64x2_t a, poly64x2_t b)	a -> Vn.2D b -> Vm.2D	PMULL2 Vd.1Q,Vn.2D,Vm.2D	Vd.1Q -> result	A32/A64
poly8x8_t vadd_p8(poly8x8_t a, poly8x8_t b)	a -> Vn.8B b -> Vm.8B	EOR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
poly16x4_t vadd_p16(poly16x4_t a, poly16x4_t b)	a -> Vn.8B b -> Vm.8B	EOR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
poly64x1_t vadd_p64(poly64x1_t a, poly64x1_t b)	a -> Vn.8B b -> Vm.8B	EOR Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
poly8x16_t vaddq_p8(poly8x16_t a, poly8x16_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
poly16x8_t vaddq_p16(poly16x8_t a, poly16x8_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
poly64x2_t vaddq_p64(poly64x2_t a, poly64x2_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
poly128_t vaddq_p128(poly128_t a, poly128_t b)	a -> Vn.16B b -> Vm.16B	EOR Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
uint32_t __crc32b(uint32_t a, uint8_t b)	a -> Wn b -> Wm	CRC32B Wd,Wn,Wm	Wd -> result	A32/A64
uint32_t __crc32h(uint32_t a, uint16_t b)	a -> Wn b -> Wm	CRC32H Wd,Wn,Wm	Wd -> result	A32/A64
uint32_t __crc32w(uint32_t a, uint32_t b)	a -> Wn b -> Wm	CRC32W Wd,Wn,Wm	Wd -> result	A32/A64
uint32_t __crc32d(uint32_t a, uint64_t b)	a -> Wn b -> Xm	CRC32X Wd,Wn,Xm	Wd -> result	A32/A64
uint32_t __crc32cb(uint32_t a, uint8_t b)	a -> Wn b -> Wm	CRC32CB Wd,Wn,Wm	Wd -> result	A32/A64
uint32_t __crc32ch(uint32_t a, uint16_t b)	a -> Wn b -> Wm	CRC32CH Wd,Wn,Wm	Wd -> result	A32/A64
uint32_t __crc32cw(uint32_t a, uint32_t b)	a -> Wn b -> Wm	CRC32CW Wd,Wn,Wm	Wd -> result	A32/A64
uint32_t __crc32cd(uint32_t a, uint64_t b)	a -> Wn b -> Xm	CRC32CX Wd,Wn,Xm	Wd -> result	A32/A64
int16x4_t vqrdmlah_s16(int16x4_t a, int16x4_t b, int16x4_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.4H	SQRDMLAH Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
int32x2_t vqrdmlah_s32(int32x2_t a, int32x2_t b, int32x2_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.2S	SQRDMLAH Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
int16x8_t vqrdmlahq_s16(int16x8_t a, int16x8_t b, int16x8_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.8H	SQRDMLAH Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t vqrdmlahq_s32(int32x4_t a, int32x4_t b, int32x4_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.4S	SQRDMLAH Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
int16x4_t vqrdmlsh_s16(int16x4_t a, int16x4_t b, int16x4_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.4H	SQRDMLSH Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
int32x2_t vqrdmlsh_s32(int32x2_t a, int32x2_t b, int32x2_t c)	a -> Vd.2S b -> Vn.2S c -> Vm.2S	SQRDMLSH Vd.2S,Vn.2S,Vm.2S	Vd.2S -> result	A64
int16x8_t vqrdmlshq_s16(int16x8_t a, int16x8_t b, int16x8_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.8H	SQRDMLSH Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
int32x4_t vqrdmlshq_s32(int32x4_t a, int32x4_t b, int32x4_t c)	a -> Vd.4S b -> Vn.4S c -> Vm.4S	SQRDMLSH Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
int16x4_t vqrdmlah_lane_s16(int16x4_t a, int16x4_t b, int16x4_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	SQRDMLAH Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
int16x8_t vqrdmlahq_lane_s16(int16x8_t a, int16x8_t b, int16x4_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	SQRDMLAH Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
int16x4_t vqrdmlah_laneq_s16(int16x4_t a, int16x4_t b, int16x8_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SQRDMLAH Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
int16x8_t vqrdmlahq_laneq_s16(int16x8_t a, int16x8_t b, int16x8_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SQRDMLAH Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
int32x2_t vqrdmlah_lane_s32(int32x2_t a, int32x2_t b, int32x2_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	SQRDMLAH Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
int32x4_t vqrdmlahq_lane_s32(int32x4_t a, int32x4_t b, int32x2_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	SQRDMLAH Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
int32x2_t vqrdmlah_laneq_s32(int32x2_t a, int32x2_t b, int32x4_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SQRDMLAH Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
int32x4_t vqrdmlahq_laneq_s32(int32x4_t a, int32x4_t b, int32x4_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SQRDMLAH Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
int16x4_t vqrdmlsh_lane_s16(int16x4_t a, int16x4_t b, int16x4_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	SQRDMLSH Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
int16x8_t vqrdmlshq_lane_s16(int16x8_t a, int16x8_t b, int16x4_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	SQRDMLSH Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
int16x4_t vqrdmlsh_laneq_s16(int16x4_t a, int16x4_t b, int16x8_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SQRDMLSH Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t vqrdmlshq_laneq_s16(int16x8_t a, int16x8_t b, int16x8_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	SQRDMLSH Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
int32x2_t vqrdmlsh_lane_s32(int32x2_t a, int32x2_t b, int32x2_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.2S 0 <= lane <= 1	SQRDMLSH Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
int32x4_t vqrdmlshq_lane_s32(int32x4_t a, int32x4_t b, int32x2_t v, const int lane)	a -> Vd.4S b -> Vn.4S v -> Vm.2S 0 <= lane <= 1	SQRDMLSH Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
int32x2_t vqrdmlsh_laneq_s32(int32x2_t a, int32x2_t b, int32x4_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SQRDMLSH Vd.2S,Vn.2S,Vm.S[lane]	Vd.2S -> result	A64
int32x4_t vqrdmlshq_laneq_s32(int32x4_t a, int32x4_t b, int32x4_t v, const int lane)	a -> Vd.2S b -> Vn.2S v -> Vm.4S 0 <= lane <= 3	SQRDMLSH Vd.4S,Vn.4S,Vm.S[lane]	Vd.4S -> result	A64
int16_t vqrdmlahh_s16(int16_t a, int16_t b, int16_t c)	a -> Hd b -> Hn c -> Hm	SQRDMLSH Hd,Hn,Hm	Hd -> result	A64
int32_t vqrdmlahs_s32(int32_t a, int32_t b, int32_t c)	a -> Sd b -> Sn c -> Sm	SQRDMLSH Sd,Sn,Sm	Sd -> result	A64
int16_t vqrdmlshh_s16(int16_t a, int16_t b, int16_t c)	a -> Hd b -> Hn c -> Hm	SQRDMLSH Hd,Hn,Hm	Hd -> result	A64
int32_t vqrdmlshs_s32(int32_t a, int32_t b, int32_t c)	a -> Sd b -> Sn c -> Sm	SQRDMLSH Sd,Sn,Sm	Sd -> result	A64
int16_t vqrdmlahh_lane_s16(int16_t a, int16_t b, int16x4_t v, const int lane)	a -> Hd b -> Hn v -> Vm.4H 0 <= lane <= 3	SQRDMLAH Hd,Hn,Vm.H[lane]	Hd -> result	A64
int16_t vqrdmlahh_laneq_s16(int16_t a, int16_t b, int16x8_t v, const int lane)	a -> Hd b -> Hn v -> Vm.8H 0 <= lane <= 7	SQRDMLAH Hd,Hn,Vm.H[lane]	Hd -> result	A64
int32_t vqrdmlahs_lane_s32(int32_t a, int32_t b, int32x2_t v, const int lane)	a -> Sd b -> Sn v -> Vm.2S 0 <= lane <= 1	SQRDMLAH Sd,Sn,Vm.S[lane]	Sd -> result	A64
int32_t vqrdmlahs_laneq_s32(int32_t a, int32_t b, int32x4_t v, const int lane)	a -> Sd b -> Sn v -> Vm.4S 0 <= lane <= 3	SQRDMLAH Sd,Sn,Vm.S[lane]	Sd -> result	A64
int16_t vqrdmlshh_lane_s16(int16_t a, int16_t b, int16x4_t v, const int lane)	a -> Hd b -> Hn v -> Vm.4H 0 <= lane <= 3	SQRDMLSH Hd,Hn,Vm.H[lane]	Hd -> result	A64
int16_t vqrdmlshh_laneq_s16(int16_t a, int16_t b, int16x8_t v, const int lane)	a -> Hd b -> Hn v -> Vm.8H 0 <= lane <= 7	SQRDMLSH Hd,Hn,Vm.H[lane]	Hd -> result	A64
int32_t vqrdmlshs_lane_s32(int32_t a, int32_t b, int32x2_t v, const int lane)	a -> Sd b -> Sn v -> Vm.2S 0 <= lane <= 1	SQRDMLSH Sd,Sn,Vm.S[lane]	Sd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32_t vqrdmlshs_laneq_s32(int32_t a, int32_t b, int32x4_t v, const int lane)	a -> Sd b -> Sn v -> Vm.4S 0 <= lane <= 3	SQRDMLSH Sd,Sn,Vm.S[lane]	Sd -> result	A64
float16_t vabsh_f16(float16_t a)	a -> Hn	FABS Hd,Hn	Hd -> result	A32/A64
uint16_t vceqzh_f16(float16_t a)	a -> Hn	FCMEQ Hd,Hn,#0	Hd -> result	A64
uint16_t vceqzh_f16(float16_t a)	a -> Hn	FCMGE Hd,Hn,#0	Hd -> result	A64
uint16_t vcgtzh_f16(float16_t a)	a -> Hn	FCMGT Hd,Hn,#0	Hd -> result	A64
uint16_t vclezh_f16(float16_t a)	a -> Hn	FCMLE Hd,Hn,#0	Hd -> result	A64
uint16_t vccltzh_f16(float16_t a)	a -> Hn	FCMLT Hd,Hn,#0	Hd -> result	A64
float16_t vcvth_f16_s16(int16_t a)	a -> Hn	SCVTF Hd,Hn	Hd -> result	A64
float16_t vcvth_f16_s32(int32_t a)	a -> Hn	SCVTF Hd,Hn	Hd -> result	A32/A64
float16_t vcvth_f16_s64(int64_t a)	a -> Hn	SCVTF Hd,Hn	Hd -> result	A64
float16_t vcvth_s32_f16(float16_t a)	a -> Hn	UCVTF Hd,Hn	Hd -> result	A64
float16_t vcvth_f16_u32(uint32_t a)	a -> Hn	UCVTF Hd,Hn	Hd -> result	A32/A64
float16_t vcvth_f16_u64(uint64_t a)	a -> Hn	UCVTF Hd,Hn	Hd -> result	A64
int16_t vcvth_s16_f16(float16_t a)	a -> Hn	FCVTZS Hd,Hn	Hd -> result	A64
int32_t vcvth_s32_f16(float16_t a)	a -> Hn	FCVTZS Hd,Hn	Hd -> result	A32/A64
int64_t vcvth_s64_f16(float16_t a)	a -> Hn	FCVTZS Hd,Hn	Hd -> result	A64
uint16_t vcvth_u16_f16(float16_t a)	a -> Hn	FCVTZU Hd,Hn	Hd -> result	A64
uint32_t vcvth_u32_f16(float16_t a)	a -> Hn	FCVTZU Hd,Hn	Hd -> result	A32/A64
uint64_t vcvth_u64_f16(float16_t a)	a -> Hn	FCVTZU Hd,Hn	Hd -> result	A64
int16_t vcvth_s16_f16(float16_t a)	a -> Hn	FCVTAS Hd,Hn	Hd -> result	A64
int32_t vcvth_s32_f16(float16_t a)	a -> Hn	FCVTAS Hd,Hn	Hd -> result	A32/A64
int64_t vcvth_s64_f16(float16_t a)	a -> Hn	FCVTAS Hd,Hn	Hd -> result	A64
uint16_t vcvth_u16_f16(float16_t a)	a -> Hn	FCVTAU Hd,Hn	Hd -> result	A64
uint32_t vcvth_u32_f16(float16_t a)	a -> Hn	FCVTAU Hd,Hn	Hd -> result	A32/A64
uint64_t vcvth_u64_f16(float16_t a)	a -> Hn	FCVTAU Hd,Hn	Hd -> result	A64
int16_t vcvtmh_s16_f16(float16_t a)	a -> Hn	FCVTMS Hd,Hn	Hd -> result	A64
int32_t vcvtmh_s32_f16(float16_t a)	a -> Hn	FCVTMS Hd,Hn	Hd -> result	A32/A64
int64_t vcvtmh_s64_f16(float16_t a)	a -> Hn	FCVTMS Hd,Hn	Hd -> result	A64
uint16_t vcvtmh_u16_f16(float16_t a)	a -> Hn	FCVTMU Hd,Hn	Hd -> result	A64
uint32_t vcvtmh_u32_f16(float16_t a)	a -> Hn	FCVTMU Hd,Hn	Hd -> result	A32/A64
uint64_t vcvtmh_u64_f16(float16_t a)	a -> Hn	FCVTMU Hd,Hn	Hd -> result	A64
int16_t vcvtnh_s16_f16(float16_t a)	a -> Hn	FCVTNS Hd,Hn	Hd -> result	A64
int32_t vcvtnh_s32_f16(float16_t a)	a -> Hn	FCVTNS Hd,Hn	Hd -> result	A32/A64
int64_t vcvtnh_s64_f16(float16_t a)	a -> Hn	FCVTNS Hd,Hn	Hd -> result	A64
uint16_t vcvtnh_u16_f16(float16_t a)	a -> Hn	FCVTNU Hd,Hn	Hd -> result	A64
uint32_t vcvtnh_u32_f16(float16_t a)	a -> Hn	FCVTNU Hd,Hn	Hd -> result	A32/A64
uint64_t vcvtnh_u64_f16(float16_t a)	a -> Hn	FCVTNU Hd,Hn	Hd -> result	A64
int16_t vcvtph_s16_f16(float16_t a)	a -> Hn	FCVTPS Hd,Hn	Hd -> result	A64
int32_t vcvtph_s32_f16(float16_t a)	a -> Hn	FCVTPS Hd,Hn	Hd -> result	A32/A64
int64_t vcvtph_s64_f16(float16_t a)	a -> Hn	FCVTPS Hd,Hn	Hd -> result	A64
uint16_t vcvtph_u16_f16(float16_t a)	a -> Hn	FCVTPU Hd,Hn	Hd -> result	A64
uint32_t vcvtph_u32_f16(float16_t a)	a -> Hn	FCVTPU Hd,Hn	Hd -> result	A32/A64
uint64_t vcvtph_u64_f16(float16_t a)	a -> Hn	FCVTPU Hd,Hn	Hd -> result	A64
float16_t vnegh_f16(float16_t a)	a -> Hn	FNEG Hd,Hn	Hd -> result	A32/A64
float16_t vrecpeh_f16(float16_t a)	a -> Hn	FRECPE Hd,Hn	Hd -> result	A64
float16_t vrecpxh_f16(float16_t a)	a -> Hn	FRECPX Hd,Hn	Hd -> result	A64
float16_t vrndh_f16(float16_t a)	a -> Hn	FRINTZ Hd,Hn	Hd -> result	A32/A64
float16_t vrndah_f16(float16_t a)	a -> Hn	FRINTA Hd,Hn	Hd -> result	A32/A64
float16_t vrndih_f16(float16_t a)	a -> Hn	FRINTI Hd,Hn	Hd -> result	A32/A64
float16_t vrndmh_f16(float16_t a)	a -> Hn	FRINTM Hd,Hn	Hd -> result	A32/A64
float16_t vrndnh_f16(float16_t a)	a -> Hn	FRINTN Hd,Hn	Hd -> result	A32/A64
float16_t vrndph_f16(float16_t a)	a -> Hn	FRINTP Hd,Hn	Hd -> result	A32/A64
float16_t vrndxh_f16(float16_t a)	a -> Hn	FRINTX Hd,Hn	Hd -> result	A32/A64
float16_t vrsqrteh_f16(float16_t a)	a -> Hn	FRSQRT Hd,Hn	Hd -> result	A64
float16_t vrsqrth_f16(float16_t a)	a -> Hn	FSQRT Hd,Hn	Hd -> result	A32/A64
float16_t vaddh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FADD Hd,Hn,Hm	Hd -> result	A32/A64
float16_t vabdh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FABD (scalar) Hd,Hn,Hm	Hd -> result	A64
uint16_t vcageh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FACGE Hd,Hn,Hm	Hd -> result	A64
uint16_t vcagth_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FACGT Hd,Hn,Hm	Hd -> result	A64
uint16_t vcaleh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FACGE Hd,Hn,Hm	Hd -> result	A64
uint16_t vcalth_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FACGT Hd,Hn,Hm	Hd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16_t_vceqh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FCMEQ Hd,Hn,Hm	Hd -> result	A64
uint16_t_vcegh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FCMGE Hd,Hn,Hm	Hd -> result	A64
uint16_t_vcgth_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FCMGT Hd,Hn,Hm	Hd -> result	A64
uint16_t_vcleh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FCMGE Hd,Hn,Hm	Hd -> result	A64
uint16_t_vclth_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FCMGT Hd,Hn,Hm	Hd -> result	A64
float16_t_vcvth_n_f16_s16(int16_t a, const int n)	a -> Hn 1 <= n <= 16	SCVTF Hd,Hn,#n	Hd -> result	A64
float16_t_vcvth_n_f16_s32(int32_t a, const int n)	a -> Hn 1 <= n <= 16	SCVTF Hd,Hn,#n	Hd -> result	A32/A64
float16_t_vcvth_n_f16_s64(int64_t a, const int n)	a -> Hn 1 <= n <= 16	SCVTF Hd,Hn,#n	Hd -> result	A64
float16_t_vcvth_n_f16_u16(uint16_t a, const int n)	a -> Hn 1 <= n <= 16	UCVTF Hd,Hn,#n	Hd -> result	A64
float16_t_vcvth_n_f16_u32(uint32_t a, const int n)	a -> Hn 1 <= n <= 16	UCVTF Hd,Hn,#n	Hd -> result	A32/A64
float16_t_vcvth_n_f16_u64(uint64_t a, const int n)	a -> Hn 1 <= n <= 16	UCVTF Hd,Hn,#n	Hd -> result	A64
int16_t_vcvth_n_s16_f16(float16_t a, const int n)	a -> Hn 1 <= n <= 16	FCVTZS Hd,Hn,#n	Hd -> result	A64
int32_t_vcvth_n_s32_f16(float16_t a, const int n)	a -> Hn 1 <= n <= 16	FCVTZS Hd,Hn,#n	Hd -> result	A32/A64
int64_t_vcvth_n_s64_f16(float16_t a, const int n)	a -> Hn 1 <= n <= 16	FCVTZS Hd,Hn,#n	Hd -> result	A64
uint16_t_vcvth_n_u16_f16(float16_t a, const int n)	a -> Hn 1 <= n <= 16	FCVTZU Hd,Hn,#n	Hd -> result	A64
uint32_t_vcvth_n_u32_f16(float16_t a, const int n)	a -> Hn 1 <= n <= 16	FCVTZU Hd,Hn,#n	Hd -> result	A32/A64
uint64_t_vcvth_n_u64_f16(float16_t a, const int n)	a -> Hn 1 <= n <= 16	FCVTZU Hd,Hn,#n	Hd -> result	A64
float16_t_vdivh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FDIV Hd,Hn,Hm	Hd -> result	A32/A64
float16_t_vmaxh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FMAX Hd,Hn,Hm	Hd -> result	A64
float16_t_vmaxnmh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FMAXNM Hd,Hn,Hm	Hd -> result	A32/A64
float16_t_vminh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FMIN Hd,Hn,Hm	Hd -> result	A64
float16_t_vminnmh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FMINNM Hd,Hn,Hm	Hd -> result	A32/A64
float16_t_vmulh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FMUL Hd,Hn,Hm	Hd -> result	A32/A64
float16_t_vmulxh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FMULX Hd,Hn,Hm	Hd -> result	A64
float16_t_vrecpsh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FRECPS Hd,Hn,Hm	Hd -> result	A64
float16_t_vrsqrtsh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FRSQRTS Hd,Hn,Hm	Hd -> result	A64
float16_t_vsubh_f16(float16_t a, float16_t b)	a -> Hn b -> Hm	FSUB Hd,Hn,Hm	Hd -> result	A32/A64
float16_t_vfmah_f16(float16_t a, float16_t b, float16_t c)	a -> Ha b -> Hn c -> Hm	FMADD Hd,Hn,Hm,Ha	Hd -> result	A32/A64
float16_t_vfmsh_f16(float16_t a, float16_t b, float16_t c)	a -> Ha b -> Hn c -> Hm	FMSUB Hd,Hn,Hm,Ha	Hd -> result	A32/A64
float16x4_t_vabs_f16(float16x4_t a)	a -> Vn.4H	FABS Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
float16x8_t_vabsq_f16(float16x8_t a)	a -> Vn.8H	FABS Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
uint16x4_t_vceqz_f16(float16x4_t a)	a -> Vn.4H	FCMEQ Vd.4H,Vn.4H,#0	Vd.4H -> result	A32/A64
uint16x8_t_vceqzq_f16(float16x8_t a)	a -> Vn.8H	FCMEQ Vd.8H,Vn.8H,#0	Vd.8H -> result	A32/A64
uint16x4_t_vcgez_f16(float16x4_t a)	a -> Vn.4H	FCMGE Vd.4H,Vn.4H,#0	Vd.4H -> result	A32/A64
uint16x8_t_vcgezq_f16(float16x8_t a)	a -> Vn.8H	FCMGE Vd.8H,Vn.8H,#0	Vd.8H -> result	A32/A64
uint16x4_t_vcgtz_f16(float16x4_t a)	a -> Vn.4H	FCMGT Vd.4H,Vn.4H,#0	Vd.4H -> result	A32/A64
uint16x8_t_vcgtzq_f16(float16x8_t a)	a -> Vn.8H	FCMGT Vd.8H,Vn.8H,#0	Vd.8H -> result	A32/A64
uint16x4_t_vclez_f16(float16x4_t a)	a -> Vn.4H	FCMLE Vd.4H,Vn.4H,#0	Vd.4H -> result	A32/A64
uint16x8_t_vclezq_f16(float16x8_t a)	a -> Vn.8H	FCMLE Vd.8H,Vn.8H,#0	Vd.8H -> result	A32/A64
uint16x4_t_vcltz_f16(float16x4_t a)	a -> Vn.4H	FCMLT Vd.4H,Vn.4H,#0	Vd.4H -> result	A32/A64
uint16x8_t_vcltzq_f16(float16x8_t a)	a -> Vn.8H	FCMLT Vd.8H,Vn.8H,#0	Vd.8H -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x4_tvcvt_f16_s16(int16x4_t a)	a -> Vn.4H	SCVTF Vd.4H,Vn.4H,#0	Vd.4H -> result	A32/A64
float16x8_tvcvtq_f16_s16(int16x8_t a)	a -> Vn.8H	SCVTF Vd.8H,Vn.8H,#0	Vd.8H -> result	A32/A64
float16x4_tvcvt_f16_u16(uint16x4_t a)	a -> Vn.4H	UCVTF Vd.4H,Vn.4H,#0	Vd.4H -> result	A32/A64
float16x8_tvcvtq_f16_u16(uint16x8_t a)	a -> Vn.8H	UCVTF Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
int16x4_tvcvt_s16_f16(float16x4_t a)	a -> Vn.4H	FCVTZS Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
int16x8_tvcvtq_s16_f16(float16x8_t a)	a -> Vn.8H	FCVTZS Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
uint16x4_tvcvt_u16_f16(float16x4_t a)	a -> Vn.4H	FCVTZS Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
uint16x8_tvcvtq_u16_f16(float16x8_t a)	a -> Vn.8H	FCVTZS Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
int16x4_tvcvt_s16_f16(float16x4_t a)	a -> Vn.4H	FCVTAS Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
int16x8_tvcvtq_s16_f16(float16x8_t a)	a -> Vn.8H	FCVTAS Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
uint16x4_tvcvt_u16_f16(float16x4_t a)	a -> Vn.4H	FCVTAU Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
uint16x8_tvcvtq_u16_f16(float16x8_t a)	a -> Vn.8H	FCVTAU Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
int16x4_tvcvtm_s16_f16(float16x4_t a)	a -> Vn.4H	FCVTMS Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
int16x8_tvcvtmq_s16_f16(float16x8_t a)	a -> Vn.8H	FCVTMS Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
uint16x4_tvcvtm_u16_f16(float16x4_t a)	a -> Vn.4H	FCVTMU Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
uint16x8_tvcvtmq_u16_f16(float16x8_t a)	a -> Vn.8H	FCVTMU Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
int16x4_tvcvtn_s16_f16(float16x4_t a)	a -> Vn.4H	FCVTNS Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
int16x8_tvcvtnq_s16_f16(float16x8_t a)	a -> Vn.8H	FCVTNS Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
uint16x4_tvcvtn_u16_f16(float16x4_t a)	a -> Vn.4H	FCVTNU Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
uint16x8_tvcvtnq_u16_f16(float16x8_t a)	a -> Vn.8H	FCVTNU Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
int16x4_tvcvtp_s16_f16(float16x4_t a)	a -> Vn.4H	FCVTPS Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
int16x8_tvcvtpq_s16_f16(float16x8_t a)	a -> Vn.8H	FCVTPS Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
uint16x4_tvcvtp_u16_f16(float16x4_t a)	a -> Vn.4H	FCVTPU Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
uint16x8_tvcvtpq_u16_f16(float16x8_t a)	a -> Vn.8H	FCVTPU Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
float16x4_tvneg_f16(float16x4_t a)	a -> Vn.4H	FNEG Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
float16x8_tvnegq_f16(float16x8_t a)	a -> Vn.8H	FNEG Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
float16x4_tvrcpe_f16(float16x4_t a)	a -> Vn.4H	FRECPE Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
float16x8_tvrcpeq_f16(float16x8_t a)	a -> Vn.8H	FRECPE Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
float16x4_tvrnd_f16(float16x4_t a)	a -> Vn.4H	FRINTZ Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
float16x8_tvrndq_f16(float16x8_t a)	a -> Vn.8H	FRINTZ Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
float16x4_tvrnda_f16(float16x4_t a)	a -> Vn.4H	FRINTA Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
float16x8_tvrndaq_f16(float16x8_t a)	a -> Vn.8H	FRINTA Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
float16x4_tvrndi_f16(float16x4_t a)	a -> Vn.4H	FRINTI Vd.4H,Vn.4H	Vd.4H -> result	A64
float16x8_tvrndiq_f16(float16x8_t a)	a -> Vn.8H	FRINTI Vd.8H,Vn.8H	Vd.8H -> result	A64
float16x4_tvrndm_f16(float16x4_t a)	a -> Vn.4H	FRINTM Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
float16x8_tvrndmq_f16(float16x8_t a)	a -> Vn.8H	FRINTM Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
float16x4_tvrndn_f16(float16x4_t a)	a -> Vn.4H	FRINTN Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
float16x8_tvrndnq_f16(float16x8_t a)	a -> Vn.8H	FRINTN Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
float16x4_tvrndp_f16(float16x4_t a)	a -> Vn.4H	FRINTP Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
float16x8_tvrndpq_f16(float16x8_t a)	a -> Vn.8H	FRINTP Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
float16x4_tvrndx_f16(float16x4_t a)	a -> Vn.4H	FRINTX Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
float16x8_tvrndxq_f16(float16x8_t a)	a -> Vn.8H	FRINTX Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
float16x4_tvsqrte_f16(float16x4_t a)	a -> Vn.4H	FRSQRT Vd.4H,Vn.4H	Vd.4H -> result	A32/A64
float16x8_tvsqrteq_f16(float16x8_t a)	a -> Vn.8H	FRSQRT Vd.8H,Vn.8H	Vd.8H -> result	A32/A64
float16x4_tvsqrt_f16(float16x4_t a)	a -> Vn.4H	FSQRT Vd.4H,Vn.4H	Vd.4H -> result	A64
float16x8_tvsqrtq_f16(float16x8_t a)	a -> Vn.8H	FSQRT Vd.8H,Vn.8H	Vd.8H -> result	A64
float16x4_tvadd_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FADD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_tvaddq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FADD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_tvabdq_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FABD Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_tvabdq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FABD Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
uint16x4_tvage_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FACGE Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
uint16x8_tvageq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FACGE Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
uint16x4_tvagtg_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FACGT Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
uint16x8_tvagtgq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FACGT Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
uint16x4_vcale_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FACGE Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
uint16x8_vcaleq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FACGE Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
uint16x4_vcalt_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FACGT Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
uint16x8_vcaltq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FACGT Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x4_t vceq_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FCMEQ Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
uint16x8_t vceqq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FCMEQ Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
uint16x4_t vcge_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FCMGE Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
uint16x8_t vcgeq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FCMGE Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
uint16x4_t vcgt_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FCMGT Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
uint16x8_t vcgtq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FCMGT Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
uint16x4_t vcle_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FCMGE Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
uint16x8_t vcleq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FCMGE Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
uint16x4_t vcilt_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FCMGT Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
uint16x8_t vciltq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FCMGT Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_t vcvt_n_f16_s16(int16x4_t a, const int n)	a -> Vn.4H 1 <= n <= 16	SCVTF Vd.4H,Vn.4H,#n	Vd.4H -> result	A32/A64
float16x8_t vcvtq_n_f16_s16(int16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 16	SCVTF Vd.8H,Vn.8H,#n	Vd.8H -> result	A32/A64
float16x4_t vcvt_n_f16_u16(uint16x4_t a, const int n)	a -> Vn.4H 1 <= n <= 16	UCVTF Vd.4H,Vn.4H,#n	Vd.4H -> result	A32/A64
float16x8_t vcvtq_n_f16_u16(uint16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 16	UCVTF Vd.8H,Vn.8H,#n	Vd.8H -> result	A32/A64
int16x4_t vcvt_n_s16_f16(float16x4_t a, const int n)	a -> Vn.4H 1 <= n <= 16	FCVTZS Vd.4H,Vn.4H,#n	Vd.4H -> result	A32/A64
int16x8_t vcvtq_n_s16_f16(float16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 16	FCVTZS Vd.8H,Vn.8H,#n	Vd.8H -> result	A32/A64
uint16x4_t vcvt_n_u16_f16(float16x4_t a, const int n)	a -> Vn.4H 1 <= n <= 16	FCVTZU Vd.4H,Vn.4H,#n	Vd.4H -> result	A32/A64
uint16x8_t vcvtq_n_u16_f16(float16x8_t a, const int n)	a -> Vn.8H 1 <= n <= 16	FCVTZU Vd.8H,Vn.8H,#n	Vd.8H -> result	A32/A64
float16x4_t vdiv_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FDIV Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
float16x8_t vdivq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FDIV Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vmax_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FMAX Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vmaxq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FMAX Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_t vmaxnm_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FMAXNM Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vmaxnmq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FMAXNM Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_t vmin_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FMIN Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vminq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FMIN Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_t vminnm_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FMINNM Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vminnmq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FMINNM Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_t vmul_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FMUL Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vmulq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FMUL Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_t vmulx_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FMULX Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
float16x8_t vmulxq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FMULX Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vpadd_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FADDP Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vpaddq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FADDP Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vpmulx_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FMAXP Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vpmulxq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FMAXP Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x4_t vpmxnm_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FMAXNMP Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
float16x8_t vpmxnmq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FMAXNMP Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vpmin_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FMINP Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vpminq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FMINP Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vpminnm_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FMINNMP Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
float16x8_t vpminnmq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FMINNMP Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vrecps_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FRECPS Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vrecpsq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FRECPS Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_t vrsqrts_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FRSQRTS Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vrsqrtsq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FRSQRTS Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_t vsub_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FSUB Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vsubq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FSUB Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_t vfma_f16(float16x4_t a, float16x4_t b, float16x4_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.4H	FMLA Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vfmaq_f16(float16x8_t a, float16x8_t b, float16x8_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.8H	FMLA Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_t vfms_f16(float16x4_t a, float16x4_t b, float16x4_t c)	a -> Vd.4H b -> Vn.4H c -> Vm.4H	FMLS Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A32/A64
float16x8_t vfmsq_f16(float16x8_t a, float16x8_t b, float16x8_t c)	a -> Vd.8H b -> Vn.8H c -> Vm.8H	FMLS Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A32/A64
float16x4_t vfma_lane_f16(float16x4_t a, float16x4_t b, float16x4_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	FMLA Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
float16x8_t vfmaq_lane_f16(float16x8_t a, float16x8_t b, float16x4_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	FMLA Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
float16x4_t vfma_laneq_f16(float16x4_t a, float16x4_t b, float16x8_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	FMLA Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
float16x8_t vfmaq_laneq_f16(float16x8_t a, float16x8_t b, float16x8_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	FMLA Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
float16x4_t vfma_n_f16(float16x4_t a, float16x4_t b, float16_t n)	a -> Vd.4H b -> Vn.4H n -> Vm.H[0]	FMLA Vd.4H,Vn.4H,Vm.H[0]	Vd.4H -> result	A64
float16x8_t vfmaq_n_f16(float16x8_t a, float16x8_t b, float16_t n)	a -> Vd.8H b -> Vn.8H n -> Vm.H[0]	FMLA Vd.8H,Vn.8H,Vm.H[0]	Vd.8H -> result	A64
float16_t vfma_lane_f16(float16_t a, float16_t b, float16x4_t v, const int lane)	a -> Hd b -> Hn v -> Vm.4H 0 <= lane <= 3	FMLA Hd,Hn,Vm.H[lane]	Hd -> result	A64
float16_t vfma_laneq_f16(float16_t a, float16_t b, float16x8_t v, const int lane)	a -> Hd b -> Hn v -> Vm.8H 0 <= lane <= 7	FMLA Hd,Hn,Vm.H[lane]	Hd -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x4_t vfms_lane_f16(float16x4_t a, float16x4_t b, float16x4_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.4H 0 <= lane <= 3	FMLS Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
float16x8_t vfmsq_lane_f16(float16x8_t a, float16x8_t b, float16x4_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.4H 0 <= lane <= 3	FMLS Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
float16x4_t vfms_laneq_f16(float16x4_t a, float16x4_t b, float16x8_t v, const int lane)	a -> Vd.4H b -> Vn.4H v -> Vm.8H 0 <= lane <= 7	FMLS Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
float16x8_t vfmsq_laneq_f16(float16x8_t a, float16x8_t b, float16x8_t v, const int lane)	a -> Vd.8H b -> Vn.8H v -> Vm.8H 0 <= lane <= 7	FMLS Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
float16x4_t vfms_n_f16(float16x4_t a, float16x4_t b, float16_t n)	a -> Vd.4H b -> Vn.4H n -> Vm.H[0]	FMLS Vd.4H,Vn.4H,Vm.H[0]	Vd.4H -> result	A64
float16x8_t vfmsq_n_f16(float16x8_t a, float16x8_t b, float16_t n)	a -> Vd.8H b -> Vn.8H n -> Vm.H[0]	FMLS Vd.8H,Vn.8H,Vm.H[0]	Vd.8H -> result	A64
float16_t vfms_h_lane_f16(float16_t a, float16_t b, float16x4_t v, const int lane)	a -> Hd b -> Hn v -> Vm.4H 0 <= lane <= 3	FMLS Hd,Hn,Vm.H[lane]	Hd -> result	A64
float16_t vfms_h_laneq_f16(float16_t a, float16_t b, float16x8_t v, const int lane)	a -> Hd b -> Hn v -> Vm.8H 0 <= lane <= 7	FMLS Hd,Hn,Vm.H[lane]	Hd -> result	A64
float16x4_t vmul_lane_f16(float16x4_t a, float16x4_t v, const int lane)	a -> Vn.4H v -> Vm.4H 0 <= lane <= 3	FMUL Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A32/A64
float16x8_t vmulq_lane_f16(float16x8_t a, float16x4_t v, const int lane)	a -> Vn.8H v -> Vm.4H 0 <= lane <= 3	FMUL Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A32/A64
float16x4_t vmul_laneq_f16(float16x4_t a, float16x8_t v, const int lane)	a -> Vn.4H v -> Vm.8H 0 <= lane <= 7	FMUL Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
float16x8_t vmulq_laneq_f16(float16x8_t a, float16x8_t v, const int lane)	a -> Vn.8H v -> Vm.8H 0 <= lane <= 7	FMUL Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
float16x4_t vmul_n_f16(float16x4_t a, float16_t n)	a -> Vn.4H n -> Vm.H[0]	FMUL Vd.4H,Vn.4H,Vm.H[0]	Vd.4H -> result	A32/A64
float16x8_t vmulq_n_f16(float16x8_t a, float16_t n)	a -> Vn.8H n -> Vm.H[0]	FMUL Vd.8H,Vn.8H,Vm.H[0]	Vd.8H -> result	A32/A64
float16_t vmul_h_lane_f16(float16_t a, float16x4_t v, const int lane)	a -> Hn v -> Vm.4H 0 <= lane <= 3	FMUL Hd,Hn,Vm.H[lane]	Hd -> result	A64
float16_t vmul_h_laneq_f16(float16_t a, float16x8_t v, const int lane)	a -> Hn v -> Vm.8H 0 <= lane <= 7	FMUL Hd,Hn,Vm.H[lane]	Hd -> result	A64
float16x4_t vmulx_lane_f16(float16x4_t a, float16x4_t v, const int lane)	a -> Vn.4H v -> Vm.4H 0 <= lane <= 3	FMULX Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
float16x8_t vmulxq_lane_f16(float16x8_t a, float16x4_t v, const int lane)	a -> Vn.8H v -> Vm.4H 0 <= lane <= 3	FMULX Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x4_t vmulx_laneq_f16(float16x4_t a, float16x8_t v, const int lane)	a -> Vn.4H v -> Vm.8H 0 <= lane <= 7	FMULX Vd.4H,Vn.4H,Vm.H[lane]	Vd.4H -> result	A64
float16x8_t vmulxq_laneq_f16(float16x8_t a, float16x8_t v, const int lane)	a -> Vn.8H v -> Vm.8H 0 <= lane <= 7	FMULX Vd.8H,Vn.8H,Vm.H[lane]	Vd.8H -> result	A64
float16x4_t vmulx_n_f16(float16x4_t a, float16_t n)	a -> Vn.4H n -> Vm.H[0]	FMULX Vd.4H,Vn.4H,Vm.H[0]	Vd.4H -> result	A64
float16x8_t vmulxq_n_f16(float16x8_t a, float16_t n)	a -> Vn.8H n -> Vm.H[0]	FMULX Vd.8H,Vn.8H,Vm.H[0]	Vd.8H -> result	A64
float16_t vmulxh_lane_f16(float16_t a, float16x4_t v, const int lane)	a -> Hn v -> Vm.4H 0 <= lane <= 3	FMULX Hd,Hn,Vm.H[lane]	Hd -> result	A64
float16_t vmulxh_laneq_f16(float16_t a, float16x8_t v, const int lane)	a -> Hn v -> Vm.8H 0 <= lane <= 7	FMULX Hd,Hn,Vm.H[lane]	Hd -> result	A64
float16_t vmaxv_f16(float16x4_t a)	a -> Vn.4H	FMAXP Hd,Vn.4H	Hd -> result	A64
float16_t vmaxvq_f16(float16x8_t a)	a -> Vn.8H	FMAXP Hd,Vn.8H	Hd -> result	A64
float16_t vminv_f16(float16x4_t a)	a -> Vn.4H	FMINP Hd,Vn.4H	Hd -> result	A64
float16_t vminvq_f16(float16x8_t a)	a -> Vn.8H	FMINP Hd,Vn.8H	Hd -> result	A64
float16_t vmaxnmv_f16(float16x4_t a)	a -> Vn.4H	FMAXNMP Hd,Vn.4H	Hd -> result	A64
float16_t vmaxnmvq_f16(float16x8_t a)	a -> Vn.8H	FMAXNMP Hd,Vn.8H	Hd -> result	A64
float16_t vminnmv_f16(float16x4_t a)	a -> Vn.4H	FMINNMP Hd,Vn.4H	Hd -> result	A64
float16_t vminnmvq_f16(float16x8_t a)	a -> Vn.8H	FMINNMP Hd,Vn.8H	Hd -> result	A64
float16x4_t vbsl_f16(uint16x4_t a, float16x4_t b, float16x4_t c)	a -> Vd.8B b -> Vn.8B c -> Vm.8B	BSL Vd.8B,Vn.8B,Vm.8B	Vd.8B -> result	v7/A32/A64
float16x8_t vbslq_f16(uint16x8_t a, float16x8_t b, float16x8_t c)	a -> Vd.16B b -> Vn.16B c -> Vm.16B	BSL Vd.16B,Vn.16B,Vm.16B	Vd.16B -> result	v7/A32/A64
float16x4x2_t vzip_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP1 Vd1.4H,Vn.4H,Vm.4H ZIP2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64
float16x8x2_t vzipq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP1 Vd1.8H,Vn.8H,Vm.8H ZIP2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
float16x4x2_t vuzp_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP1 Vd1.4H,Vn.4H,Vm.4H UZP2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64
float16x8x2_t vuzpq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP1 Vd1.8H,Vn.8H,Vm.8H UZP2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
float16x4x2_t vtrn_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN1 Vd1.4H,Vn.4H,Vm.4H TRN2 Vd2.4H,Vn.4H,Vm.4H	Vd1.4H -> result.val[0] Vd2.4H -> result.val[1]	v7/A32/A64
float16x8x2_t vtrnq_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN1 Vd1.8H,Vn.8H,Vm.8H TRN2 Vd2.8H,Vn.8H,Vm.8H	Vd1.8H -> result.val[0] Vd2.8H -> result.val[1]	v7/A32/A64
float16x4_t vmov_n_f16(float16_t value)	value -> rn	DUP Vd.4H,rn	Vd.4H -> result	v7/A32/A64
float16x8_t vmovq_n_f16(float16_t value)	value -> rn	DUP Vd.8H,rn	Vd.8H -> result	v7/A32/A64
float16x4_t vdup_n_f16(float16_t value)	value -> rn	DUP Vd.4H,rn	Vd.4H -> result	v7/A32/A64
float16x8_t vdupq_n_f16(float16_t value)	value -> rn	DUP Vd.8H,rn	Vd.8H -> result	v7/A32/A64
float16x4_t vdup_lane_f16(float16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Vd.4H,Vn.H[lane]	Vd.4H -> result	v7/A32/A64
float16x8_t vdupq_lane_f16(float16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Vd.8H,Vn.H[lane]	Vd.8H -> result	v7/A32/A64
float16x4_t vext_f16(float16x4_t a, float16x4_t b, const int n)	a -> Vn.8B b -> Vm.8B 0 <= n <= 3	EXT Vd.8B,Vn.8B,Vm.8B,(n<<1)	Vd.8B -> result	v7/A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t vextq_f16(float16x8_t a, float16x8_t b, const int n)	a -> Vn.16B b -> Vm.16B 0 <= n <= 7	EXT Vd.16B,Vn.16B,Vm.16B,#(n<<1)	Vd.16B -> result	v7/A32/A64
float16x4_t vrev64_f16(float16x4_t vec)	vec -> Vn.4H	REV64 Vd.4H,Vn.4H	Vd.4H -> result	v7/A32/A64
float16x8_t vrev64q_f16(float16x8_t vec)	vec -> Vn.8H	REV64 Vd.8H,Vn.8H	Vd.8H -> result	v7/A32/A64
float16x4_t vzip1_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP1 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
float16x8_t vzip1q_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP1 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vzip2_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	ZIP2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
float16x8_t vzip2q_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	ZIP2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vuzp1_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP1 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
float16x8_t vuzp1q_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP1 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vuzp2_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	UZP2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
float16x8_t vuzp2q_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	UZP2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vtrn1_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN1 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
float16x8_t vtrn1q_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN1 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vtrn2_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	TRN2 Vd.4H,Vn.4H,Vm.4H	Vd.4H -> result	A64
float16x8_t vtrn2q_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	TRN2 Vd.8H,Vn.8H,Vm.8H	Vd.8H -> result	A64
float16x4_t vdup_laneq_f16(float16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Vd.4H,Vn.H[lane]	Vd.4H -> result	A64
float16x8_t vdupq_laneq_f16(float16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Vd.8H,Vn.H[lane]	Vd.8H -> result	A64
float16_t vduph_lane_f16(float16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Hd,Vn.H[lane]	Hd -> result	A64
float16_t vduph_laneq_f16(float16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Hd,Vn.H[lane]	Hd -> result	A64
uint32x2_t vdot_u32(uint32x2_t r, uint8x8_t a, uint8x8_t b)	r -> Vd.2S a -> Vn.8B b -> Vm.8B	UDOT Vd.2S,Vn.8B,Vm.8B	Vd.2S -> result	A32/A64
int32x2_t vdot_s32(int32x2_t r, int8x8_t a, int8x8_t b)	r -> Vd.2S a -> Vn.8B b -> Vm.8B	SDOT Vd.2S,Vn.8B,Vm.8B	Vd.2S -> result	A32/A64
uint32x4_t vdotq_u32(uint32x4_t r, uint8x16_t a, uint8x16_t b)	r -> Vd.4S a -> Vn.16B b -> Vm.16B	UDOT Vd.4S,Vn.16B,Vm.16B	Vd.4S -> result	A32/A64
int32x4_t vdotq_s32(int32x4_t r, int8x16_t a, int8x16_t b)	r -> Vd.4S a -> Vn.16B b -> Vm.16B	SDOT Vd.4S,Vn.16B,Vm.16B	Vd.4S -> result	A32/A64
uint32x2_t vdot_lane_u32(uint32x2_t r, uint8x8_t a, uint8x8_t b, const int lane)	r -> Vd.2S a -> Vn.8B b -> Vm.4B 0 <= lane <= 1	UDOT Vd.2S,Vn.8B,Vm.4B[lane]	Vd.2S -> result	A32/A64
int32x2_t vdot_lane_s32(int32x2_t r, int8x8_t a, int8x8_t b, const int lane)	r -> Vd.2S a -> Vn.8B b -> Vm.4B 0 <= lane <= 1	SDOT Vd.2S,Vn.8B,Vm.4B[lane]	Vd.2S -> result	A32/A64
uint32x4_t vdotq_laneq_u32(uint32x4_t r, uint8x16_t a, uint8x16_t b, const int lane)	r -> Vd.4S a -> Vn.16B b -> Vm.4B 0 <= lane <= 3	UDOT Vd.4S,Vn.16B,Vm.4B[lane]	Vd.4S -> result	A64
int32x4_t vdotq_laneq_s32(int32x4_t r, int8x16_t a, int8x16_t b, const int lane)	r -> Vd.4S a -> Vn.16B b -> Vm.4B 0 <= lane <= 3	SDOT Vd.4S,Vn.16B,Vm.4B[lane]	Vd.4S -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x2_t vdot_laneq_u32(uint32x2_t r, uint8x8_t a, uint8x16_t b, const int lane)	r -> Vd.2S a -> Vn.8B b -> Vm.4B 0 <= lane <= 3	UDOT Vd.2S,Vn.8B,Vm.4B[lane]	Vd.2S -> result	A64
int32x2_t vdot_laneq_s32(int32x2_t r, int8x8_t a, int8x16_t b, const int lane)	r -> Vd.2S a -> Vn.8B b -> Vm.4B 0 <= lane <= 3	SDOT Vd.2S,Vn.8B,Vm.4B[lane]	Vd.2S -> result	A64
uint32x4_t vdotq_lane_u32(uint32x4_t r, uint8x16_t a, uint8x8_t b, const int lane)	r -> Vd.4S a -> Vn.16B b -> Vm.4B 0 <= lane <= 1	UDOT Vd.4S,Vn.16B,Vm.4B[lane]	Vd.4S -> result	A32/A64
int32x4_t vdotq_lane_s32(int32x4_t r, int8x16_t a, int8x8_t b, const int lane)	r -> Vd.4S a -> Vn.16B b -> Vm.4B 0 <= lane <= 1	SDOT Vd.4S,Vn.16B,Vm.4B[lane]	Vd.4S -> result	A32/A64
uint64x2_t vsha512hq_u64(uint64x2_t hash_ed, uint64x2_t hash_gf, uint64x2_t kwh_kwh2)	hash_ed -> Qd hash_gf -> Qn	SHA512H Qd,Qn,Vm.2D	Qd -> result	A64
uint64x2_t vsha512hq_u64(uint64x2_t sum_ab, uint64x2_t hash_c_, uint64x2_t hash_ab)	sum_ab -> Qd hash_c_ -> Qn	SHA512H2 Qd,Qn,Vm.2D	Qd -> result	A64
uint64x2_t vsha512su0q_u64(uint64x2_t w0_1, uint64x2_t w2_)	w0_1 -> Vd.2D w2_ -> Vn.2D	SHA512SU0 Vd.2D,Vn.2D	Vd.2D -> result	A64
uint64x2_t vsha512su1q_u64(uint64x2_t s01_s02, uint64x2_t w14_15, uint64x2_t w9_10)	s01_s02 -> Vd.2D w14_15 -> Vn.2D	SHA512SU1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint8x16_t veor3q_u8(uint8x16_t a, uint8x16_t b, uint8x16_t c)	a -> Vn.16B	EOR3 Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
uint16x8_t veor3q_u16(uint16x8_t a, uint16x8_t b, uint16x8_t c)	a -> Vn.16B	EOR3 Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
uint32x4_t veor3q_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c)	a -> Vn.16B	EOR3 Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
uint64x2_t veor3q_u64(uint64x2_t a, uint64x2_t b, uint64x2_t c)	a -> Vn.16B	EOR3 Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
int8x16_t veor3q_s8(int8x16_t a, int8x16_t b, int8x16_t c)	a -> Vn.16B	EOR3 Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
int16x8_t veor3q_s16(int16x8_t a, int16x8_t b, int16x8_t c)	a -> Vn.16B	EOR3 Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
int32x4_t veor3q_s32(int32x4_t a, int32x4_t b, int32x4_t c)	a -> Vn.16B	EOR3 Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
int64x2_t veor3q_s64(int64x2_t a, int64x2_t b, int64x2_t c)	a -> Vn.16B	EOR3 Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
uint64x2_t vrax1q_u64(uint64x2_t a, uint64x2_t b)	a -> Vn.2D	RAX1 Vd.2D,Vn.2D,Vm.2D	Vd.2D -> result	A64
uint64x2_t vxarq_u64(uint64x2_t a, uint64x2_t b, const int imm6)	a -> Vn.2D 0 <= imm6 <= 63	XAR Vd.2D,Vn.2D,Vm.2D,imm6	Vd.2D -> result	A64
uint8x16_t vbcaxq_u8(uint8x16_t a, uint8x16_t b, uint8x16_t c)	a -> Vn.16B	BCAX Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
uint16x8_t vbcaxq_u16(uint16x8_t a, uint16x8_t b, uint16x8_t c)	a -> Vn.16B	BCAX Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
uint32x4_t vbcaxq_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c)	a -> Vn.16B	BCAX Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
uint64x2_t vbcaxq_u64(uint64x2_t a, uint64x2_t b, uint64x2_t c)	a -> Vn.16B	BCAX Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
int8x16_t vbcaxq_s8(int8x16_t a, int8x16_t b, int8x16_t c)	a -> Vn.16B	BCAX Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
int16x8_t vbcaxq_s16(int16x8_t a, int16x8_t b, int16x8_t c)	a -> Vn.16B	BCAX Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
int32x4_t vbcaxq_s32(int32x4_t a, int32x4_t b, int32x4_t c)	a -> Vn.16B	BCAX Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64
int64x2_t vbcaxq_s64(int64x2_t a, int64x2_t b, int64x2_t c)	a -> Vn.16B	BCAX Vd.16B,Vn.16B,Vm.16B,Va.16B	Vd.16B -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t vsm3ss1q_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c)	a -> Vn.4S	SM3SS1 Vd.4S,Vn.4S,Vm.4S,Va.4S	Vd.4S -> result	A64
uint32x4_t vsm3tt1aq_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c, const int imm2)	a -> Vd.4S 0 <= imm2 <= 3	SM3TT1A Vd.4S,Vn.4S,Vm.4S[imm2]	Vd.4S -> result	A64
uint32x4_t vsm3tt1bq_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c, const int imm2)	a -> Vd.4S 0 <= imm2 <= 3	SM3TT1B Vd.4S,Vn.4S,Vm.4S[imm2]	Vd.4S -> result	A64
uint32x4_t vsm3tt2aq_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c, const int imm2)	a -> Vd.4S 0 <= imm2 <= 3	SM3TT2A Vd.4S,Vn.4S,Vm.4S[imm2]	Vd.4S -> result	A64
uint32x4_t vsm3tt2bq_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c, const int imm2)	a -> Vd.4S 0 <= imm2 <= 3	SM3TT2B Vd.4S,Vn.4S,Vm.4S[imm2]	Vd.4S -> result	A64
uint32x4_t vsm3partw1q_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c)	a -> Vd.4S	SM3PARTW1 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
uint32x4_t vsm3partw2q_u32(uint32x4_t a, uint32x4_t b, uint32x4_t c)	a -> Vd.4S	SM3PARTW2 Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
uint32x4_t vsm4eq_u32(uint32x4_t a, uint32x4_t b)	a -> Vd.4S	SM4E Vd.4S,Vn.4S	Vd.4S -> result	A64
uint32x4_t vsm4eqkey_u32(uint32x4_t a, uint32x4_t b)	a -> Vn.4S	SM4EKEY Vd.4S,Vn.4S,Vm.4S	Vd.4S -> result	A64
float32x2_t vfmlal_low_f16(float32x2_t r, float16x4_t a, float16x4_t b)	r -> Vd.2S	FMLAL Vd.2S,Vn.2H,Vm.2H	Vd.2S -> result	A32/A64
float32x2_t vfmlsl_low_f16(float32x2_t r, float16x4_t a, float16x4_t b)	r -> Vd.2S	FMLSL Vd.2S,Vn.2H,Vm.2H	Vd.2S -> result	A32/A64
float32x4_t vfmlalq_low_f16(float32x4_t r, float16x8_t a, float16x8_t b)	r -> Vd.4S	FMLAL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	A32/A64
float32x4_t vfmlslq_low_f16(float32x4_t r, float16x8_t a, float16x8_t b)	r -> Vd.4S	FMLSL Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	A32/A64
float32x2_t vfmlal_high_f16(float32x2_t r, float16x4_t a, float16x4_t b)	r -> Vd.2S	FMLAL2 Vd.2S,Vn.2H,Vm.2H	Vd.2S -> result	A32/A64
float32x2_t vfmlsl_high_f16(float32x2_t r, float16x4_t a, float16x4_t b)	r -> Vd.2S	FMLSL2 Vd.2S,Vn.2H,Vm.2H	Vd.2S -> result	A32/A64
float32x4_t vfmlalq_high_f16(float32x4_t r, float16x8_t a, float16x8_t b)	r -> Vd.4S	FMLAL2 Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	A32/A64
float32x4_t vfmlslq_high_f16(float32x4_t r, float16x8_t a, float16x8_t b)	r -> Vd.4S	FMLSL2 Vd.4S,Vn.4H,Vm.4H	Vd.4S -> result	A32/A64
float32x2_t vfmlal_lane_low_f16(float32x2_t r, float16x4_t a, float16x4_t b, const int lane)	r -> Vd.2S 0 <= lane <= 3	FMLAL Vd.2S,Vn.2H,Vm.H[lane]	Vd.2S -> result	A32/A64
float32x2_t vfmlal_laneq_low_f16(float32x2_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.2S 0 <= lane <= 7	FMLAL Vd.2S,Vn.2H,Vm.H[lane]	Vd.2S -> result	A32/A64
float32x4_t vfmlalq_lane_low_f16(float32x4_t r, float16x8_t a, float16x4_t b, const int lane)	r -> Vd.4S 0 <= lane <= 3	FMLAL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A32/A64
float32x4_t vfmlalq_laneq_low_f16(float32x4_t r, float16x8_t a, float16x8_t b, const int lane)	r -> Vd.4S 0 <= lane <= 7	FMLAL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A32/A64
float32x2_t vfmlsl_lane_low_f16(float32x2_t r, float16x4_t a, float16x4_t b, const int lane)	r -> Vd.2S 0 <= lane <= 3	FMLSL Vd.2S,Vn.2H,Vm.H[lane]	Vd.2S -> result	A32/A64
float32x2_t vfmlsl_laneq_low_f16(float32x2_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.2S 0 <= lane <= 7	FMLSL Vd.2S,Vn.2H,Vm.H[lane]	Vd.2S -> result	A32/A64
float32x4_t vfmlslq_lane_low_f16(float32x4_t r, float16x8_t a, float16x4_t b, const int lane)	r -> Vd.4S 0 <= lane <= 3	FMLSL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A32/A64
float32x4_t vfmlslq_laneq_low_f16(float32x4_t r, float16x8_t a, float16x8_t b, const int lane)	r -> Vd.4S 0 <= lane <= 7	FMLSL Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A32/A64
float32x2_t vfmlal_lane_high_f16(float32x2_t r, float16x4_t a, float16x4_t b, const int lane)	r -> Vd.2S 0 <= lane <= 3	FMLAL2 Vd.2S,Vn.2H,Vm.H[lane]	Vd.2S -> result	A32/A64
float32x2_t vfmlsl_lane_high_f16(float32x2_t r, float16x4_t a, float16x4_t b, const int lane)	r -> Vd.2S 0 <= lane <= 3	FMLSL2 Vd.2S,Vn.2H,Vm.H[lane]	Vd.2S -> result	A32/A64
float32x4_t vfmlalq_lane_high_f16(float32x4_t r, float16x8_t a, float16x4_t b, const int lane)	r -> Vd.4S 0 <= lane <= 3	FMLAL2 Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A32/A64
float32x4_t vfmlslq_lane_high_f16(float32x4_t r, float16x8_t a, float16x4_t b, const int lane)	r -> Vd.4S 0 <= lane <= 3	FMLSL2 Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x2_t vfmmlal_laneq_high_f16(float32x2_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.2S 0 <= lane <= 7	FMLAL2 Vd.2S,Vn.2H,Vm.H[lane]	Vd.2S -> result	A32/A64
float32x2_t vfmmlsl_laneq_high_f16(float32x2_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.2S 0 <= lane <= 7	FMLSL2 Vd.2S,Vn.2H,Vm.H[lane]	Vd.2S -> result	A32/A64
float32x4_t vfmmlal_laneq_high_f16(float32x4_t r, float16x8_t a, float16x8_t b, const int lane)	r -> Vd.4S 0 <= lane <= 7	FMLAL2 Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A32/A64
float32x4_t vfmmlsl_laneq_high_f16(float32x4_t r, float16x8_t a, float16x8_t b, const int lane)	r -> Vd.4S 0 <= lane <= 7	FMLSL2 Vd.4S,Vn.4H,Vm.H[lane]	Vd.4S -> result	A32/A64
float16x4_t vcadd_rot90_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FCADD Vd.4H,Vn.4H,Vm.4H,#90	Vd.4H -> result	A32/A64
float32x2_t vcadd_rot90_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FCADD Vd.2S,Vn.2S,Vm.2S,#90	Vd.2S -> result	A32/A64
float16x8_t vcadd_rot90_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FCADD Vd.8H,Vn.8H,Vm.8H,#90	Vd.8H -> result	A32/A64
float32x4_t vcadd_rot90_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FCADD Vd.4S,Vn.4S,Vm.4S,#90	Vd.4S -> result	A32/A64
float64x2_t vcadd_rot90_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FCADD Vd.2D,Vn.2D,Vm.2D,#90	Vd.2D -> result	A64
float16x4_t vcadd_rot270_f16(float16x4_t a, float16x4_t b)	a -> Vn.4H b -> Vm.4H	FCADD Vd.4H,Vn.4H,Vm.4H,#270	Vd.4H -> result	A32/A64
float32x2_t vcadd_rot270_f32(float32x2_t a, float32x2_t b)	a -> Vn.2S b -> Vm.2S	FCADD Vd.2S,Vn.2S,Vm.2S,#270	Vd.2S -> result	A32/A64
float16x8_t vcadd_rot270_f16(float16x8_t a, float16x8_t b)	a -> Vn.8H b -> Vm.8H	FCADD Vd.8H,Vn.8H,Vm.8H,#270	Vd.8H -> result	A32/A64
float32x4_t vcadd_rot270_f32(float32x4_t a, float32x4_t b)	a -> Vn.4S b -> Vm.4S	FCADD Vd.4S,Vn.4S,Vm.4S,#270	Vd.4S -> result	A32/A64
float64x2_t vcadd_rot270_f64(float64x2_t a, float64x2_t b)	a -> Vn.2D b -> Vm.2D	FCADD Vd.2D,Vn.2D,Vm.2D,#270	Vd.2D -> result	A64
float16x4_t vcmla_f16(float16x4_t r, float16x4_t a, float16x4_t b)	r -> Vd.4H	FCMLA Vd.4H,Vn.4H,Vm.4H,#0	Vd.4H -> result	A32/A64
float32x2_t vcmla_f32(float32x2_t r, float32x2_t a, float32x2_t b)	r -> Vd.2S	FCMLA Vd.2S,Vn.2S,Vm.2S,#0	Vd.2S -> result	A32/A64
float16x4_t vcmla_lane_f16(float16x4_t r, float16x4_t a, float16x4_t b, const int lane)	r -> Vd.4H 0 <= lane <= 1	FCMLA Vd.4H,Vn.4H,Vm.H[lane],#0	Vd.4H -> result	A32/A64
float32x2_t vcmla_lane_f32(float32x2_t r, float32x2_t a, float32x2_t b, const int lane)	r -> Vd.2S lane == 0	FCMLA Vd.2S,Vn.2S,Vm.2S,#0	Vd.2S -> result	A32/A64
float16x4_t vcmla_laneq_f16(float16x4_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.4H 0 <= lane <= 1	FCMLA Vd.4H,Vn.4H,Vm.H[lane],#0	Vd.4H -> result	A32/A64
float16x4_t vcmla_laneq_f16(float16x4_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.4H 2 <= lane <= 3	DUP Dm,Vm.D[1] FCMLA Vd.4H,Vn.4H,Vm.H[lane % 2],#0	Vd.4H -> result	A32/A64
float32x2_t vcmla_laneq_f32(float32x2_t r, float32x2_t a, float32x4_t b, const int lane)	r -> Vd.2S 0 <= lane <= 1	DUP Dm,Vm.D[1] FCMLA Vd.2S,Vn.2S,Vm.2S,#0	Vd.2S -> result	A32/A64
float16x8_t vcmla_laneq_f16(float16x8_t r, float16x8_t a, float16x8_t b)	r -> Vd.8H	FCMLA Vd.8H,Vn.8H,Vm.8H,#0	Vd.8H -> result	A32/A64
float32x4_t vcmla_laneq_f32(float32x4_t r, float32x4_t a, float32x4_t b)	r -> Vd.4S	FCMLA Vd.4S,Vn.4S,Vm.4S,#0	Vd.4S -> result	A32/A64
float64x2_t vcmla_laneq_f64(float64x2_t r, float64x2_t a, float64x2_t b)	r -> Vd.2D	FCMLA Vd.2D,Vn.2D,Vm.2D,#0	Vd.2D -> result	A64
float16x8_t vcmla_lane_f16(float16x8_t r, float16x8_t a, float16x4_t b, const int lane)	r -> Vd.8H 0 <= lane <= 1	FCMLA Vd.8H,Vn.8H,Vm.H[lane],#0	Vd.8H -> result	A32/A64
float32x4_t vcmla_lane_f32(float32x4_t r, float32x4_t a, float32x2_t b, const int lane)	r -> Vd.4S lane == 0	FCMLA Vd.4S,Vn.4S,Vm.S[lane],#0	Vd.4S -> result	A32/A64
float16x8_t vcmla_laneq_f16(float16x8_t r, float16x8_t a, float16x8_t b, const int lane)	r -> Vd.8H 0 <= lane <= 3	FCMLA Vd.8H,Vn.8H,Vm.H[lane],#0	Vd.8H -> result	A32/A64
float32x4_t vcmla_laneq_f32(float32x4_t r, float32x4_t a, float32x4_t b, const int lane)	r -> Vd.4S 0 <= lane <= 1	FCMLA Vd.4S,Vn.4S,Vm.S[lane],#0	Vd.4S -> result	A32/A64
float16x4_t vcmla_rot90_f16(float16x4_t r, float16x4_t a, float16x4_t b)	r -> Vd.4H	FCMLA Vd.4H,Vn.4H,Vm.4H,#90	Vd.4H -> result	A32/A64
float32x2_t vcmla_rot90_f32(float32x2_t r, float32x2_t a, float32x2_t b)	r -> Vd.2S	FCMLA Vd.2S,Vn.2S,Vm.2S,#90	Vd.2S -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x4_t vcmla_rot90_lane_f16(float16x4_t r, float16x4_t a, float16x4_t b, const int lane)	r -> Vd.4H 0 <= lane <= 1	FCMLA Vd.4H,Vn.4H,Vm.H[lane],#90	Vd.4H -> result	A32/A64
float32x2_t vcmla_rot90_lane_f32(float32x2_t r, float32x2_t a, float32x2_t b, const int lane)	r -> Vd.2S lane == 0	FCMLA Vd.2S,Vn.2S,Vm.2S,#90	Vd.2S -> result	A32/A64
float16x4_t vcmla_rot90_laneq_f16(float16x4_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.4H 0 <= lane <= 1	FCMLA Vd.4H,Vn.4H,Vm.H[lane],#90	Vd.4H -> result	A32/A64
float16x4_t vcmla_rot90_laneq_f16(float16x4_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.4H 2 <= lane <= 3	DUP Dm,Vm.D[1] FCMLA Vd.4H,Vn.4H,Vm.H[lane % 2],#90	Vd.4H -> result	A32/A64
float32x2_t vcmla_rot90_laneq_f32(float32x2_t r, float32x2_t a, float32x4_t b, const int lane)	r -> Vd.2S 0 <= lane <= 1	DUP Dm,Vm.D[1] FCMLA Vd.2S,Vn.2S,Vm.2S,#90	Vd.2S -> result	A32/A64
float16x8_t vcmlaq_rot90_f16(float16x8_t r, float16x8_t a, float16x8_t b)	r -> Vd.8H	FCMLA Vd.8H,Vn.8H,Vm.8H,#90	Vd.8H -> result	A32/A64
float32x4_t vcmlaq_rot90_f32(float32x4_t r, float32x4_t a, float32x4_t b)	r -> Vd.4S	FCMLA Vd.4S,Vn.4S,Vm.4S,#90	Vd.4S -> result	A32/A64
float64x2_t vcmlaq_rot90_f64(float64x2_t r, float64x2_t a, float64x2_t b)	r -> Vd.2D	FCMLA Vd.2D,Vn.2D,Vm.2D,#90	Vd.2D -> result	A64
float16x8_t vcmlaq_rot90_lane_f16(float16x8_t r, float16x8_t a, float16x4_t b, const int lane)	r -> Vd.8H 0 <= lane <= 1	FCMLA Vd.8H,Vn.8H,Vm.H[lane],#90	Vd.8H -> result	A32/A64
float32x4_t vcmlaq_rot90_lane_f32(float32x4_t r, float32x4_t a, float32x2_t b, const int lane)	r -> Vd.4S lane == 0	FCMLA Vd.4S,Vn.4S,Vm.S[lane],#90	Vd.4S -> result	A32/A64
float16x8_t vcmlaq_rot90_laneq_f16(float16x8_t r, float16x8_t a, float16x8_t b, const int lane)	r -> Vd.8H 0 <= lane <= 3	FCMLA Vd.8H,Vn.8H,Vm.H[lane],#90	Vd.8H -> result	A32/A64
float32x4_t vcmlaq_rot90_laneq_f32(float32x4_t r, float32x4_t a, float32x4_t b, const int lane)	r -> Vd.4S 0 <= lane <= 1	FCMLA Vd.4S,Vn.4S,Vm.S[lane],#90	Vd.4S -> result	A32/A64
float16x4_t vcmla_rot180_f16(float16x4_t r, float16x4_t a, float16x4_t b)	r -> Vd.4H	FCMLA Vd.4H,Vn.4H,Vm.4H,#180	Vd.4H -> result	A32/A64
float32x2_t vcmla_rot180_f32(float32x2_t r, float32x2_t a, float32x2_t b)	r -> Vd.2S	FCMLA Vd.2S,Vn.2S,Vm.2S,#180	Vd.2S -> result	A32/A64
float16x4_t vcmla_rot180_lane_f16(float16x4_t r, float16x4_t a, float16x4_t b, const int lane)	r -> Vd.4H 0 <= lane <= 1	FCMLA Vd.4H,Vn.4H,Vm.H[lane],#180	Vd.4H -> result	A32/A64
float32x2_t vcmla_rot180_lane_f32(float32x2_t r, float32x2_t a, float32x2_t b, const int lane)	r -> Vd.2S lane == 0	FCMLA Vd.2S,Vn.2S,Vm.2S,#180	Vd.2S -> result	A32/A64
float16x4_t vcmla_rot180_laneq_f16(float16x4_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.4H 0 <= lane <= 1	FCMLA Vd.4H,Vn.4H,Vm.H[lane],#180	Vd.4H -> result	A32/A64
float16x4_t vcmla_rot180_laneq_f16(float16x4_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.4H 2 <= lane <= 3	DUP Dm,Vm.D[1] FCMLA Vd.4H,Vn.4H,Vm.H[lane % 2],#180	Vd.4H -> result	A32/A64
float32x2_t vcmla_rot180_laneq_f32(float32x2_t r, float32x2_t a, float32x4_t b, const int lane)	r -> Vd.2S 0 <= lane <= 1	DUP Dm,Vm.D[1] FCMLA Vd.2S,Vn.2S,Vm.2S,#180	Vd.2S -> result	A32/A64
float16x8_t vcmlaq_rot180_f16(float16x8_t r, float16x8_t a, float16x8_t b)	r -> Vd.8H	FCMLA Vd.8H,Vn.8H,Vm.8H,#180	Vd.8H -> result	A32/A64
float32x4_t vcmlaq_rot180_f32(float32x4_t r, float32x4_t a, float32x4_t b)	r -> Vd.4S	FCMLA Vd.4S,Vn.4S,Vm.4S,#180	Vd.4S -> result	A32/A64
float64x2_t vcmlaq_rot180_f64(float64x2_t r, float64x2_t a, float64x2_t b)	r -> Vd.2D	FCMLA Vd.2D,Vn.2D,Vm.2D,#180	Vd.2D -> result	A64
float16x8_t vcmlaq_rot180_lane_f16(float16x8_t r, float16x8_t a, float16x4_t b, const int lane)	r -> Vd.8H 0 <= lane <= 1	FCMLA Vd.8H,Vn.8H,Vm.H[lane],#180	Vd.8H -> result	A32/A64
float32x4_t vcmlaq_rot180_lane_f32(float32x4_t r, float32x4_t a, float32x2_t b, const int lane)	r -> Vd.4S lane == 0	FCMLA Vd.4S,Vn.4S,Vm.S[lane],#180	Vd.4S -> result	A32/A64
float16x8_t vcmlaq_rot180_laneq_f16(float16x8_t r, float16x8_t a, float16x8_t b, const int lane)	r -> Vd.8H 0 <= lane <= 3	FCMLA Vd.8H,Vn.8H,Vm.H[lane],#180	Vd.8H -> result	A32/A64
float32x4_t vcmlaq_rot180_laneq_f32(float32x4_t r, float32x4_t a, float32x4_t b, const int lane)	r -> Vd.4S 0 <= lane <= 1	FCMLA Vd.4S,Vn.4S,Vm.S[lane],#180	Vd.4S -> result	A32/A64
float16x4_t vcmla_rot270_f16(float16x4_t r, float16x4_t a, float16x4_t b)	r -> Vd.4H	FCMLA Vd.4H,Vn.4H,Vm.4H,#270	Vd.4H -> result	A32/A64
float32x2_t vcmla_rot270_f32(float32x2_t r, float32x2_t a, float32x2_t b)	r -> Vd.2S	FCMLA Vd.2S,Vn.2S,Vm.2S,#270	Vd.2S -> result	A32/A64
float16x4_t vcmla_rot270_lane_f16(float16x4_t r, float16x4_t a, float16x4_t b, const int lane)	r -> Vd.4H 0 <= lane <= 1	FCMLA Vd.4H,Vn.4H,Vm.H[lane],#270	Vd.4H -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x2_t vcmla_rot270_lane_f32(float32x2_t r, float32x2_t a, float32x2_t b, const int lane)	r -> Vd.2S lane == 0	FCMLA Vd.2S,Vn.2S,Vm.2S,#270	Vd.2S -> result	A32/A64
float16x4_t vcmla_rot270_laneq_f16(float16x4_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.4H 0 <= lane <= 1	FCMLA Vd.4H,Vn.4H,Vm.H[lane],#270	Vd.4H -> result	A32/A64
float16x4_t vcmla_rot270_laneq_f16(float16x4_t r, float16x4_t a, float16x8_t b, const int lane)	r -> Vd.4H 2 <= lane <= 3	DUP Dm,Vm.D[1] FCMLA Vd.4H,Vn.4H,Vm.H[lane % 2],#270	Vd.4H -> result	A32/A64
float32x2_t vcmla_rot270_laneq_f32(float32x2_t r, float32x2_t a, float32x4_t b, const int lane)	r -> Vd.2S 0 <= lane <= 1	DUP Dm,Vm.D[1] FCMLA Vd.2S,Vn.2S,Vm.2S,#270	Vd.2S -> result	A32/A64
float16x8_t vcmlaq_rot270_f16(float16x8_t r, float16x8_t a, float16x8_t b)	r -> Vd.8H	FCMLA Vd.8H,Vn.8H,Vm.8H,#270	Vd.8H -> result	A32/A64
float32x4_t vcmlaq_rot270_f32(float32x4_t r, float32x4_t a, float32x4_t b)	r -> Vd.4S	FCMLA Vd.4S,Vn.4S,Vm.4S,#270	Vd.4S -> result	A32/A64
float64x2_t vcmlaq_rot270_f64(float64x2_t r, float64x2_t a, float64x2_t b)	r -> Vd.2D	FCMLA Vd.2D,Vn.2D,Vm.2D,#270	Vd.2D -> result	A64
float16x8_t vcmlaq_rot270_lane_f16(float16x8_t r, float16x8_t a, float16x4_t b, const int lane)	r -> Vd.8H 0 <= lane <= 1	FCMLA Vd.8H,Vn.8H,Vm.H[lane],#270	Vd.8H -> result	A32/A64
float32x4_t vcmlaq_rot270_lane_f32(float32x4_t r, float32x4_t a, float32x2_t b, const int lane)	r -> Vd.4S lane == 0	FCMLA Vd.4S,Vn.4S,Vm.S[lane],#270	Vd.4S -> result	A32/A64
float16x8_t vcmlaq_rot270_laneq_f16(float16x8_t r, float16x8_t a, float16x8_t b, const int lane)	r -> Vd.8H 0 <= lane <= 3	FCMLA Vd.8H,Vn.8H,Vm.H[lane],#270	Vd.8H -> result	A32/A64
float32x4_t vcmlaq_rot270_laneq_f32(float32x4_t r, float32x4_t a, float32x4_t b, const int lane)	r -> Vd.4S 0 <= lane <= 1	FCMLA Vd.4S,Vn.4S,Vm.S[lane],#270	Vd.4S -> result	A32/A64
float32x2_t vrnd32z_f32(float32x2_t a)	a -> Vn	FRINT32Z Vd.2S,Vn.2S	Vd -> result	A64
float32x4_t vrnd32zq_f32(float32x4_t a)	a -> Vn	FRINT32Z Vd.4S,Vn.4S	Vd -> result	A64
float64x1_t vrnd32z_f64(float64x1_t a)	a -> Dn	FRINT32Z Dd,Dn	Dd -> result	A64
float64x2_t vrnd32zq_f64(float64x2_t a)	a -> Vn	FRINT32Z Vd.2D,Vn.2D	Vd -> result	A64
float32x2_t vrnd64z_f32(float32x2_t a)	a -> Vn	FRINT64Z Vd.2S,Vn.2S	Vd -> result	A64
float32x4_t vrnd64zq_f32(float32x4_t a)	a -> Vn	FRINT64Z Vd.4S,Vn.4S	Vd -> result	A64
float64x1_t vrnd64z_f64(float64x1_t a)	a -> Dn	FRINT64Z Dd,Dn	Dd -> result	A64
float64x2_t vrnd64zq_f64(float64x2_t a)	a -> Vn	FRINT64Z Vd.2D,Vn.2D	Vd -> result	A64
float32x2_t vrnd32x_f32(float32x2_t a)	a -> Vn	FRINT32X Vd.2S,Vn.2S	Vd -> result	A64
float32x4_t vrnd32xq_f32(float32x4_t a)	a -> Vn	FRINT32X Vd.4S,Vn.4S	Vd -> result	A64
float64x1_t vrnd32x_f64(float64x1_t a)	a -> Dn	FRINT32X Dd,Dn	Dd -> result	A64
float64x2_t vrnd32xq_f64(float64x2_t a)	a -> Vn	FRINT32X Vd.2D,Vn.2D	Vd -> result	A64
float32x2_t vrnd64x_f32(float32x2_t a)	a -> Vn	FRINT64X Vd.2S,Vn.2S	Vd -> result	A64
float32x4_t vrnd64xq_f32(float32x4_t a)	a -> Vn	FRINT64X Vd.4S,Vn.4S	Vd -> result	A64
float64x1_t vrnd64x_f64(float64x1_t a)	a -> Dn	FRINT64X Dd,Dn	Dd -> result	A64
float64x2_t vrnd64xq_f64(float64x2_t a)	a -> Vn	FRINT64X Vd.2D,Vn.2D	Vd -> result	A64
int32x4_t vmmlaq_s32(int32x4_t r, int8x16_t a, int8x16_t b)	r -> Vd.4S a -> Vn.16B b -> Vm.16B	SMMLA Vd.4S,Vn.16B,Vm.16B	Vd.4S -> result	A32/A64
uint32x4_t vmmlaq_u32(uint32x4_t r, uint8x16_t a, uint8x16_t b)	r -> Vd.4S a -> Vn.16B b -> Vm.16B	UMMLA Vd.4S,Vn.16B,Vm.16B	Vd.4S -> result	A32/A64
int32x4_t vusmmlaq_s32(int32x4_t r, uint8x16_t a, int8x16_t b)	r -> Vd.4S a -> Vn.16B b -> Vm.16B	USMMLA Vd.4S,Vn.16B,Vm.16B	Vd.4S -> result	A32/A64
int32x2_t vusdot_s32(int32x2_t r, uint8x8_t a, int8x8_t b)	r -> Vd.2S a -> Vn.8B b -> Vm.8B	USDOT Vd.2S,Vn.8B,Vm.8B	Vd.2S -> result	A32/A64
int32x2_t vusdot_lane_s32(int32x2_t r, uint8x8_t a, int8x8_t b, const int lane)	r -> Vd.2S a -> Vn.8B b -> Vm.4B 0 <= lane <= 1	USDOT Vd.2S,Vn.8B,Vm.4B[lane]	Vd.2S -> result	A32/A64
int32x2_t vsudot_lane_s32(int32x2_t r, int8x8_t a, uint8x8_t b, const int lane)	r -> Vd.2S a -> Vn.8B b -> Vm.4B 0 <= lane <= 1	SUDOT Vd.2S,Vn.8B,Vm.4B[lane]	Vd.2S -> result	A32/A64
int32x2_t vusdot_laneq_s32(int32x2_t r, uint8x8_t a, int8x16_t b, const int lane)	r -> Vd.2S a -> Vn.8B b -> Vm.4B 0 <= lane <= 3	USDOT Vd.2S,Vn.8B,Vm.4B[lane]	Vd.2S -> result	A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x2_t vsudot_laneq_s32(int32x2_t r, int8x8_t a, uint8x16_t b, const int lane)	r -> Vd.2S a -> Vn.8B b -> Vm.4B 0 <= lane <= 3	SUDOT Vd.2S,Vn.8B,Vm.4B[lane]	Vd.2S -> result	A64
int32x4_t vusdotq_s32(int32x4_t r, uint8x16_t a, int8x16_t b)	r -> Vd.4S a -> Vn.16B b -> Vm.16B	USDOT Vd.4S,Vn.16B,Vm.16B	Vd.4S -> result	A64
int32x4_t vusdotq_lane_s32(int32x4_t r, uint8x16_t a, int8x8_t b, const int lane)	r -> Vd.4S a -> Vn.16B b -> Vm.4B 0 <= lane <= 1	USDOT Vd.4S,Vn.16B,Vm.4B[lane]	Vd.4S -> result	A32/A64
int32x4_t vsudotq_lane_s32(int32x4_t r, int8x16_t a, uint8x8_t b, const int lane)	r -> Vd.4S a -> Vn.8B b -> Vm.4B 0 <= lane <= 1	SUDOT Vd.4S,Vn.16B,Vm.4B[lane]	Vd.4S -> result	A32/A64
int32x4_t vusdotq_laneq_s32(int32x4_t r, uint8x16_t a, int8x16_t b, const int lane)	r -> Vd.4S a -> Vn.16B b -> Vm.4B 0 <= lane <= 3	USDOT Vd.4S,Vn.16B,Vm.4B[lane]	Vd.4S -> result	A64
int32x4_t vsudotq_laneq_s32(int32x4_t r, int8x16_t a, uint8x16_t b, const int lane)	r -> Vd.4S a -> Vn.8B b -> Vm.4B 0 <= lane <= 3	SUDOT Vd.4S,Vn.16B,Vm.4B[lane]	Vd.4S -> result	A64
bfloat16x4_t vcreate_bf16(uint64_t a)	a -> Xn	INS Vd.D[0],Xn	Vd.4H -> result	A32/A64
bfloat16x4_t vdup_n_bf16(bfloat16_t value)	value -> rn	DUP Vd.4H,rn	Vd.4H -> result	A32/A64
bfloat16x8_t vdupq_n_bf16(bfloat16_t value)	value -> rn	DUP Vd.8H,rn	Vd.8H -> result	A32/A64
bfloat16x4_t vdup_lane_bf16(bfloat16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Vd.4H,Vn.H[lane]	Vd.4H -> result	A32/A64
bfloat16x8_t vdupq_lane_bf16(bfloat16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Vd.8H,Vn.H[lane]	Vd.8H -> result	A32/A64
bfloat16x4_t vdup_laneq_bf16(bfloat16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Vd.4H,Vn.H[lane]	Vd.4H -> result	A32/A64
bfloat16x8_t vdupq_laneq_bf16(bfloat16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Vd.8H,Vn.H[lane]	Vd.8H -> result	A32/A64
bfloat16x8_t vcombine_bf16(bfloat16x4_t low, bfloat16x4_t high)	low -> Vn.4H high -> Vm.4H	DUP Vd.1D,Vn.D[0] INS Vd.D[1],Vm.D[0]	Vd.8H -> result	A32/A64
bfloat16x4_t vget_high_bf16(bfloat16x8_t a)	a -> Vn.8H	DUP Vd.1D,Vn.D[1]	Vd.4H -> result	A32/A64
bfloat16x4_t vget_low_bf16(bfloat16x8_t a)	a -> Vn.8H	DUP Vd.1D,Vn.D[0]	Vd.4H -> result	A32/A64
bfloat16_t vget_lane_bf16(bfloat16x4_t v, const int lane)	v -> Vn.4H 0 <= lane <= 3	DUP Hd,Vn.H[lane]	Hd -> result	A32/A64
bfloat16_t vgetq_lane_bf16(bfloat16x8_t v, const int lane)	v -> Vn.8H 0 <= lane <= 7	DUP Hd,Vn.H[lane]	Hd -> result	A32/A64
bfloat16x4_t vset_lane_bf16(bfloat16_t a, bfloat16x4_t v, const int lane)	a -> VnH v -> Vd.4H 0 <= lane <= 3	INS Vd.H[lane],Vn.H[0]	Vd.4H -> result	A32/A64
bfloat16x8_t vsetq_lane_bf16(bfloat16_t a, bfloat16x8_t v, const int lane)	a -> VnH v -> Vd.8H 0 <= lane <= 7	INS Vd.H[lane],Vn.H[0]	Vd.8H -> result	A32/A64
bfloat16_t vduph_lane_bf16(bfloat16x4_t vec, const int lane)	vec -> Vn.4H 0 <= lane <= 3	DUP Hd,Vn.H[lane]	Hd -> result	A32/A64
bfloat16_t vduph_laneq_bf16(bfloat16x8_t vec, const int lane)	vec -> Vn.8H 0 <= lane <= 7	DUP Hd,Vn.H[lane]	Hd -> result	A32/A64
bfloat16x4_t vld1_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H},[Xn]	Vt.4H -> result	A32/A64
bfloat16x8_t vld1q_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H},[Xn]	Vt.8H -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
bfloat16x4_t vld1_lane_bf16(bfloat16_t const * ptr, bfloat16x4_t src, const int lane)	ptr -> Xn src -> Vt.4H 0 <= lane <= 3	LD1 {Vt.H}[lane],[Xn]	Vt.4H -> result	A32/A64
bfloat16x8_t vld1q_lane_bf16(bfloat16_t const * ptr, bfloat16x8_t src, const int lane)	ptr -> Xn src -> Vt.8H 0 <= lane <= 7	LD1 {Vt.H}[lane],[Xn]	Vt.8H -> result	A32/A64
bfloat16x4_t vld1_dup_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD1R {Vt.4H},[Xn]	Vt.4H -> result	A32/A64
bfloat16x8_t vld1q_dup_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD1R {Vt.8H},[Xn]	Vt.8H -> result	A32/A64
void vst1_bf16(bfloat16_t * ptr, bfloat16x4_t val)	ptr -> Xn val -> Vt.4H	ST1 {Vt.4H},[Xn]	void -> result	A32/A64
void vst1q_bf16(bfloat16_t * ptr, bfloat16x8_t val)	ptr -> Xn val -> Vt.8H	ST1 {Vt.8H},[Xn]	void -> result	A32/A64
void vst1_lane_bf16(bfloat16_t * ptr, bfloat16x4_t val, const int lane)	ptr -> Xn val -> Vt.4H 0 <= lane <= 3	ST1 {Vt.h}[lane],[Xn]	void -> result	A32/A64
void vst1q_lane_bf16(bfloat16_t * ptr, bfloat16x8_t val, const int lane)	ptr -> Xn val -> Vt.8H 0 <= lane <= 7	ST1 {Vt.h}[lane],[Xn]	void -> result	A32/A64
bfloat16x4x2_t vld2_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD2 {Vt.4H - Vt.2.4H},[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	A32/A64
bfloat16x8x2_t vld2q_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD2 {Vt.8H - Vt.2.8H},[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	A32/A64
bfloat16x4x3_t vld3_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD3 {Vt.4H - Vt.3.4H},[Xn]	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	A32/A64
bfloat16x8x3_t vld3q_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD3 {Vt.8H - Vt.3.8H},[Xn]	Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	A32/A64
bfloat16x4x4_t vld4_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD4 {Vt.4H - Vt.4.4H},[Xn]	Vt.4.4H -> result.val[3] Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	A32/A64
bfloat16x8x4_t vld4q_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD4 {Vt.8H - Vt.4.8H},[Xn]	Vt.4.8H -> result.val[3] Vt.3.8H -> result.val[2] Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	A32/A64
bfloat16x4x2_t vld2_dup_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD2R {Vt.4H - Vt.2.4H},[Xn]	Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	A32/A64
bfloat16x8x2_t vld2q_dup_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD2R {Vt.8H - Vt.2.8H},[Xn]	Vt.2.8H -> result.val[1] Vt.8H -> result.val[0]	A32/A64
bfloat16x4x3_t vld3_dup_bf16(bfloat16_t const * ptr)	ptr -> Xn	LD3R {Vt.4H - Vt.3.4H},[Xn]	Vt.3.4H -> result.val[2] Vt.2.4H -> result.val[1] Vt.4H -> result.val[0]	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>bfloat16x8x3_t vld3q_dup_bf16(bfloat16_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD3R {Vt.8H - Vt.3.8H},[Xn]</code>	<code>Vt.3.8H -> result.val[2]</code> <code>Vt.2.8H -> result.val[1]</code> <code>Vt.8H -> result.val[0]</code>	A32/A64
<code>bfloat16x4x4_t vld4_dup_bf16(bfloat16_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD4R {Vt.4H - Vt.4.4H},[Xn]</code>	<code>Vt.4.4H -> result.val[3]</code> <code>Vt.3.4H -> result.val[2]</code> <code>Vt.2.4H -> result.val[1]</code> <code>Vt.4H -> result.val[0]</code>	A32/A64
<code>bfloat16x8x4_t vld4q_dup_bf16(bfloat16_t const * ptr)</code>	<code>ptr -> Xn</code>	<code>LD4R {Vt.8H - Vt.4.8H},[Xn]</code>	<code>Vt.4.8H -> result.val[3]</code> <code>Vt.3.8H -> result.val[2]</code> <code>Vt.2.8H -> result.val[1]</code> <code>Vt.8H -> result.val[0]</code>	A32/A64
<code>void vst2_bf16(bfloat16_t * ptr, bfloat16x2_t val)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.2.4H</code> <code>val.val[0] -> Vt.4H</code>	<code>ST2 {Vt.4H - Vt.2.4H},[Xn]</code>	<code>void -> result</code>	A32/A64
<code>void vst2q_bf16(bfloat16_t * ptr, bfloat16x8x2_t val)</code>	<code>ptr -> Xn</code> <code>val.val[1] -> Vt.2.8H</code> <code>val.val[0] -> Vt.8H</code>	<code>ST2 {Vt.8H - Vt.2.8H},[Xn]</code>	<code>void -> result</code>	A32/A64
<code>void vst3_bf16(bfloat16_t * ptr, bfloat16x4x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt.3.4H</code> <code>val.val[1] -> Vt.2.4H</code> <code>val.val[0] -> Vt.4H</code>	<code>ST3 {Vt.4H - Vt.3.4H},[Xn]</code>	<code>void -> result</code>	A32/A64
<code>void vst3q_bf16(bfloat16_t * ptr, bfloat16x8x3_t val)</code>	<code>ptr -> Xn</code> <code>val.val[2] -> Vt.3.8H</code> <code>val.val[1] -> Vt.2.8H</code> <code>val.val[0] -> Vt.8H</code>	<code>ST3 {Vt.8H - Vt.3.8H},[Xn]</code>	<code>void -> result</code>	A32/A64
<code>void vst4_bf16(bfloat16_t * ptr, bfloat16x4x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt.4.4H</code> <code>val.val[2] -> Vt.3.4H</code> <code>val.val[1] -> Vt.2.4H</code> <code>val.val[0] -> Vt.4H</code>	<code>ST4 {Vt.4H - Vt.4.4H},[Xn]</code>	<code>void -> result</code>	A32/A64
<code>void vst4q_bf16(bfloat16_t * ptr, bfloat16x8x4_t val)</code>	<code>ptr -> Xn</code> <code>val.val[3] -> Vt.4.8H</code> <code>val.val[2] -> Vt.3.8H</code> <code>val.val[1] -> Vt.2.8H</code> <code>val.val[0] -> Vt.8H</code>	<code>ST4 {Vt.8H - Vt.4.8H},[Xn]</code>	<code>void -> result</code>	A32/A64
<code>bfloat16x4x2_t vld2_lane_bf16(bfloat16_t const * ptr, bfloat16x4x2_t src, const int lane)</code>	<code>ptr -> Xn</code> <code>src.val[1] -> Vt.2.4H</code> <code>src.val[0] -> Vt.4H</code> <code>0 <= lane <= 3</code>	<code>LD2 {Vt.h - Vt.h},[lane],[Xn]</code>	<code>Vt.2.4H -> result.val[1]</code> <code>Vt.4H -> result.val[0]</code>	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>bfloat16x8x2_t vld2q_lane_bf16(bfloat16_t const * ptr, bfloat16x8x2_t src, const int lane)</code>	ptr -> Xn src.val[1] -> Vt2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD2 {Vt.h - Vt2.h}[lane], [Xn]	Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	A32/A64
<code>bfloat16x4x3_t vld3_lane_bf16(bfloat16_t const * ptr, bfloat16x4x3_t src, const int lane)</code>	ptr -> Xn src.val[2] -> Vt3.4H src.val[1] -> Vt2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD3 {Vt.h - Vt3.h}[lane], [Xn]	Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	A32/A64
<code>bfloat16x8x3_t vld3q_lane_bf16(bfloat16_t const * ptr, bfloat16x8x3_t src, const int lane)</code>	ptr -> Xn src.val[2] -> Vt3.8H src.val[1] -> Vt2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD3 {Vt.h - Vt3.h}[lane], [Xn]	Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	A32/A64
<code>bfloat16x4x4_t vld4_lane_bf16(bfloat16_t const * ptr, bfloat16x4x4_t src, const int lane)</code>	ptr -> Xn src.val[3] -> Vt4.4H src.val[2] -> Vt3.4H src.val[1] -> Vt2.4H src.val[0] -> Vt.4H 0 <= lane <= 3	LD4 {Vt.h - Vt4.h}[lane], [Xn]	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	A32/A64
<code>bfloat16x8x4_t vld4q_lane_bf16(bfloat16_t const * ptr, bfloat16x8x4_t src, const int lane)</code>	ptr -> Xn src.val[3] -> Vt4.8H src.val[2] -> Vt3.8H src.val[1] -> Vt2.8H src.val[0] -> Vt.8H 0 <= lane <= 7	LD4 {Vt.h - Vt4.h}[lane], [Xn]	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	A32/A64
<code>void vst2_lane_bf16(bfloat16_t * ptr, bfloat16x4x2_t val, const int lane)</code>	ptr -> Xn val.val[1] -> Vt2.4H val.val[0] -> Vt.4H 0 <= lane <= 3	ST2 {Vt.h - Vt2.h}[lane], [Xn]	void -> result	A32/A64
<code>void vst2q_lane_bf16(bfloat16_t * ptr, bfloat16x8x2_t val, const int lane)</code>	ptr -> Xn val.val[1] -> Vt2.8H val.val[0] -> Vt.8H 0 <= lane <= 7	ST2 {Vt.h - Vt2.h}[lane], [Xn]	void -> result	A32/A64
<code>void vst3_lane_bf16(bfloat16_t * ptr, bfloat16x4x3_t val, const int lane)</code>	ptr -> Xn val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H 0 <= lane <= 3	ST3 {Vt.h - Vt3.h}[lane], [Xn]	void -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void vst3q_lane_bf16(bfloat16_t * ptr, bfloat16x8x3_t val, const int lane)</code>	ptr -> Xn val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H 0 <= lane <= 7	ST3 {Vt.h - Vt3.h}[lane],[Xn]	void -> result	A32/A64
<code>void vst4_lane_bf16(bfloat16_t * ptr, bfloat16x4x4_t val, const int lane)</code>	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H 0 <= lane <= 3	ST4 {Vt.h - Vt4.h}[lane],[Xn]	void -> result	A32/A64
<code>void vst4q_lane_bf16(bfloat16_t * ptr, bfloat16x8x4_t val, const int lane)</code>	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H 0 <= lane <= 7	ST4 {Vt.h - Vt4.h}[lane],[Xn]	void -> result	A32/A64
<code>void vst1_bf16_x2(bfloat16_t * ptr, bfloat16x4x2_t val)</code>	ptr -> Xn val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST1 {Vt.4H - Vt2.4H],[Xn]	void -> result	A32/A64
<code>void vst1q_bf16_x2(bfloat16_t * ptr, bfloat16x8x2_t val)</code>	ptr -> Xn val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST1 {Vt.8H - Vt2.8H],[Xn]	void -> result	A32/A64
<code>void vst1_bf16_x3(bfloat16_t * ptr, bfloat16x4x3_t val)</code>	ptr -> Xn val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST1 {Vt.4H - Vt3.4H],[Xn]	void -> result	A32/A64
<code>void vst1q_bf16_x3(bfloat16_t * ptr, bfloat16x8x3_t val)</code>	ptr -> Xn val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST1 {Vt.8H - Vt3.8H],[Xn]	void -> result	A32/A64
<code>void vst1_bf16_x4(bfloat16_t * ptr, bfloat16x4x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.4H val.val[2] -> Vt3.4H val.val[1] -> Vt2.4H val.val[0] -> Vt.4H	ST1 {Vt.4H - Vt4.4H],[Xn]	void -> result	A32/A64
<code>void vst1q_bf16_x4(bfloat16_t * ptr, bfloat16x8x4_t val)</code>	ptr -> Xn val.val[3] -> Vt4.8H val.val[2] -> Vt3.8H val.val[1] -> Vt2.8H val.val[0] -> Vt.8H	ST1 {Vt.8H - Vt4.8H],[Xn]	void -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
bfloat16x4x2_t vld1_bf16_x2(bfloat16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt2.4H},{Xn}	Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	A32/A64
bfloat16x8x2_t vld1q_bf16_x2(bfloat16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt2.8H},{Xn}	Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	A32/A64
bfloat16x4x3_t vld1_bf16_x3(bfloat16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt3.4H},{Xn}	Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	v7/A32/A64
bfloat16x8x3_t vld1q_bf16_x3(bfloat16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt3.8H},{Xn}	Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	v7/A32/A64
bfloat16x4x4_t vld1_bf16_x4(bfloat16_t const * ptr)	ptr -> Xn	LD1 {Vt.4H - Vt4.4H},{Xn}	Vt4.4H -> result.val[3] Vt3.4H -> result.val[2] Vt2.4H -> result.val[1] Vt.4H -> result.val[0]	A32/A64
bfloat16x8x4_t vld1q_bf16_x4(bfloat16_t const * ptr)	ptr -> Xn	LD1 {Vt.8H - Vt4.8H},{Xn}	Vt4.8H -> result.val[3] Vt3.8H -> result.val[2] Vt2.8H -> result.val[1] Vt.8H -> result.val[0]	A32/A64
bfloat16x4_t vreinterpret_bf16_s8(int8x8 t a)	a -> Vd.8B	NOP	Vd.4H -> result	A32/A64
bfloat16x4_t vreinterpret_bf16_s16(int16x4 t a)	a -> Vd.4H	NOP	Vd.4H -> result	A32/A64
bfloat16x4_t vreinterpret_bf16_s32(int32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	A32/A64
bfloat16x4_t vreinterpret_bf16_f32(float32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	A32/A64
bfloat16x4_t vreinterpret_bf16_u8(uint8x8 t a)	a -> Vd.8B	NOP	Vd.4H -> result	A32/A64
bfloat16x4_t vreinterpret_bf16_u16(uint16x4 t a)	a -> Vd.4H	NOP	Vd.4H -> result	A32/A64
bfloat16x4_t vreinterpret_bf16_u32(uint32x2 t a)	a -> Vd.2S	NOP	Vd.4H -> result	A32/A64
bfloat16x4_t vreinterpret_bf16_p8(poly8x8 t a)	a -> Vd.8B	NOP	Vd.4H -> result	A32/A64
bfloat16x4_t vreinterpret_bf16_p16(poly16x4 t a)	a -> Vd.4H	NOP	Vd.4H -> result	A32/A64
bfloat16x4_t vreinterpret_bf16_u64(uint64x1 t a)	a -> Vd.1D	NOP	Vd.4H -> result	A32/A64
bfloat16x4_t vreinterpret_bf16_s64(int64x1 t a)	a -> Vd.1D	NOP	Vd.4H -> result	A32/A64
bfloat16x8_t vreinterpretq_bf16_s8(int8x16 t a)	a -> Vd.16B	NOP	Vd.8H -> result	A32/A64
bfloat16x8_t vreinterpretq_bf16_s16(int16x8 t a)	a -> Vd.8H	NOP	Vd.8H -> result	A32/A64
bfloat16x8_t vreinterpretq_bf16_s32(int32x4 t a)	a -> Vd.4S	NOP	Vd.8H -> result	A32/A64
bfloat16x8_t vreinterpretq_bf16_f32(float32x4 t a)	a -> Vd.4S	NOP	Vd.8H -> result	A32/A64
bfloat16x8_t vreinterpretq_bf16_u8(uint8x16 t a)	a -> Vd.16B	NOP	Vd.8H -> result	A32/A64
bfloat16x8_t vreinterpretq_bf16_u16(uint16x8 t a)	a -> Vd.8H	NOP	Vd.8H -> result	A32/A64
bfloat16x8_t vreinterpretq_bf16_u32(uint32x4 t a)	a -> Vd.4S	NOP	Vd.8H -> result	A32/A64
bfloat16x8_t vreinterpretq_bf16_p8(poly8x16 t a)	a -> Vd.16B	NOP	Vd.8H -> result	A32/A64
bfloat16x8_t vreinterpretq_bf16_p16(poly16x8 t a)	a -> Vd.8H	NOP	Vd.8H -> result	A32/A64
bfloat16x8_t vreinterpretq_bf16_u64(uint64x2 t a)	a -> Vd.2D	NOP	Vd.8H -> result	A32/A64
bfloat16x8_t vreinterpretq_bf16_s64(int64x2 t a)	a -> Vd.2D	NOP	Vd.8H -> result	A32/A64
bfloat16x4_t vreinterpret_bf16_f64(float64x1 t a)	a -> Vd.1D	NOP	Vd.4H -> result	A64
bfloat16x8_t vreinterpretq_bf16_f64(float64x2 t a)	a -> Vd.2D	NOP	Vd.8H -> result	A64
bfloat16x4_t vreinterpretpret_bf16_p64(poly64x1 t a)	a -> Vd.1D	NOP	Vd.4H -> result	A32/A64
bfloat16x8_t vreinterpretpretq_bf16_p64(poly64x2 t a)	a -> Vd.2D	NOP	Vd.8H -> result	A32/A64
bfloat16x8_t vreinterpretpretq_bf16_p128(poly128 t a)	a -> Vd.1Q	NOP	Vd.8H -> result	A32/A64
int8x8_t vreinterpret_s8_bf16(bfloat16x4 t a)	a -> Vd.4H	NOP	Vd.8B -> result	A32/A64
int16x4_t vreinterpretpret_s16_bf16(bfloat16x4 t a)	a -> Vd.4H	NOP	Vd.4H -> result	A32/A64
int32x2_t vreinterpretpret_s32_bf16(bfloat16x4 t a)	a -> Vd.4H	NOP	Vd.2S -> result	A32/A64
float32x2_t vreinterpretpret_f32_bf16(bfloat16x4 t a)	a -> Vd.4H	NOP	Vd.2S -> result	A32/A64
uint8x8_t vreinterpret_u8_bf16(bfloat16x4 t a)	a -> Vd.4H	NOP	Vd.8B -> result	A32/A64
uint16x4_t vreinterpretpret_u16_bf16(bfloat16x4 t a)	a -> Vd.4H	NOP	Vd.4H -> result	A32/A64
uint32x2_t vreinterpretpret_u32_bf16(bfloat16x4 t a)	a -> Vd.4H	NOP	Vd.2S -> result	A32/A64
poly8x8_t vreinterpretpret_p8_bf16(bfloat16x4 t a)	a -> Vd.4H	NOP	Vd.8B -> result	A32/A64
poly16x4_t vreinterpretpret_p16_bf16(bfloat16x4 t a)	a -> Vd.4H	NOP	Vd.4H -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64x1_t vreinterpret_u64_bf16(bfloat16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	A32/A64
int64x1_t vreinterpret_s64_bf16(bfloat16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	A32/A64
float64x1_t vreinterpret_f64_bf16(bfloat16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	A64
poly64x1_t vreinterpret_p64_bf16(bfloat16x4_t a)	a -> Vd.4H	NOP	Vd.1D -> result	A32/A64
int8x16_t vreinterpretq_s8_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	A32/A64
int16x8_t vreinterpretq_s16_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	A32/A64
int32x4_t vreinterpretq_s32_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	A32/A64
float32x4_t vreinterpretq_f32_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	A32/A64
uint8x16_t vreinterpretq_u8_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	A32/A64
uint16x8_t vreinterpretq_u16_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	A32/A64
uint32x4_t vreinterpretq_u32_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.4S -> result	A32/A64
poly8x16_t vreinterpretq_p8_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.16B -> result	A32/A64
poly16x8_t vreinterpretq_p16_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.8H -> result	A32/A64
uint64x2_t vreinterpretq_u64_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A32/A64
int64x2_t vreinterpretq_s64_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A32/A64
float64x2_t vreinterpretq_f64_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A64
poly64x2_t vreinterpretq_p64_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.2D -> result	A32/A64
poly128_t vreinterpretq_p128_bf16(bfloat16x8_t a)	a -> Vd.8H	NOP	Vd.1Q -> result	A32/A64
float32x4_t vcvt_f32_bf16(bfloat16x4_t a)	a -> Vn.8H	SHLL Vd.4S,Vn.8H,#16	Vd.4S -> result	A32/A64
float32x4_t vcvtq_low_f32_bf16(bfloat16x8_t a)	a -> Vn.8H	SHLL Vd.4S,Vn.8H,#16	Vd.4S -> result	A32/A64
float32x4_t vcvtq_high_f32_bf16(bfloat16x8_t a)	a -> Vn.8H	SHLL2 Vd.4S,Vn.8H,#16	Vd.4S -> result	A32/A64
bfloat16x4_t vcvt_bf16_f32(float32x4_t a)	a -> Vn.4S	BFCVTN Vd.4H,Vn.4S	Vd.4H -> result	A32/A64
bfloat16x8_t vcvtq_low_bf16_f32(float32x4_t a)	a -> Vn.4S	BFCVTN Vd.4H,Vn.4S	Vd.8H -> result	A32/A64
bfloat16x8_t vcvtq_high_bf16_f32(bfloat16x8_t inactive, float32x4_t a)	inactive -> Vd.8H a -> Vn.4S	BFCVTN2 Vd.8H,Vn.4S	Vd.8H -> result	A32/A64
bfloat16_t vcvt_bf16_f32(float32_t a)	a -> Sn	BFCVT Hd,Sn	Hd -> result	A32/A64
float32_t vcvtah_f32_bf16(bfloat16_t a)	a -> Hn	SHL Dd,Dn,#16	Sd -> result	A32/A64
bfloat16x4_t vcopy_lane_bf16(bfloat16x4_t a, const int lane1, bfloat16x4_t b, const int lane2)	a -> Vd.4H 0 <= lane1 <= 3 b -> Vn.4H 0 <= lane2 <= 3	INS Vd.H[lane1],Vn.H[lane2]	Vd.4H -> result	A64
bfloat16x8_t vcopyq_lane_bf16(bfloat16x8_t a, const int lane1, bfloat16x4_t b, const int lane2)	a -> Vd.8H 0 <= lane1 <= 7 b -> Vn.4H 0 <= lane2 <= 3	INS Vd.H[lane1],Vn.H[lane2]	Vd.8H -> result	A64
bfloat16x4_t vcopy_laneq_bf16(bfloat16x4_t a, const int lane1, bfloat16x8_t b, const int lane2)	a -> Vd.4H 0 <= lane1 <= 3 b -> Vn.8H 0 <= lane2 <= 7	INS Vd.H[lane1],Vn.H[lane2]	Vd.4H -> result	A64
bfloat16x8_t vcopyq_laneq_bf16(bfloat16x8_t a, const int lane1, bfloat16x8_t b, const int lane2)	a -> Vd.8H 0 <= lane1 <= 7 b -> Vn.8H 0 <= lane2 <= 7	INS Vd.H[lane1],Vn.H[lane2]	Vd.8H -> result	A64
float32x2_t vbfdot_f32(float32x2_t r, bfloat16x4_t a, bfloat16x4_t b)	r -> Vd.2S a -> Vn.4H b -> Vm.4H	BFDOT Vd.2S,Vn.4H,Vm.4H	Vd.2S -> result	A32/A64
float32x4_t vbfdotq_f32(float32x4_t r, bfloat16x8_t a, bfloat16x8_t b)	r -> Vd.4S a -> Vn.8H b -> Vm.8H	BFDOT Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A32/A64
float32x2_t vbfdot_lane_f32(float32x2_t r, bfloat16x4_t a, bfloat16x4_t b, const int lane)	r -> Vd.2S a -> Vn.4H b -> Vm.4H 0 <= lane <= 1	BFDOT Vd.2S,Vn.4H,Vm.2H[lane]	Vd.2S -> result	A32/A64
float32x4_t vbfdotq_laneq_f32(float32x4_t r, bfloat16x8_t a, bfloat16x8_t b, const int lane)	r -> Vd.4S a -> Vn.8H b -> Vm.8H 0 <= lane <= 3	BFDOT Vd.4S,Vn.8H,Vm.2H[lane]	Vd.4S -> result	A32/A64

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x2_t vbfdot_laneq_f32(float32x2_t r, bfloat16x4_t a, bfloat16x8_t b, const int lane)	r -> Vd.2S a -> Vn.4H b -> Vm.8H 0 <= lane <= 3	BFDOT Vd.2S,Vn.4H,Vm.2H[lane]	Vd.2S -> result	A32/A64
float32x4_t vbfdotq_lane_f32(float32x4_t r, bfloat16x8_t a, bfloat16x4_t b, const int lane)	r -> Vd.4S a -> Vn.8H b -> Vm.4H 0 <= lane <= 1	BFDOT Vd.4S,Vn.8H,Vm.2H[lane]	Vd.4S -> result	A32/A64
float32x4_t vbfmmlaq_f32(float32x4_t r, bfloat16x8_t a, bfloat16x8_t b)	r -> Vd.4S a -> Vn.8H b -> Vm.8H	BFMMLA Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A32/A64
float32x4_t vbfmlalbq_f32(float32x4_t r, bfloat16x8_t a, bfloat16x8_t b)	r -> Vd.4S a -> Vn.8H b -> Vm.8H	BFMLALB Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A32/A64
float32x4_t vbfmlaltq_f32(float32x4_t r, bfloat16x8_t a, bfloat16x8_t b)	r -> Vd.4S a -> Vn.8H b -> Vm.8H	BFMLALT Vd.4S,Vn.8H,Vm.8H	Vd.4S -> result	A32/A64
float32x4_t vbfmlalbq_lane_f32(float32x4_t r, bfloat16x8_t a, bfloat16x4_t b, const int lane)	r -> Vd.4S a -> Vn.8H b -> Vm.4H 0 <= lane <= 3	BFMLALB Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A32/A64
float32x4_t vbfmlalbq_laneq_f32(float32x4_t r, bfloat16x8_t a, bfloat16x8_t b, const int lane)	r -> Vd.4S a -> Vn.8H b -> Vm.8H 0 <= lane <= 7	BFMLALB Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A32/A64
float32x4_t vbfmlaltq_lane_f32(float32x4_t r, bfloat16x8_t a, bfloat16x4_t b, const int lane)	r -> Vd.4S a -> Vn.8H b -> Vm.4H 0 <= lane <= 3	BFMLALT Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A32/A64
float32x4_t vbfmlaltq_laneq_f32(float32x4_t r, bfloat16x8_t a, bfloat16x8_t b, const int lane)	r -> Vd.4S a -> Vn.8H b -> Vm.8H 0 <= lane <= 7	BFMLALT Vd.4S,Vn.8H,Vm.H[lane]	Vd.4S -> result	A32/A64