

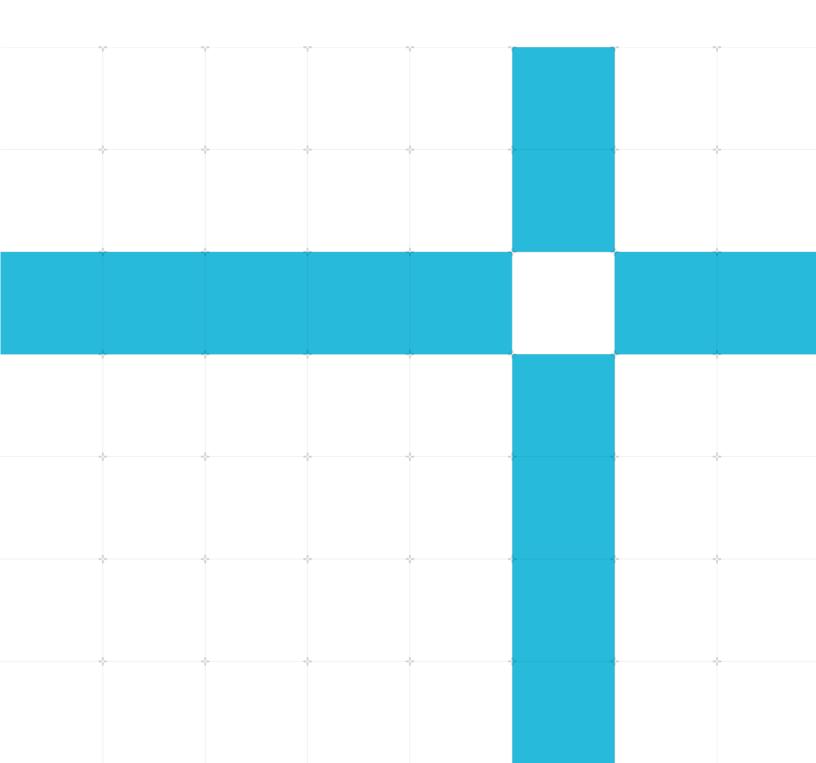
# Arm MVE Intrinsics Reference for ACLE Q4 2019

**Non-Confidential** 

Copyright  $\ \odot$  2020 Arm Limited (or its affiliates). All rights reserved.

Issue Q419-00

101809



Arm MVE Intrinsics Reference 101809

# **Arm MVE Intrinsics**

## Reference

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

## **Release information**

#### **Document history**

Issue	Date	Confidentiality	Change
Q219-00	30 June 2019	Non-Confidential	Version ACLE Q2 2019.
Q319-00	30 September 2019	Non-Confidential	Version ACLE Q3 2019
Q419-00	31 December 2019	Non-Confidential	Version ACLE Q4 2019

# **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.

Arm MVE Intrinsics Reference 101809

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 © 2019 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
int8x16_t [arm_]vrshlq_m_n[_s8](int8x16_t a, int32_t	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VRSHLT.S8 Qda,Rm		
float16x8_t [_arm_]vcreateq_f16(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
float32x4_t [arm_]vcreateq_f32(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
int8x16_t [arm_]vcreateq_s8(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
int16x8_t [arm_]vcreateq_s16(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
int32x4_t [arm_]vcreateq_s32(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
int64x2_t [arm_]vcreateq_s64(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint8x16_t [arm_]vcreateq_u8(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint16x8_t [arm_]vcreateq_u16(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint32x4_t [_arm_]vcreateq_u32(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint64x2_t [arm_]vcreateq_u64(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint8x16_t [arm_]vddupq[_n]_u8(uint32_t a, const int imm)	a -> Rn imm in [1,2,4,8]	VDDUP.U8 Qd,Rn,imm	Qd -> result	MVE
uint16x8_t [arm_]vddupq[_n]_u16(uint32_t a, const int imm)	a -> Rn imm in [1,2,4,8]	VDDUP.U16 Qd,Rn,imm	Qd -> result	MVE
uint32x4_t [arm_]vddupq[_n]_u32(uint32_t a, const int imm)	a -> Rn imm in [1,2,4,8]	VDDUP.U32 Qd,Rn,imm	Qd -> result	MVE
uint8x16_t [arm_]vddupq[_wb]_u8(uint32_t * a, const int imm)	*a -> Rn imm in [1,2,4,8]	VDDUP.U8 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [arm_]vddupq[_wb]_u16(uint32_t * a, const int imm)	*a -> Rn imm in [1,2,4,8]	VDDUP.U16 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [arm_]vddupq[_wb]_u32(uint32_t * a, const int imm)	*a -> Rn imm in [1,2,4,8]	VDDUP.U32 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [_arm_]vddupq_m[_n_u8](uint8x16_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U8 Qd,Rn,imm	Qd -> result	MVE
uint16x8_t [arm_]vddupq_m[_n_u16](uint16x8_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U16 Qd,Rn,imm	Qd -> result	MVE
uint32x4_t [_arm_]vddupq_m[_n_u32](uint32x4_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U32 Qd,Rn,imm	Qd -> result	MVE
uint8x16_t [arm_]vddupq_m[_wb_u8](uint8x16_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U8 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [_arm_]vddupq_m[_wb_u16](uint16x8_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U16 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]vddupq_m[_wb_u32](uint32x4_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U32 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [_arm_]vddupq_x[_n]_u8(uint32_t a, const int imm, mve_pred16_t p)	a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U8 Qd,Rn,imm	Qd -> result	MVE
uint16x8_t [arm_]vddupq_x[_n]_u16(uint32_t a, const int imm, mve_pred16_t p)	a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U16 Qd,Rn,imm	Qd -> result	MVE
uint32x4_t [arm_]vddupq_x[_n]_u32(uint32_t a, const int imm, mve_pred16_t p)	a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U32 Qd,Rn,imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [arm_]vddupq_x[_wb]_u8(uint32_t * a,	*a -> Rn	VMSR P0,Rp	Qd -> result	MVE
const int imm, mve_pred16_t p)	imm in [1,2,4,8] p -> Rp	VPST VDDUPT.U8 Qd,Rn,imm	Rn -> *a	
uint16x8_t [arm_]vddupq_x[_wb]_u16(uint32_t * a,	*a -> Rn	VMSR P0,Rp	Qd -> result	MVE
const int imm, mve_pred16_t p)	imm in [1,2,4,8]	VPST	Rn -> *a	
	p -> Rp	VDDUPT.U16 Qd,Rn,imm	0.1	NOW
uint32x4_t [arm_]vddupq_x[_wb]_u32(uint32_t * a, const int imm, mve_pred16_t p)	*a -> Rn imm in [1,2,4,8]	VMSR P0,Rp VPST	Qd -> result Rn -> *a	MVE
const int mini, inve_pred10_t p)	p -> Rp	VDDUPT.U32 Qd,Rn,imm	Kii -> a	
uint8x16_t [arm_]vdwdupq[_n]_u8(uint32_t a, uint32_t	a -> Rn	VDWDUP.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
b, const int imm)	b -> Rm			
	imm in [1,2,4,8]			
uint16x8_t [arm_]vdwdupq[_n]_u16(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm	VDWDUP.U16 Qd,Rn,Rm,imm	Qd -> result	MVE
umt32_t b, const int imin)	imm in [1,2,4,8]			
uint32x4_t [arm_]vdwdupq[_n]_u32(uint32_t a,	a -> Rn	VDWDUP.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32_t b, const int imm)	b -> Rm			
	imm in [1,2,4,8]			
uint8x16_t [arm_]vdwdupq[_wb]_u8(uint32_t * a,	*a -> Rn	VDWDUP.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32_t b, const int imm)	b -> Rm		Rn -> *a	
uint16x8 t[ arm ]vdwdupq[ wb] u16(uint32 t*a,	imm in [1,2,4,8] *a -> Rn	VDWDUP.U16 Qd,Rn,Rm,imm	Od -> result	MVE
uint32_t b, const int imm)	b -> Rm	VDWD01:010 Qu,Kii,Kiii,iiiiiii	Rn -> *a	WIVE
	imm in [1,2,4,8]			
uint32x4_t [arm_]vdwdupq[_wb]_u32(uint32_t * a,	*a -> Rn	VDWDUP.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32_t b, const int imm)	b -> Rm		Rn -> *a	
1.0.16.1	imm in [1,2,4,8]	VII (CD DO D	0.1	) am
uint8x16_t [arm_]vdwdupq_m[_n_u8](uint8x16_t inactive, uint32_t a, uint32_t b, const int imm,	inactive -> Qd a -> Rn	VMSR P0,Rp VPST	Qd -> result	MVE
mve_pred16_t p)	b -> Rm	VDWDUPT.U8 Qd,Rn,Rm,imm		
mve_prearo_t p)	imm in [1,2,4,8]	VD WDCI 1.00 Qu,kii,kiii,iiiiii		
	p -> Rp			
uint16x8_t [arm_]vdwdupq_m[_n_u16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32_t a, uint32_t b, const int imm,	a -> Rn	VPST		
mve_pred16_t p)	b -> Rm imm in [1,2,4,8]	VDWDUPT.U16 Qd,Rn,Rm,imm		
	p -> Rp			
uint32x4_t [arm_]vdwdupq_m[_n_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32_t a, uint32_t b, const int imm,	a -> Rn	VPST		
mve_pred16_t p)	b -> Rm	VDWDUPT.U32 Qd,Rn,Rm,imm		
	imm in [1,2,4,8]			
uint8x16_t [arm_]vdwdupq_m[_wb_u8](uint8x16_t	p -> Rp inactive -> Od	VMSR P0,Rp	Od -> result	MVE
inactive, uint32_t * a, uint32_t b, const int imm,	*a -> Rn	VPST	Rn -> *a	111.12
mve_pred16_t p)	b -> Rm	VDWDUPT.U8 Qd,Rn,Rm,imm		
	imm in [1,2,4,8]			
: 46.0 (	p -> Rp	VAMOR DO R	0.1	MATE
uint16x8_t [arm_]vdwdupq_m[_wb_u16](uint16x8_t inactive, uint32_t * a, uint32_t b, const int imm,	inactive -> Qd *a -> Rn	VMSR P0,Rp VPST	Qd -> result Rn -> *a	MVE
mve_pred16_t p)	b -> Rm	VDWDUPT.U16 Od,Rn,Rm,imm	Kii -> a	
	imm in [1,2,4,8]			
	p -> Rp			
uint32x4_t [arm_]vdwdupq_m[_wb_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32_t * a, uint32_t b, const int imm,	*a -> Rn b -> Rm	VPST VDWDUPT.U32 Qd,Rn,Rm,imm	Rn -> *a	
mve_pred16_t p)	imm in [1,2,4,8]	VDWDOF1.032 Qu,Kii,Kiii,Iiiiiii		
	p -> Rp			
uint8x16_t [arm_]vdwdupq_x[_n]_u8(uint32_t a,	a -> Rn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, const int imm, mve_pred16_t p)	b -> Rm	VPST		
	imm in [1,2,4,8]	VDWDUPT.U8 Qd,Rn,Rm,imm		
uint16x8_t [arm_]vdwdupq_x[_n]_u16(uint32_t a,	p -> Rp a -> Rn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, const int imm, mve_pred16_t p)	b -> Rm	VPST	Zu > icsuit	
_ · · · · · · · · · · · · · · · · · · ·	imm in [1,2,4,8]	VDWDUPT.U16 Qd,Rn,Rm,imm		
	p -> Rp			
uint32x4_t [_arm_]vdwdupq_x[_n]_u32(uint32_t a,	a -> Rn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, const int imm, mve_pred16_t p)	b -> Rm imm in [1,2,4,8]	VPST VDWDUPT.U32 Qd,Rn,Rm,imm		
	p -> Rp	2 2 c. 1. c. 2 Qu,Kii,Kiii,Iiiiii		
uint8x16_t [arm_]vdwdupq_x[_wb]_u8(uint32_t * a,	*a -> Rn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, const int imm, mve_pred16_t p)	b -> Rm	VPST	Rn -> *a	
	imm in [1,2,4,8]	VDWDUPT.U8 Qd,Rn,Rm,imm		
nint16v9 + f arm hydrodona of orb3 -16/-1-22 (*	p -> Rp	VMCD DO D-	Od > =====14	MVE
uint16x8_t [arm_]vdwdupq_x[_wb]_u16(uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	*a -> Rn b -> Rm	VMSR P0,Rp VPST	Qd -> result Rn -> *a	MVE
ame_t o, const me min, mvc_preuto_t p)			1.11 / U	1
	imm in [1,2,4,8]	VDWDUPT.U16 Qd,Rn,Rm,imm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vdwdupq_x[_wb]_u32(uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	*a -> Rn b -> Rm imm in [1,2,4,8]	VMSR P0,Rp VPST VDWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [arm_]vidupq[_n]_u8(uint32_t a, const int	p -> Rp a -> Rn	VIDUP.U8 Qd,Rn,imm	Qd -> result	MVE
imm) uint16x8_t [_arm_]vidupq[_n]_u16(uint32_t a, const int	imm in [1,2,4,8] a -> Rn	VIDUP.U16 Qd,Rn,imm	Qd -> result	MVE
imm) uint32x4_t [_arm_]vidupq[_n]_u32(uint32_t a, const int imm)	imm in [1,2,4,8] a -> Rn imm in [1,2,4,8]	VIDUP.U32 Qd,Rn,imm	Qd -> result	MVE
uint8x16_t [_arm_]vidupq[_wb]_u8(uint32_t * a, const int imm)	*a -> Rn imm in [1,2,4,8]	VIDUP.U8 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [_arm_]vidupq[_wb]_u16(uint32_t * a, const int imm)	*a -> Rn imm in [1,2,4,8]	VIDUP.U16 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [arm_]vidupq[_wb]_u32(uint32_t * a, const int imm)	*a -> Rn imm in [1,2,4,8]	VIDUP.U32 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [arm_]vidupq_m[_n_u8](uint8x16_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8]	VMSR P0,Rp VPST VIDUPT.U8 Qd,Rn,imm	Qd -> result	MVE
uint16x8_t [arm_]vidupq_m[_n_u16](uint16x8_t inactive, uint32_t a, const int imm, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U16 Qd,Rn,imm	Qd -> result	MVE
uint32x4_t [_arm_]vidupq_m[_n_u32](uint32x4_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U32 Qd,Rn,imm	Qd -> result	MVE
uint8x16_t [arm_]vidupq_m[_wb_u8](uint8x16_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U8 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [_arm_]vidupq_m[_wb_u16](uint16x8_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U16 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]vidupq_m[_wb_u32](uint32x4_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U32 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [arm_]vidupq_x[_n]_u8(uint32_t a, const int imm, mve_pred16_t p)	a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U8 Qd,Rn,imm	Qd -> result	MVE
uint16x8_t [arm_]vidupq_x[_n]_u16(uint32_t a, const int imm, mve_pred16_t p)	a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U16 Qd,Rn,imm	Qd -> result	MVE
uint32x4_t [arm_]vidupq_x[_n]_u32(uint32_t a, const int imm, mve_pred16_t p)	a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U32 Qd,Rn,imm	Qd -> result	MVE
uint8x16_t [_arm_]vidupq_x[_wb]_u8(uint32_t * a, const int imm, mve_pred16_t p)	*a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U8 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [_arm_]vidupq_x[_wb]_u16(uint32_t * a, const int imm, mve_pred16_t p)	*a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U16 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [arm_]vidupq_x[_wb]_u32(uint32_t * a, const int imm, mve_pred16_t p)	*a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U32 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [_arm_]viwdupq[_n]_u8(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm imm in [1,2,4,8]	VIWDUP.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
uint16x8_t [_arm_]viwdupq[_n]_u16(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm imm in [1,2,4,8]	VIWDUP.U16 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32x4_t [_arm_]viwdupq[_n]_u32(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm imm in [1,2,4,8]	VIWDUP.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint8x16_t [_arm_]viwdupq[_wb]_u8(uint32_t * a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VIWDUP.U8 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [arm_]viwdupq[_wb]_u16(uint32_t * a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VIWDUP.U16 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]viwdupq[_wb]_u32(uint32_t * a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VIWDUP.U32 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [_arm_]viwdupq_m[_n_u8](uint8x16_t inactive, uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
uint16x8_t [_arm_]viwdupq_m[_n_u16](uint16x8_t inactive, uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32x4_t [_arm_]viwdupq_m[_n_u32](uint32x4_t inactive, uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint8x16_t [_arm_]viwdupq_m[_wb_u8](uint8x16_t inactive, uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U8 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [_arm_]viwdupq_m[_wb_u16](uint16x8_t inactive, uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]viwdupq_m[_wb_u32](uint32x4_t inactive, uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [arm_]viwdupq_x[_n]_u8(uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
uint16x8_t [_arm_]viwdupq_x[_n]_u16(uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32x4_t [_arm_]viwdupq_x[_n]_u32(uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint8x16_t [_arm_]viwdupq_x[_wb]_u8(uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	*a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U8 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [_arm_]viwdupq_x[_wb]_u16(uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	*a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]viwdupq_x[_wb]_u32(uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	*a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
int8x16_t [arm_]vdupq_n_s8(int8_t a)	a -> Rt	VDUP.8 Qd,Rt	Qd -> result	MVE/NEON
int16x8_t [_arm_]vdupq_n_s16(int16_t a)	a -> Rt	VDUP.16 Qd,Rt	Qd -> result	MVE/NEON
int32x4_t [arm_]vdupq_n_s32(int32_t a) uint8x16_t [arm_]vdupq_n_u8(uint8_t a)	a -> Rt a -> Rt	VDUP.32 Qd,Rt VDUP.8 Qd,Rt	Qd -> result Qd -> result	MVE/NEON MVE/NEON
uint16x8_t [arm_]vdupq_n_u16(uint16_t a)	a -> Rt	VDUP.16 Qd,Rt	Qd -> result	MVE/NEON MVE/NEON
uint32x4_t [arm_]vdupq_n_u32(uint32_t a)	a -> Rt	VDUP.32 Qd,Rt	Qd -> result	MVE/NEON
float16x8_t [arm_]vdupq_n_f16(float16_t a)	a -> Rt	VDUP.16 Qd,Rt	Qd -> result	MVE/NEON
float32x4_t [arm_]vdupq_n_f32(float32_t a)	a -> Rt	VDUP.32 Qd,Rt	Qd -> result	MVE/NEON
int8x16_t [arm_]vdupq_m[_n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.8 Qd,Rt	Qd -> result	MVE
int16x8_t [arm_]vdupq_m[_n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.16 Qd,Rt	Qd -> result	MVE
int32x4_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.32 Qd,Rt	Qd -> result	MVE
uint8x16_t [arm_]vdupq_m[_n_u8](uint8x16_t inactive, uint8_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.8 Qd,Rt	Qd -> result	MVE
uint16x8_t [arm_]vdupq_m[_n_u16](uint16x8_t inactive, uint16_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.16 Qd,Rt	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [arm_]vdupq_m[_n_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32_t a, mve_pred16_t p)	a -> Rt	VPST		
float16x8_t [arm_]vdupq_m[_n_f16](float16x8_t	p -> Rp inactive -> Qd	VDUPT.32 Qd,Rt VMSR P0,Rp	Od -> result	MVE
inactive, float16_t a, mve_pred16_t p)	a -> Rt	VPST	\(\frac{1}{2} \cdot \cdo	
float32x4_t [arm_]vdupq_m[_n_f32](float32x4_t	p -> Rp inactive -> Qd	VDUPT.16 Qd,Rt VMSR P0,Rp	Od -> result	MVE
inactive, float32_t a, mve_pred16_t p)	a -> Rt	VMSR PU,RP VPST	Qd -> result	MVE
	p -> Rp	VDUPT.32 Qd,Rt		
int8x16_t [arm_]vdupq_x_n_s8(int8_t a, mve_pred16_t p)	a -> Rt p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
p)	p -> Kp	VDUPT.8 Qd,Rt		
int16x8_t [arm_]vdupq_x_n_s16(int16_t a,	a -> Rt	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VDUPT.16 Qd,Rt		
int32x4_t [arm_]vdupq_x_n_s32(int32_t a,	a -> Rt	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
uint8x16_t [arm_]vdupq_x_n_u8(uint8_t a,	a -> Rt	VDUPT.32 Qd,Rt VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu -> resuit	WIVE
		VDUPT.8 Qd,Rt		
uint16x8_t [arm_]vdupq_x_n_u16(uint16_t a,	a -> Rt	VMSR P0,Rp VPST	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VDUPT.16 Qd,Rt		
uint32x4_t [arm_]vdupq_x_n_u32(uint32_t a,	a -> Rt	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VDUPT.32 Qd,Rt		
float16x8_t [arm_]vdupq_x_n_f16(float16_t a,	a -> Rt	VMSR P0.Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	\(\frac{1}{2} \cdot \cdo	
C (22 4 (	. D:	VDUPT.16 Qd,Rt	01 . 1	Marc
float32x4_t [arm_]vdupq_x_n_f32(float32_t a, mve_pred16_t p)	a -> Rt p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
	r · · · · · ·	VDUPT.32 Qd,Rt		
mve_pred16_t [arm_]vcmpeqq[_f16](float16x8_t a,	a -> Qn	VCMP.F16 eq,Qn,Qm	Rd -> result	MVE
float16x8_t b)  mve_pred16_t [arm_]vcmpeqq[_f32](float32x4_t a,	b -> Qm a -> On	VMRS Rd,P0 VCMP.F32 eq,Qn,Qm	Rd -> result	MVE
float32x4_t b)	b -> Qm	VMRS Rd,P0	ra > resur	W L
mve_pred16_t [arm_]vcmpeqq[_s8](int8x16_t a,	a -> Qn	VCMP.I8 eq,Qn,Qm	Rd -> result	MVE
int8x16_t b)  mve_pred16_t [arm_]vcmpeqq[_s16](int16x8_t a,	b -> Qm a -> Qn	VMRS Rd,P0 VCMP.I16 eq,Qn,Qm	Rd -> result	MVE
int16x8_t b)	b -> Qm	VMRS Rd,P0	red > result	WYE
mve_pred16_t [arm_]vcmpeqq[_s32](int32x4_t a,	a -> Qn	VCMP.I32 eq,Qn,Qm	Rd -> result	MVE
int32x4_t b)  mve_pred16_t [arm_]vcmpeqq[_u8](uint8x16_t a,	b -> Qm a -> On	VMRS Rd,P0 VCMP.I8 eq,Qn,Qm	Rd -> result	MVE
uint8x16_t b)	b -> Qm	VMRS Rd,P0	ra > resur	W L
mve_pred16_t [arm_]vcmpeqq[_u16](uint16x8_t a,	a -> Qn	VCMP.I16 eq,Qn,Qm	Rd -> result	MVE
uint16x8_t b)  mve_pred16_t [arm_]vcmpeqq[_u32](uint32x4_t a,	b -> Qm a -> On	VMRS Rd,P0 VCMP.I32 eq,Qn,Qm	Rd -> result	MVE
uint32x4_t b)	b -> Qm	VMRS Rd,P0	Ku -> result	WW
mve_pred16_t [arm_]vcmpeqq[_n_f16](float16x8_t a,	a -> Qn	VCMP.F16 eq,Qn,Rm	Rd -> result	MVE
float16_t b)  mve pred16 t [ arm ]vcmpeqq[ n f32](float32x4 t a,	b -> Rm a -> Qn	VMRS Rd,P0 VCMP.F32 eq,Qn,Rm	Rd -> result	MVE
float32_t b)	b -> Rm	VMRS Rd,P0	Ru -> resuit	WIVE
mve_pred16_t [arm_]vcmpeqq[_n_s8](int8x16_t a,	a -> Qn	VCMP.I8 eq,Qn,Rm	Rd -> result	MVE
int8_t b)  mve_pred16_t [arm_]vcmpeqq[_n_s16](int16x8_t a,	b -> Rm a -> Qn	VMRS Rd,P0 VCMP.I16 eq,Qn,Rm	Rd -> result	MVE
int16_t b)	b -> Rm	VMRS Rd,P0	Ru -> resuit	WIVE
mve_pred16_t [arm_]vcmpeqq[_n_s32](int32x4_t a,	a -> Qn	VCMP.I32 eq,Qn,Rm	Rd -> result	MVE
int32_t b)  mve_pred16_t [arm_]vcmpeqq[_n_u8](uint8x16_t a,	b -> Rm a -> Qn	VMRS Rd,P0 VCMP.I8 eq,Qn,Rm	Rd -> result	MVE
uint8_t b)	b -> Rm	VCMF.18 eq,QII,KIII VMRS Rd,P0	Ku -> resuit	IVI V L
mve_pred16_t [arm_]vcmpeqq[_n_u16](uint16x8_t a,	a -> Qn	VCMP.I16 eq,Qn,Rm	Rd -> result	MVE
uint16_t b) mve_pred16_t [arm_]vcmpeqq[_n_u32](uint32x4_t a,	b -> Rm	VMRS Rd,P0 VCMP.I32 eq,Qn,Rm	Rd -> result	MVE
mve_pred16_t [arm_jvcmpeqq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VMRS Rd,P0	Ku -> resuit	IVI V E
mve_pred16_t [arm_]vcmpeqq_m[_f16](float16x8_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.F16 eq,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.F32 eq,Qn,Qm VMRS Rd,P0		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpeqq_m[_s8](int8x16_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.I8 eq,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_s16](int16x8_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.I16 eq,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_s32](int32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.I32 eq,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_u8](uint8x16_t a,	a -> On	VMSR P0,Rp	Rd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.I8 eq,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.I16 eq,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.I32 eq,Qn,Qm		
mve_pred16_t [arm_]vcmpeqq_m[_n_f16](float16x8_t	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
a, float16_t b, mve_pred16_t p)	b -> Rm	VPST	Ttu > Tesure	111,2
	p -> Rp	VCMPT.F16 eq,Qn,Rm		
mve_pred16_t [arm_]vcmpeqq_m[_n_f32](float32x4_t	a -> On	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
a, float32_t b, mve_pred16_t p)	b -> Rm	VPST	rta > resurt	WY E
	p -> Rp	VCMPT.F32 eq,Qn,Rm		
mve_pred16_t [arm_]vcmpeqq_m[_n_s8](int8x16_t a,	a -> On	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
int8_t b, mve_pred16_t p)	b -> Rm	VPST	Ru -> resurt	WYL
	p -> Rp	VCMPT.I8 eq,Qn,Rm		
mve_pred16_t [arm_]vcmpeqq_m[_n_s16](int16x8_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
int16_t b, mve_pred16_t p)	b -> Rm	VPST	Ru -> resurt	WYL
	p -> Rp	VCMPT.I16 eq,Qn,Rm		
mve_pred16_t [arm_]vcmpeqq_m[_n_s32](int32x4_t a,	a -> On	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST	rta > resurt	W. C.
	p -> Rp	VCMPT.I32 eq,Qn,Rm		
mve_pred16_t [arm_]vcmpeqq_m[_n_u8](uint8x16_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
uint8_t b, mve_pred16_t p)	b -> Rm	VPST	Ru -> resurt	WYL
	p -> Rp	VCMPT.I8 eq,Qn,Rm		
mve_pred16_t [arm_]vcmpeqq_m[_n_u16](uint16x8_t	a -> On	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
a, uint16_t b, mve_pred16_t p)	b -> Rm	VPST	Ru -> resurt	WVE
	p -> Rp	VCMPT.I16 eq,Qn,Rm		
mve_pred16_t [arm_]vcmpeqq_m[_n_u32](uint32x4_t	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
a, uint32_t b, mve_pred16_t p)	b -> Rm	VPST	Ku -> iesuit	WIVE
	p -> Rp	VCMPT.I32 eq,Qn,Rm		
mve pred16 t [ arm ]vcmpneq[ f16](float16x8 t a,	a -> Qn	VMRS Rd,P0 VCMP.F16 ne,On,Om	Rd -> result	MVE
float16x8_t b)	b -> Qm	VMRS Rd,P0	rta > resurt	WY E
mve_pred16_t [arm_]vcmpneq[_f32](float32x4_t a,	a -> Qn	VCMP.F32 ne,Qn,Qm	Rd -> result	MVE
float32x4_t b)  mve pred16 t [ arm ]vcmpneq[ s8](int8x16 t a,	b -> Qm a -> On	VMRS Rd,P0 VCMP.I8 ne,On,Om	Rd -> result	MVE
int8x16_t b)	b -> Qm	VMRS Rd,P0	Ku -/ ICSUIT	IVI V E
mve_pred16_t [arm_]vcmpneq[_s16](int16x8_t a,	a -> Qn	VCMP.I16 ne,Qn,Qm	Rd -> result	MVE
int16x8_t b) mve_pred16_t [arm_]vcmpneq[_s32](int32x4_t a,	b -> Qm a -> Qn	VMRS Rd,P0 VCMP.I32 ne,On,Om	Rd -> result	MVE
int32x4_t b)	b -> Qm	VMRS Rd,P0	Nu -> result	,.
mve_pred16_t [arm_]vcmpneq[_u8](uint8x16_t a,	a -> Qn	VCMP.I8 ne,Qn,Qm	Rd -> result	MVE
uint8x16_t b) mve_pred16_t [arm_]vcmpneq[_u16](uint16x8_t a,	b -> Qm a -> Qn	VMRS Rd,P0 VCMP.I16 ne,Qn,Qm	Rd -> result	MVE
uint16x8_t b)	a -> Qn b -> Qm	VCMP.116 ne,Qn,Qm VMRS Rd,P0	Ku -> resuit	IVI V I.
mve_pred16_t [arm_]vcmpneq[_u32](uint32x4_t a,	a -> Qn	VCMP.I32 ne,Qn,Qm	Rd -> result	MVE
uint32x4_t b)	b -> Qm	VMRS Rd,P0	D.4 >14	MVE
mve_pred16_t [arm_]vcmpneq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rd -> result	MVE
	$p \rightarrow Rp$	VCMPT.F16 ne,Qn,Qm		
		VMRS Rd,P0		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpneq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_n_f16](float16x8_t a, float16_t b)  mve_pred16_t [_ arm ]vcmpneq[_n_f12](float32x4_t a,	a -> Qn b -> Rm a -> On	VCMP.F16 ne,Qn,Rm VMRS Rd,P0 VCMP.F32 ne,Qn,Rm	Rd -> result	MVE MVE
float32_t b)  mve_pred16_t [arm_]vcmpneq[_n_s8](int8x16_t a,	b -> Rm a -> Qn	VMRS Rd,P0 VCMP.I8 ne,Qn,Rm	Rd -> result	MVE
int8_t b)  mve_pred16_t [arm_]vcmpneq[_n_s16](int16x8_t a,	b -> Rm a -> Qn	VMRS Rd,P0 VCMP.I16 ne,Qn,Rm	Rd -> result	MVE
int16_t b) mve_pred16_t [arm_]vcmpneq[_n_s32](int32x4_t a, int32_t b)	b -> Rm a -> Qn b -> Rm	VMRS Rd,P0 VCMP.I32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VCMP.I8 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VCMP.I16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VCMP.I32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I8 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I8 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpgeq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMP.F16 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Om	VCMP.F32 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VCMP.S8 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VCMP.S16 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VCMP.S32 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgeq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR PO,Rp VPST VCMPT.F16 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S8 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S16 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S32 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VCMP.F16 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VCMP.F32 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VCMP.S8 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VCMP.S16 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VCMP.S32 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S8 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S16 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	$\begin{array}{l} a \rightarrow Qn \\ b \rightarrow Rm \\ p \rightarrow Rp \end{array}$	VMSR P0,Rp VPST VCMPT.S32 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMP.F16 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMP.F32 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VCMP.S8 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgtq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VCMP.S16 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgtq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VCMP.S32 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 gt,Qn,Qm VMRS Rd.P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpgtq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgtq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S8 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgtq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S16 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S32 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VCMP.F16 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VCMP.F32 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VCMP.S8 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VCMP.S16 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VCMP.S32 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S8 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgtq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S16 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S32 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMP.F16 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMP.F32 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VCMP.S8 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VCMP.S16 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VCMP.S32 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMRS R0,P0 VMSR P0,Rp VPST VCMPT.S8 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpleq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S16 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S32 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpleq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VCMP.F16 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpleq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VCMP.F32 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq[_n_s8](int8x16_t a, int8 t b)	a -> Qn b -> Rm	VCMP.S8 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq[_n_s16](int16x8_t a, int16 t b)	a -> Qn b -> Rm	VCMP.S16 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VCMP.S32 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpleq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpleq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpleq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S8 le,Qn,Rm VMRS Rd.P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpleq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S16 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpleq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S32 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMP.F16 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMP.F32 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VCMP.S8 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VCMP.S16 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VCMP.S32 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S8 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S16 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S32 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VCMP.F16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VCMP.F32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VCMP.S8 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VCMP.S16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VCMP.S32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpltq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S8 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VCMP.U8 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VCMP.U16 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VCMP.U32 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq_m[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U8 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq_m[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U16 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U32 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VCMP.U8 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VCMP.U16 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VCMP.U32 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq_m[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.U8 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq_m[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.U16 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq_m[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.U32 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VCMP.U8 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmphiq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VCMP.U16 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VCMP.U32 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq_m[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U8 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq_m[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U16 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U32 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VCMP.U8 hi,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VCMP.U16 hi,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmphiq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VCMP.U32 hi,Qn,Rm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmphiq_m[_n_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
uint8_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.U8 hi,Qn,Rm VMRS Rd.P0		
mve_pred16_t [arm_]vcmphiq_m[_n_u16](uint16x8_t	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
a, uint16_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.U16 hi,Qn,Rm VMRS Rd,P0		
mve_pred16_t [arm_]vcmphiq_m[_n_u32](uint32x4_t	a -> On	VMSR P0,Rp	Rd -> result	MVE
a, uint32_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.U32 hi,Qn,Rm		
int8x16_t [arm_]vminq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VMRS Rd,P0 VMIN.S8 Qd,Qn,Qm	Od -> result	MVE/NEON
	b -> Qm		Ç	
int16x8_t [arm_]vminq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VMIN.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vminq[_s32](int32x4_t a, int32x4_t b)	b -> Qm a -> On	VMIN.S32 Qd,Qn,Qm	Od -> result	MVE/NEON
mt32x4_t [am_]vmmq[_332](mt32x4_t a, mt32x4_t b)	b -> Qm	VWIIV.552 Qu,Qii,Qiii	Qu -> result	WIVE/NEON
uint8x16_t [arm_]vminq[_u8](uint8x16_t a, uint8x16_t	a -> Qn	VMIN.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
b)	b -> Qm	VMIN III CO I O. O.	0.1	MVENIEON
uint16x8_t [arm_]vminq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Om	VMIN.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vminq[_u32](uint32x4_t a, uint32x4_t	a -> Qn	VMIN.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
b)	b -> Qm	VINTAR DO T		100
int8x16_t [_arm_]vminq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
mtox10_t a, mtox10_t b, mve_pred10_t p)	b -> Qm	VMINT.S8 Od,On,Om		
	p -> Rp			
int16x8_t [arm_]vminq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VMINT.S16 Qd,Qn,Qm		
	p -> Rp	V.VIII.V.I.510 Qu,Qii,Qiii		
int32x4_t [arm_]vminq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Om	VPST VMINT.S32 Qd,Qn,Qm		
	p -> QIII	VIVIINT.332 Qu,Qii,Qiii		
uint8x16_t [arm_]vminq_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VMINT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vminq_m[_u16](uint16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VMINT.U16 Qd,Qn,Qm		
uint32x4 t [ arm ]vming m[ u32](uint32x4 t inactive,	inactive -> Od	VMSR P0,Rp	Od -> result	MVE
uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VMINT.U32 Qd,Qn,Qm		
int8x16_t [arm_]vminq_x[_s8](int8x16_t a, int8x16_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Qm	VPST	Ç	
1.45 0.45 0.45 0.45 0.45 0.45 0.45 0.45 0	p -> Rp	VMINT.S8 Qd,Qn,Qm	01 1	) arm
int16x8_t [arm_]vminq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VMINT.S16 Qd,Qn,Qm		
int32x4_t [arm_]vminq_x[_s32](int32x4_t a, int32x4_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINT.S32 Qd,Qn,Qm		
uint8x16_t [arm_]vminq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
vintleve of some lumino of -167/-i-16-0 t-	p -> Rp	VMINT.U8 Qd,Qn,Qm	0414	MVE
uint16x8_t [arm_]vminq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VMINT.U16 Qd,Qn,Qm		
uint32x4_t [_arm_]vminq_x[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINT.U32 Qd,Qn,Qm		
uint8x16_t [arm_]vminaq[_s8](uint8x16_t a, int8x16_t	a -> Qda	VMINA.S8 Qda,Qm	Qda -> result	MVE
b)	b -> Qm			
uint16x8_t [arm_]vminaq[_s16](uint16x8_t a, int16x8_t b)	a -> Qda	VMINA.S16 Qda,Qm	Qda -> result	MVE
b) uint32x4_t [arm_]vminaq[_s32](uint32x4_t a, int32x4_t	b -> Qm a -> Qda	VMINA.S32 Qda,Qm	Qda -> result	MVE
b)	b -> Qm		Zun z Tobuit	
uint8x16_t [arm_]vminaq_m[_s8](uint8x16_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int8x16_t b, mve_pred16_t p)	b -> Qm	VPST VMINAT S8 Oda Om		
	p -> Rp	VMINAT.S8 Qda,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vminaq_m[_s16](uint16x8_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VMINAT.S16 Qda,Qm	0.114	MVE
uint32x4_t [arm_]vminaq_m[_s32](uint32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qda b -> Qm	VMSR P0,Rp VPST	Qda -> result	MVE
mi32x4_t o, mvc_picuro_t p)	p -> Rp	VMINAT.S32 Qda,Qm		
int8_t [arm_]vminvq[_s8](int8_t a, int8x16_t b)	a -> Rda	VMINV.S8 Rda,Qm	Rda -> result	MVE
	b -> Qm	/ 2		
int16_t [arm_]vminvq[_s16](int16_t a, int16x8_t b)	a -> Rda	VMINV.S16 Rda,Qm	Rda -> result	MVE
int32 t [ arm ]vminvq[ s32](int32 t a, int32x4 t b)	b -> Qm a -> Rda	VMINV.S32 Rda,Om	Rda -> result	MVE
int32_t [atm_]viiiiivq[_332](int32_t a, int32x+_t b)	b -> Qm	VIVIII V .552 Rda,Qiii	Rda -> resurt	WIVE
uint8_t [arm_]vminvq[_u8](uint8_t a, uint8x16_t b)	a -> Rda	VMINV.U8 Rda,Qm	Rda -> result	MVE
	b -> Qm			
uint16_t [arm_]vminvq[_u16](uint16_t a, uint16x8_t b)	a -> Rda	VMINV.U16 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vminvq[_u32](uint32_t a, uint32x4_t b)	b -> Qm a -> Rda	VMINV.U32 Rda,Qm	Rda -> result	MVE
umi32_t [atm_Jvmmvq[_u32](umi32_t a, umi32x4_t b)	b -> Qm	VWIIIVV.032 Kda,QIII	Kua => resurt	WIVE
int8_t [arm_]vminvq_p[_s8](int8_t a, int8x16_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VMINVT.S8 Rda,Qm		
int16_t [_arm_]vminvq_p[_s16](int16_t a, int16x8_t b,	a -> Rda b -> Om	VMSR P0,Rp VPST	Rda -> result	MVE
mve_pred16_t p)	p -> Qm p -> Rp	VMINVT.S16 Rda,Om		
int32 t [ arm ]vminvq p[ s32](int32 t a, int32x4 t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VMINVT.S32 Rda,Qm		
uint8_t [arm_]vminvq_p[_u8](uint8_t a, uint8x16_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINVT.U8 Rda,Qm		
uint16_t [arm_]vminvq_p[_u16](uint16_t a, uint16x8_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		1
	p -> Rp	VMINVT.U16 Rda,Qm		
uint32_t [arm_]vminvq_p[_u32](uint32_t a, uint32x4_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINVT.U32 Rda,Qm		
uint8_t [arm_]vminavq[_s8](uint8_t a, int8x16_t b)	a -> Rda	VMINAV.S8 Rda,Qm	Rda -> result	MVE
amo_t [am_], mma vq[_so](amto_t a, mto.rro_t o)	b -> Qm	viviii vi v ibo itaa, giii	reda y result	111 / 12
uint16_t [arm_]vminavq[_s16](uint16_t a, int16x8_t b)	a -> Rda	VMINAV.S16 Rda,Qm	Rda -> result	MVE
	b -> Qm			
uint32_t [arm_]vminavq[_s32](uint32_t a, int32x4_t b)	a -> Rda	VMINAV.S32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vminavq_p[_s8](uint8_t a, int8x16_t b,	b -> Qm a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm	VPST	Rda -> resurt	WIVE
	p -> Rp	VMINAVT.S8 Rda,Qm		
uint16_t [arm_]vminavq_p[_s16](uint16_t a, int16x8_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINAVT.S16 Rda,Om		
uint32_t [arm_]vminavq_p[_s32](uint32_t a, int32x4_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Om	VPST	redu > resurt	III V E
	p -> Rp	VMINAVT.S32 Rda,Qm		
float16x8_t [_arm_]vminnmq[_f16](float16x8_t a,	a -> Qn	VMINNM.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t b) float32x4 t [ arm ]vminnmq[ f32](float32x4 t a,	b -> Qm a -> Qn	VMINNM.F32 Qd,Qn,Qm	Od -> result	MVE/NEON
float32x4_t [arm_]vmmmq[_152](float32x4_t a, float32x4_t b)	b -> Qm	VMINNWI.F32 Qd,Qii,Qiii	Qu -> resuit	WIVE/INEON
float16x8_t [arm_]vminnmq_m[_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VMINNMT.F16 Qd,Qn,Qm		
float32x4_t [arm_]vminnmq_m[_f32](float32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn	VMSR PO,RP VPST	Qu -> resuit	MIVE
	b -> Qm	VMINNMT.F32 Qd,Qn,Qm		
	p -> Rp			
float16x8_t [_arm_]vminnmq_x[_f16](float16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
float32x4_t [arm_]vminnmq_x[_f32](float32x4_t a,	p -> Rp a -> Qn	VMINNMT.F16 Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
float32x4_t [arm_]vminnmq_x[_132](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR PO,RP VPST	Qu -> resuit	IVI V E
	p -> Rp	VMINNMT.F32 Qd,Qn,Qm		
float16x8_t [arm_]vminnmaq[_f16](float16x8_t a,	a -> Qda	VMINNMA.F16 Qda,Qm	Qda -> result	MVE
float16x8_t b)	b -> Qm		<b>_</b>	1
float32x4_t [arm_]vminnmaq[_f32](float32x4_t a,	a -> Qda	VMINNMA.F32 Qda,Qm	Qda -> result	MVE
float32x4_t b) float16x8_t [arm_]vminnmaq_m[_f16](float16x8_t a,	b -> Qm a -> Qda	VMSR P0,Rp	Qda -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VMSK FO,KP VPST	Qua -> result	17172
			•	•

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [arm_]vminnmaq_m[_f32](float32x4_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINNMAT.F32 Qda,Qm		
float16_t [arm_]vminnmvq[_f16](float16_t a,	a -> Rda	VMINNMV.F16 Rda,Qm	Rda -> result	MVE
float16x8_t b)	b -> Qm			
float32_t [arm_]vminnmvq[_f32](float32_t a, float32x4_t b)	a -> Rda b -> Qm	VMINNMV.F32 Rda,Qm	Rda -> result	MVE
float16_t [arm_]vminnmvq_p[_f16](float16_t a,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
float32_t [arm_]vminnmvq_p[_f32](float32_t a,	p -> Rp a -> Rda	VMINNMVT.F16 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Rua -> result	WVE
	p -> Rp	VMINNMVT.F32 Rda,Qm		1.00
float16_t [arm_]vminnmavq[_f16](float16_t a, float16x8_t b)	a -> Rda b -> Qm	VMINNMAV.F16 Rda,Qm	Rda -> result	MVE
float32_t [_arm_]vminnmavq[_f32](float32_t a, float32x4_t b)	a -> Rda b -> Qm	VMINNMAV.F32 Rda,Qm	Rda -> result	MVE
float16_t [_arm_]vminnmavq_p[_f16](float16_t a,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
float32 t [ arm ]vminnmavq p[ f32](float32 t a,	p -> Rp a -> Rda	VMINNMAVT.F16 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Ttua > Tesure	111,12
	p -> Rp	VMINNMAVT.F32 Rda,Qm	0.1	MUENICON
int8x16_t [arm_]vmaxq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMAX.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vmaxq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMAX.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vmaxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMAX.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vmaxq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMAX.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vmaxq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMAX.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vmaxq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Om	VMAX.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vmaxq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VMAXT.S8 Qd,Qn,Qm		
	p -> Rp	VWAX1.56 Qu,Qii,Qiii		
int16x8_t [arm_]vmaxq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VMAXT.S16 Qd,Qn,Qm		
	p -> Rp	2 . 2 . 2		
int32x4_t [arm_]vmaxq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
intoza4_t a, intoza4_t b, inve_preuto_t p)	b -> Qm	VMAXT.S32 Qd,Qn,Qm		
	p -> Rp			
uint8x16_t [_arm_]vmaxq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
unitox10_t a, unitox10_t b, inve_pleu10_t p)	b -> Qm	VMAXT.U8 Qd,Qn,Qm		
	p -> Rp	Andre Po P		) am
uint16x8_t [arm_]vmaxq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qm	VMAXT.U16 Qd,Qn,Qm		
uint32x4 t [ arm ]vmaxq m[ u32](uint32x4 t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t [arm_]vmaxq_m[_u32](uint32x4_t mactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qu	VMSK FO,KP VPST	Qu -> resuit	NIVE
· · - · - · - · ·	b -> Qm	VMAXT.U32 Qd,Qn,Qm		
int8x16_t [arm_]vmaxq_x[_s8](int8x16_t a, int8x16_t	p -> Rp a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST	Qu => resurt	WVE
intle-0 tf house of 100 tf 0 t 100 t	p -> Rp	VMAXT.S8 Qd,Qn,Qm	01 2 1	MVE
int16x8_t [arm_]vmaxq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
· -• -•	p -> Rp	VMAXT.S16 Qd,Qn,Qm		
int32x4_t [arm_]vmaxq_x[_s32](int32x4_t a, int32x4_t b, mve pred16 t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
o, mvc_picuio_t p)	b -> Qm p -> Rp	VMAXT.S32 Qd,Qn,Qm		
uint8x16_t [arm_]vmaxq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vmaxq_x[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST	-	
uint32x4_t [arm_]vmaxq_x[_u32](uint32x4_t a,	p -> Rp a -> Qn	VMAXT.U16 Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
uint32x4_t tarm_jvinaxq_xt_u32j(uint32x4_t a, uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST	2a > resum	
	p -> Rp	VMAXT.U32 Qd,Qn,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [_arm_]vmaxaq[_s8](uint8x16_t a, int8x16_t b)	a -> Qda b -> Qm	VMAXA.S8 Qda,Qm	Qda -> result	MVE
uint16x8_t [_arm_]vmaxaq[_s16](uint16x8_t a, int16x8_t b)	a -> Qda b -> Qm	VMAXA.S16 Qda,Qm	Qda -> result	MVE
uint32x4_t [_arm_]vmaxaq[_s32](uint32x4_t a, int32x4_t b)	a -> Qda b -> Qm	VMAXA.S32 Qda,Qm	Qda -> result	MVE
uint8x16_t [arm_]vmaxaq_m[_s8](uint8x16_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXAT.S8 Qda,Qm		
uint16x8_t [arm_]vmaxaq_m[_s16](uint16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qda b -> Qm	VMSR P0,Rp VPST	Qda -> result	MVE
uint32x4_t [arm_]vmaxaq_m[_s32](uint32x4_t a,	p -> Rp a -> Qda	VMAXAT.S16 Qda,Qm VMSR P0,Rp	Qda -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXAT.S32 Qda,Qm		
int8_t [_arm_]vmaxvq[_s8](int8_t a, int8x16_t b)	a -> Rda b -> Qm	VMAXV.S8 Rda,Qm	Rda -> result	MVE
int16_t [arm_]vmaxvq[_s16](int16_t a, int16x8_t b)	a -> Rda b -> Qm	VMAXV.S16 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vmaxvq[_s32](int32_t a, int32x4_t b)	a -> Rda	VMAXV.S32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vmaxvq[_u8](uint8_t a, uint8x16_t b)	b -> Qm a -> Rda	VMAXV.U8 Rda,Qm	Rda -> result	MVE
uint16_t [_arm_]vmaxvq[_u16](uint16_t a, uint16x8_t b)	b -> Qm a -> Rda	VMAXV.U16 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vmaxvq[_u32](uint32_t a, uint32x4_t b)	b -> Qm a -> Rda	VMAXV.U32 Rda,Qm	Rda -> result	MVE
int8_t [arm_]vmaxvq_p[_s8](int8_t a, int8x16_t b,	b -> Qm a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXVT.S8 Rda,Om		
int16_t [_arm_]vmaxvq_p[_s16](int16_t a, int16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
	p -> Rp	VMAXVT.S16 Rda,Qm	D.L. Is	) Marie
int32_t [_arm_]vmaxvq_p[_s32](int32_t a, int32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
uint8_t [arm_]vmaxvq_p[_u8](uint8_t a, uint8x16_t b,	p -> Rp a -> Rda	VMAXVT.S32 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXVT.U8 Rda,Qm		
uint16_t [_arm_]vmaxvq_p[_u16](uint16_t a, uint16x8_t b, mve_pred16_t p)	a -> Rda b -> Om	VMSR P0,Rp VPST	Rda -> result	MVE
uint32_t [_arm_lvmaxvq_p[_u32](uint32_t a, uint32x4_t	p -> Rp a -> Rda	VMAXVT.U16 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXVT.U32 Rda,Qm	reda > resure	MVE
uint8_t [arm_]vmaxavq[_s8](uint8_t a, int8x16_t b)	a -> Rda	VMAXAV1.U32 Rda,Qm	Rda -> result	MVE
uint16_t [_arm_]vmaxavq[_s16](uint16_t a, int16x8_t b)	b -> Qm a -> Rda	VMAXAV.S16 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vmaxavq[_s32](uint32_t a, int32x4_t b)	b -> Qm a -> Rda	VMAXAV.S32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vmaxavq_p[_s8](uint8_t a, int8x16_t b,	b -> Qm a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXAVT.S8 Rda,Qm		
uint16_t [_arm_]vmaxavq_p[_s16](uint16_t a, int16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
uint32_t [_arm_]vmaxavq_p[_s32](uint32_t a, int32x4_t	p -> Rp a -> Rda	VMAXAVT.S16 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
uint32_t [arm_]vmaxavq_p[_s32](uint32_t a, int32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Nua -> resuit	IVI V E
float16x8_t [_arm_]vmaxnmq[_f16](float16x8_t a,	p -> Rp a -> Qn	VMAXAVT.S32 Rda,Qm VMAXNM.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t b) float32x4_t [_arm_]vmaxnmq[_f32](float32x4_t a,	b -> Qm a -> Qn	VMAXNM.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t b) float16x8_t [arm_]vmaxnmq_m[_f16](float16x8_t	b -> Qm inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VMAXNMT.F16 Qd,Qn,Qm		
float32x4_t [_arm_]vmaxnmq_m[_f32](float32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VMAXNMT.F32 Qd,Qn,Qm	2a > Icoun	
floating at any leaves of floating at a	p -> Rp		04 5 10	MVE
float16x8_t [arm_]vmaxnmq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VMAXNMT.F16 Qd,Qn,Qm	_1	

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [arm_]vmaxnmq_x[_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXNMT.F32 Qd,Qn,Qm		
float16x8_t [_arm_]vmaxnmaq[_f16](float16x8_t a, float16x8_t b)	a -> Qda b -> Qm	VMAXNMA.F16 Qda,Qm	Qda -> result	MVE
float32x4_t [_arm_]vmaxnmaq[_f32](float32x4_t a, float32x4_t b)	a -> Qda b -> Qm	VMAXNMA.F32 Qda,Qm	Qda -> result	MVE
float16x8_t [arm_]vmaxnmaq_m[_f16](float16x8_t a,	a -> Qda	VMSR P0,Rp	Oda -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXNMAT.F16 Qda,Qm		
float32x4_t [arm_]vmaxnmaq_m[_f32](float32x4_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXNMAT.F32 Qda,Qm		
float16_t [arm_]vmaxnmvq[_f16](float16_t a, float16x8_t b)	a -> Rda b -> Qm	VMAXNMV.F16 Rda,Qm	Rda -> result	MVE
float32_t [_arm_]vmaxnmvq[_f32](float32_t a, float32_x4_t b)	a -> Rda b -> Qm	VMAXNMV.F32 Rda,Qm	Rda -> result	MVE
float16_t [arm_]vmaxnmvq_p[_f16](float16_t a,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXNMVT.F16 Rda,Qm		
float32_t [arm_]vmaxnmvq_p[_f32](float32_t a,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
G -16 -1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1	p -> Rp	VMAXNMVT.F32 Rda,Qm	D1 1	) (T ) (T )
float16_t [arm_]vmaxnmavq[_f16](float16_t a, float16x8_t b)	a -> Rda b -> Qm	VMAXNMAV.F16 Rda,Qm	Rda -> result	MVE
float32_t [_arm_]vmaxnmavq[_f32](float32_t a, float32x4_t b)	a -> Rda b -> Qm	VMAXNMAV.F32 Rda,Qm	Rda -> result	MVE
float16_t [arm_]vmaxnmavq_p[_f16](float16_t a,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMAXNMAVT.F16 Rda,Qm		
float32_t [arm_]vmaxnmavq_p[_f32](float32_t a,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VMAXNMAVT.F32 Rda,Qm		
uint32_t [_arm_]vabavq[_s8](uint32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> On	VABAV.S8 Rda,Qn,Qm	Rda -> result	MVE
mox10_t e/	c -> Qm			
uint32_t [arm_]vabavq[_s16](uint32_t a, int16x8_t b, int16x8_t c)	a -> Rda b -> Qn	VABAV.S16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [arm_]vabavq[_s32](uint32_t a, int32x4_t b,	c -> Qm a -> Rda	VABAV.S32 Rda,Qn,Qm	Rda -> result	MVE
int32x4_t c)	b -> Qn c -> Qm			
uint32_t [arm_]vabavq[_u8](uint32_t a, uint8x16_t b,	a -> Rda	VABAV.U8 Rda,Qn,Qm	Rda -> result	MVE
uint8x16_t c)	b -> Qn c -> Qm			
uint32_t [arm_]vabavq[_u16](uint32_t a, uint16x8_t b,	a -> Rda	VABAV.U16 Rda,Qn,Qm	Rda -> result	MVE
uint16x8_t c)	b -> Qn c -> Om			
uint32_t [arm_]vabavq[_u32](uint32_t a, uint32x4_t b,	a -> Rda	VABAV.U32 Rda,Qn,Qm	Rda -> result	MVE
uint32x4_t c)	b -> Qn c -> Qm			
uint32_t [arm_]vabavq_p[_s8](uint32_t a, int8x16_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
int8x16_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Qm p -> Rp	VABAVT.S8 Rda,Qn,Qm		
uint32_t [arm_]vabavq_p[_s16](uint32_t a, int16x8_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
int16x8_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Qm	VABAVT.S16 Rda,Qn,Qm		
uint32_t [arm_]vabavq_p[_s32](uint32_t a, int32x4_t b,	p -> Rp a -> Rda	VMSR P0,Rp	Rda -> result	MVE
int32x4_t c, mve_pred16_t p)	b -> Qn	VPST		1
	c -> Qm p -> Rp	VABAVT.S32 Rda,Qn,Qm		
uint32_t [arm_]vabavq_p[_u8](uint32_t a, uint8x16_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
uint8x16_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Qm p -> Rp	VABAVT.U8 Rda,Qn,Qm		
uint32_t [arm_]vabavq_p[_u16](uint32_t a, uint16x8_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, uint16x8_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Qm p -> Rp	VABAVT.U16 Rda,Qn,Qm		
uint32_t [arm_]vabavq_p[_u32](uint32_t a, uint32x4_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, uint32x4_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Qm p -> Rp	VABAVT.U32 Rda,Qn,Qm		
int8x16_t [arm_]vabdq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VABD.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
	b -> Qm			

	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vabdq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VABD.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vabdq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VABD.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vabdq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VABD.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vabdq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VABD.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vabdq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VABD.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vabdq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VABD.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]vabdq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VABD.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vabdq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vabdq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4\_t \ [\_arm\_]vabdq\_m [\_s32] (int32x4\_t \ inactive, \\ int32x4\_t \ a, int32x4\_t \ b, mve\_pred16\_t \ p)$	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vabdq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vabdq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [arm_]vabdq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [arm_]vabdq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vabdq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.F32 Qd,Qn,Qm	Qd -> result	MVE
$int8x16\_t \ [\_arm\_]vabdq\_x[\_s8](int8x16\_t \ a, int8x16\_t \ b, \\ mve\_pred16\_t \ p)$	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vabdq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vabdq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [arm_]vabdq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vabdq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vabdq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vabdq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vabdq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vabsq[_f16](float16x8_t a) float32x4_t [_arm_]vabsq[_f32](float32x4_t a)	a -> Qm	VABS.F16 Qd,Qm VABS.F32 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
int8x16_t [arm_]vabsq[_s8](int8x16_t a)	a -> Qm a -> Qm	VABS.F32 Qd,Qm VABS.S8 Qd,Qm	Qd -> result  Qd -> result	MVE/NEON MVE/NEON
int16x8_t [_arm_]vabsq[_s16](int16x8_t a) int32x4_t [_arm_]vabsq[_s32](int32x4_t a)	a -> Qm a -> Qm	VABS.S16 Qd,Qm VABS.S32 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [_arm_]vabsq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.F16 Od,Om	Qd -> result	MVE
float32x4_t [_arm_]vabsq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.F32 Qd,Qm	Qd -> result	MVE
int8x16_t [_arm_]vabsq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S8 Qd,Qm	Qd -> result	MVE
int16x8_t [_arm_]vabsq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S16 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vabsq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S32 Qd,Qm	Qd -> result	MVE
float16x8_t [_arm_]vabsq_x[_f16](float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vabsq_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.F32 Qd,Qm	Qd -> result	MVE
int8x16_t [_arm_]vabsq_x[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S8 Qd,Qm	Qd -> result	MVE
int16x8_t [_arm_]vabsq_x[_s16](int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S16 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vabsq_x[_s32](int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S32 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vadciq[_s32](int32x4_t a, int32x4_t b, unsigned * carry_out)	a -> Qn b -> Qm	VADCI.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
uint32x4_t [_arm_]vadciq[_u32](uint32x4_t a, uint32x4_t b, unsigned * carry_out)	a -> Qn b -> Qm	VADCI.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
int32x4_t [_arm_]vadciq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, unsigned * carry_out, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADCIT.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
uint32x4_t [_arm_]vadciq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, unsigned * carry_out, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADCIT.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
int32x4_t [_arm_]vadcq[_s32](int32x4_t a, int32x4_t b, unsigned * carry)	a -> Qn b -> Qm *carry -> Rt	VMRS Rs,FPSCR_nzcvqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzcvqc,Rs VADC.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
uint32x4_t [_arm_]vadcq[_u32](uint32x4_t a, uint32x4_t b, unsigned * carry)	a -> Qn b -> Qm *carry -> Rt	VMRS Rs,FPSCR_nzevqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzevqc,Rs VADC.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
int32x4_t [_arm_]vadcq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, unsigned * carry, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm *carry -> Rt p -> Rp	VMRS Rs,FPSCR_nzcvqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzcvqc,Rs VMSR P0,Rp VPST VADCT.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vadcq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, unsigned * carry, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm *carry -> Rt p -> Rp	VMRS Rs,FPSCR_nzevqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzevqe,Rs VMSR P0,Rp VPST VADCT.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
float16x8_t [arm_]vaddq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VADD.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]vaddq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VADD.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vaddq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VADD.F16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [arm_]vaddq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VADD.F32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vaddq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VADD.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vaddq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VADD.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vaddq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VADD.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vaddq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VADD.I8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [_arm_]vaddq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VADD.I16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vaddq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VADD.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vaddq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VADD.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vaddq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VADD.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vaddq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VADD.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vaddq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VADD.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vaddq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VADD.I16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vaddq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VADD.I32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [_arm_]vaddq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vaddq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [arm_]vaddq_m[_n_f16](float16x8_t inactive, float16x8_t a, float16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.F16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [_arm_]vaddq_m[_n_f32](float32x4_t inactive, float32x4_t a, float32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.F32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vaddq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vaddq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vaddq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vaddq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vaddq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I16 Qd,Qn,Rm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [_arm_]vaddq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vaddq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vaddq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vaddq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [arm_]vaddq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [arm_]vaddq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm	VMSR P0,Rp VPST VADDT.I16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vaddq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [arm_]vaddq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.F16 Od,On,Om	Qd -> result	MVE
float32x4_t [_arm_]vaddq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vaddq_x[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.F16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [arm_]vaddq_x[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.F32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vaddq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vaddq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm p -> Rp	VADDT.18 Qd,Qii,Qiii  VMSR P0,Rp  VPST  VADDT.116 Qd,Qii,Qiii	Qd -> result	MVE
int32x4_t [arm_]vaddq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vaddq_x[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [_arm_]vaddq_x[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vaddq_x[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vaddq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vaddq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vaddq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vaddq_x[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vaddq_x[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vaddq_x[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VADDT.ITO Qd,Qn,Rm  VMSR P0,Rp  VPST  VADDT.I32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vclsq[_s8](int8x16_t a)	a -> Qm	VCLS.S8 Qd,Qm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vclsq[_s16](int16x8_t a)	a -> Qm	VCLS.S16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vclsq[_s32](int32x4_t a)	a -> Qm	VCLS.S32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vclsq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCLST.S8 Qd,Qm		
int16x8_t [arm_]vclsq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
1.00 4 . 5 . 1 . 1 . 5 . 201/3 . 20 . 4 . 1 . 3	p -> Rp	VCLST.S16 Qd,Qm	0.1	) arm
int32x4_t [_arm_]vclsq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VCLST.S32 Qd,Qm		
int8x16_t [arm_]vclsq_x[_s8](int8x16_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VPST	Qu => resuit	WIVE
inve_pred10_t p)	p -> Kp	VCLST.S8 Qd,Qm		
int16x8_t [arm_]vclsq_x[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		·
_i _ i /		VCLST.S16 Qd,Qm		
int32x4_t [arm_]vclsq_x[_s32](int32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VCLST.S32 Qd,Qm		
int8x16_t [arm_]vclzq[_s8](int8x16_t a)	a -> Qm	VCLZ.I8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vclzq[_s16](int16x8_t a)	a -> Qm	VCLZ.I16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vclzq[_s32](int32x4_t a)	a -> Qm	VCLZ.I32 Qd,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vclzq[_u8](uint8x16_t a)	a -> Qm	VCLZ.I8 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vclzq[_u16](uint16x8_t a)	a -> Qm	VCLZ.I16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vclzq[_u32](uint32x4_t a)	a -> Qm	VCLZ.I32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vclzq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
: 16 0 . 6 1 1 1 5 1676 16 0	p -> Rp	VCLZT.I8 Qd,Qm	0.1 1:	MATE
int16x8_t [arm_]vclzq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
introxe_t a, inve_predio_t p)	p -> Rp	VCLZT.I16 Qd,Qm		
int32x4_t [arm_]vclzq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int32x4_t iarmver2d_int_s32s(int32x4_t inactive, int32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Qu => resuit	WIVE
miszka-t a, mvo_predio_t p)	p -> Rp	VCLZT.I32 Qd,Qm		
uint8x16_t [arm_]vclzq_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
uint8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
_ · · _ • · _ •	p -> Rp	VCLZT.I8 Qd,Qm		
uint16x8_t [arm_]vclzq_m[_u16](uint16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCLZT.I16 Qd,Qm		
uint32x4_t [arm_]vclzq_m[_u32](uint32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
int8x16 t[ arm ]vclzq x[ s8](int8x16 t a,	p -> Rp	VCLZT.I32 Qd,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	a -> Qm p -> Rp	VMSK PO,KP VPST	Qu -> resuit	MIVE
mvc_pred10_t p)	p -> Kp	VCLZT.I8 Qd,Qm		
int16x8_t [arm_]vclzq_x[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu' > Tesan	
-1 - 1 <sup>2</sup>		VCLZT.I16 Qd,Qm		
int32x4_t [arm_]vclzq_x[_s32](int32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
	_	VCLZT.I32 Qd,Qm		1
uint8x16_t [_arm_]vclzq_x[_u8](uint8x16_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
vint16v0 tf ann lyalaa yf y16l/vint16v0 ta	a > 0	VCLZT.I8 Qd,Qm VMSR P0,Rp	Od -> result	MVE
uint16x8_t [arm_]vclzq_x[_u16](uint16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSK PU,KP VPST	Qu -> resuit	MIVE
nive_pred10_t p)	p -> Kp	VCLZT.I16 Qd,Qm		
uint32x4_t [arm_]vclzq_x[_u32](uint32x4_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu > resurt	11112
	rr	VCLZT.I32 Qd,Qm		
float16x8_t [arm_]vnegq[_f16](float16x8_t a)	a -> Qm	VNEG.F16 Qd,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]vnegq[_f32](float32x4_t a)	a -> Qm	VNEG.F32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vnegq[_s8](int8x16_t a)	a -> Qm	VNEG.S8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vnegq[_s16](int16x8_t a)	a -> Qm	VNEG.S16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vnegq[_s32](int32x4_t a)	a -> Qm	VNEG.S32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vnegq_m[_f16](float16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VNEGT.F16 Qd,Qm		
float32x4_t [arm_]vnegq_m[_f32](float32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VNEGT.F32 Qd,Qm		1.07
int8x16_t [_arm_]vnegq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VNEGT.S8 Qd,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vnegq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VNEGT.S16 Qd,Qm		
int32x4_t [arm_]vnegq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
float16x8_t [arm_]vnegq_x[_f16](float16x8_t a,	p -> Rp a -> Qm	VNEGT.S32 Qd,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VMSK FO,Kp VPST	Qu -> resuit	WIVE
		VNEGT.F16 Qd,Qm		
float32x4_t [_arm_]vnegq_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VNEGT.F32 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vnegq_x[_s8](int8x16_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VNEGT.S8 Qd,Qm		
int16x8_t [arm_]vnegq_x[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
int32x4_t [arm_]vnegq_x[_s32](int32x4_t a,	a -> Qm	VNEGT.S16 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu -> result	WIVE
		VNEGT.S32 Qd,Qm		
int8x16_t [arm_]vmulhq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vmulhq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VMULH.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4 t [ arm ]vmulhq[ s32](int32x4 t a, int32x4 t b)	b -> Qm a -> On	VMULH.S32 Qd,Qn,Qm	Od -> result	MVE
mis2x4_t [arm_jvmumq[_s32](mis2x4_t a, mis2x4_t b)	b -> Qm	VMULH.332 Qd,Qii,Qiii	Qu -> resuit	MIVE
uint8x16_t [arm_]vmulhq[_u8](uint8x16_t a, uint8x16_t	a -> Qn	VMULH.U8 Qd,Qn,Qm	Qd -> result	MVE
b) uint16x8_t [arm_]vmulhq[_u16](uint16x8_t a,	b -> Qm a -> On	VMULH.U16 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t b)	b -> Qm	. , , , ,		
uint32x4_t [_arm_]vmulhq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMULH.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vmulhq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Om	VPST VMULHT.S8 Qd,Qn,Qm		
	p -> Rp	VMOLH1.38 Qu,Qii,Qiii		
int16x8_t [arm_]vmulhq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VMULHT.S16 Qd,Qn,Qm		
	p -> Rp	, 1102111.010 Qu,Q.i,Q.ii		
int32x4_t [_arm_]vmulhq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMULHT.S32 Qd,Qn,Qm		
	p -> Rp	2 . 2 . 2		
uint8x16_t [_arm_]vmulhq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
unitox10_t a, unitox10_t b, inve_pred10_t p)	b -> Qm	VMULHT.U8 Qd,Qn,Qm		
	p -> Rp			
uint16x8_t [_arm_]vmulhq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
amitono_t a, amitono_t e, mve_prearo_t p/	b -> Qm	VMULHT.U16 Qd,Qn,Qm		
uint32x4_t [arm_]vmulhq_m[_u32](uint32x4_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn	VPST	Qu -> resuit	WIVE
-	b -> Qm	VMULHT.U32 Qd,Qn,Qm		
int8x16_t [arm_]vmulhq_x[_s8](int8x16_t a, int8x16_t	p -> Rp a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST	22 3 100000	1
int16x8 t [ arm ]vmulhq x[ s16](int16x8 t a, int16x8 t	p -> Rp	VMULHT.S8 Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR PU,Rp VPST	Qu -> resuit	IVI V L
	p -> Rp	VMULHT.S16 Qd,Qn,Qm		1,0=
int32x4_t [arm_]vmulhq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
· -• -• -•	p -> Rp	VMULHT.S32 Qd,Qn,Qm		
uint8x16_t [arm_]vmulhq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMULHT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vmulhq_x[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMULHT.U16 Qd,Qn,Qm		
uint32x4_t [arm_]vmulhq_x[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
uint16x8_t [arm_]vmullbq_poly[_p8](uint8x16_t a,	p -> Rp a -> Qn	VMULHT.U32 Qd,Qn,Qm VMULLB.P8 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t b)	b -> Qm		Q_ : Toodie	

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vmullbq_poly[_p16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMULLB.P16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vmullbq_int[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMULLB.S8 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vmullbq_int[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMULLB.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [_arm_]vmullbq_int[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMULLB.S32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmullbq_int[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Om	VMULLB.U8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmullbq_int[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMULLB.U16 Qd,Qn,Qm	Qd -> result	MVE
uint64x2_t [_arm_]vmullbq_int[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMULLB.U32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmullbq_poly_m[_p8](uint16x8_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.P8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmullbq_poly_m[_p16](uint32x4_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.P16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vmullbq_int_m[_s8](int16x8_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.S8 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vmullbq_int_m[_s16](int32x4_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [arm_]vmullbq_int_m[_s32](int64x2_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmullbq_int_m[_u8](uint16x8_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmullbq_int_m[_u16](uint32x4_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint64x2_t [_arm_]vmullbq_int_m[_u32](uint64x2_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.U32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmullbq_poly_x[_p8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.P8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmullbq_poly_x[_p16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.P16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vmullbq_int_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.S8 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vmullbq_int_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [arm_]vmullbq_int_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vmullbq_int_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [arm_]vmullbq_int_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint64x2_t [_arm_]vmullbq_int_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLBT.U32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmulltq_poly[_p8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMULLT.P8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmulltq_poly[_p16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMULLT.P16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vmulltq_int[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMULLT.S8 Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [_arm_]vmulltq_int[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMULLT.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [_arm_]vmulltq_int[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMULLT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmulltq_int[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMULLT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmulltq_int[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMULLT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint64x2_t [_arm_]vmulltq_int[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMULLT.U32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmulltq_poly_m[_p8](uint16x8_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.P8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmulltq_poly_m[_p16](uint32x4_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.P16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vmulltq_int_m[_s8](int16x8_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.S8 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vmulltq_int_m[_s16](int32x4_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [arm_]vmulltq_int_m[_s32](int64x2_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmulltq_int_m[_u8](uint16x8_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmulltq_int_m[_u16](uint32x4_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint64x2_t [_arm_]vmulltq_int_m[_u32](uint64x2_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.U32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmulltq_poly_x[_p8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.P8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmulltq_poly_x[_p16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.P16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vmulltq_int_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.S8 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vmulltq_int_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [arm_]vmulltq_int_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmulltq_int_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmulltq_int_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint64x2_t [_arm_]vmulltq_int_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULLTT.U32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [arm_]vmulq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VMUL.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [arm_]vmulq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VMUL.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [_arm_]vmulq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VMUL.F16 Qd,Qn,Rm	Qd -> result	MVE/NEON
float32x4_t [_arm_]vmulq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VMUL.F32 Qd,Qn,Rm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vmulq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMUL.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vmulq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMUL.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vmulq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMUL.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vmulq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VMUL.I8 Qd,Qn,Rm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vmulq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VMUL.I16 Qd,Qn,Rm	Qd -> result	MVE/NEON
int32x4_t [arm_]vmulq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VMUL.I32 Qd,Qn,Rm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vmulq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMUL.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vmulq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMUL.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vmulq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMUL.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vmulq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VMUL.I8 Qd,Qn,Rm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vmulq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VMUL.I16 Qd,Qn,Rm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vmulq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VMUL.I32 Qd,Qn,Rm	Qd -> result	MVE/NEON
float16x8_t [_arm_]vmulq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qm p -> Rp	VMULT.F16 Qd,Qn,Qm		
float32x4_t [_arm_]vmulq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vmulq_m[_n_f16](float16x8_t inactive, float16x8_t a, float16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.F16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [_arm_]vmulq_m[_n_f32](float32x4_t inactive, float32x4_t a, float32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.F32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vmulq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vmulq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vmulq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vmulq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [_arm_]vmulq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vmulq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vmulq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmulq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmulq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I32 Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [arm_]vmulq_m[_n_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VMULT.I8 Qd,Qn,Rm		
1450.5	p -> Rp	VII (OD DO D	0.1	) a m
uint16x8_t [_arm_]vmulq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, unitroxo_t a, unitro_t b, mve_predro_t p)	b -> Rm	VMULT.I16 Qd,Qn,Rm		
	p -> Rp	VMC21.110 Qu,Qu,Ittiii		
uint32x4_t [arm_]vmulq_m[_n_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm p -> Rp	VMULT.I32 Qd,Qn,Rm		
float16x8_t [arm_]vmulq_x[_f16](float16x8_t a,	a -> Qn	VMSR P0,Rp	Od -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Qu' > Tesan	111,12
	p -> Rp	VMULT.F16 Qd,Qn,Qm		
float32x4_t [arm_]vmulq_x[_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
float16x8_t [arm_]vmulq_x[_n_f16](float16x8_t a,	p -> Rp a -> On	VMULT.F32 Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
float16_t b, mve_pred16_t p)	b -> Rm	VPST	Qu -> resuit	WIVE
	p -> Rp	VMULT.F16 Qd,Qn,Rm		
float32x4_t [arm_]vmulq_x[_n_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float32_t b, mve_pred16_t p)	b -> Rm	VPST		
int8x16_t [arm_]vmulq_x[_s8](int8x16_t a, int8x16_t b,	p -> Rp a -> On	VMULT.F32 Qd,Qn,Rm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	b -> Qn	VMSK PO,KP VPST	Qu -> resuit	MVE
invo_prearo_t p)	p -> Rp	VMULT.I8 Qd,Qn,Qm		
int16x8_t [arm_]vmulq_x[_s16](int16x8_t a, int16x8_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
1.00 4.5	p -> Rp	VMULT.I16 Qd,Qn,Qm	0.1	No.
int32x4_t [arm_]vmulq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
b, live_pied1o_t p)	p -> Rp	VMULT.I32 Qd,Qn,Qm		
int8x16_t [arm_]vmulq_x[_n_s8](int8x16_t a, int8_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Rm	VPST	`	
	p -> Rp	VMULT.I8 Qd,Qn,Rm		
int16x8_t [arm_]vmulq_x[_n_s16](int16x8_t a, int16_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VMULT.I16 Qd,Qn,Rm		
int32x4_t [arm_]vmulq_x[_n_s32](int32x4_t a, int32_t	a -> On	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST	(a ·	1
	p -> Rp	VMULT.I32 Qd,Qn,Rm		
uint8x16_t [arm_]vmulq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMULT.I8 Qd,Qn,Qm		
uint16x8 t [ arm ]vmulq x[ u16](uint16x8 t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST	(a ·	1
	p -> Rp	VMULT.I16 Qd,Qn,Qm		
uint32x4_t [arm_]vmulq_x[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
uint8x16_t [arm_]vmulq_x[_n_u8](uint8x16_t a, uint8_t	p -> Rp a -> Qn	VMULT.I32 Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST	Qu > resuit	1
	p -> Rp	VMULT.I8 Qd,Qn,Rm		
uint16x8_t [arm_]vmulq_x[_n_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint16_t b, mve_pred16_t p)	b -> Rm	VPST VMULT.I16 Qd,Qn,Rm		
uint32x4_t [arm_]vmulq_x[_n_u32](uint32x4_t a,	p -> Rp a -> Qn	VMULT.116 Qd,Qn,Rm VMSR P0,Rp	Qd -> result	MVE
uint32x4_t [armjvintiq_x[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	b -> Rm	VPST	Zu > resuit	
	p -> Rp	VMULT.I32 Qd,Qn,Rm		
int32x4_t [arm_]vsbciq[_s32](int32x4_t a, int32x4_t b,	a -> Qn	VSBCI.I32 Qd,Qn,Qm	Qd -> result	MVE
unsigned * carry_out)	b -> Qm	VMRS Rt,FPSCR_nzcvqc	Rt ->	
		LSR Rt,#29 AND Rt,#1	*carry_out	
uint32x4_t [arm_]vsbciq[_u32](uint32x4_t a,	a -> Qn	VSBCI.I32 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t b, unsigned * carry_out)	b -> Qm	VMRS Rt,FPSCR_nzcvqc	Rt ->	
		LSR Rt,#29	*carry_out	
1.00 4.15		AND Rt,#1	01.	MUE
int32x4_t [_arm_]vsbciq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result Rt ->	MVE
int32x4_t a, int32x4_t b, unsigned * carry_out, mve_pred16_t p)	a -> Qn b -> Qm	VPS1 VSBCIT.I32 Qd,Qn,Qm	*carry_out	
	p -> Rp	VMRS Rt,FPSCR_nzcvqc		
		LSR Rt,#29		
		AND Rt,#1		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vsbciq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, unsigned * carry_out, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSBCIT.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
int32x4_t [_arm_]vsbcq[_s32](int32x4_t a, int32x4_t b, unsigned * carry)	a -> Qn b -> Qm *carry -> Rt	VMRS Rs,FPSCR_nzevqe BFI Rs,Rt,#29,#1 VMSR FPSCR_nzevqe,Rs VSBC.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqe LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
uint32x4_t [_arm_]vsbcq[_u32](uint32x4_t a, uint32x4_t b, unsigned * carry)	a -> Qn b -> Qm *carry -> Rt	VMRS Rs,FPSCR_nzevqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzevqc,Rs VSBC.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
int32x4_t [_arm_]vsbcq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, unsigned * carry, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm *carry -> Rt p -> Rp	VMRS Rs,FPSCR_nzevqe BFI Rs,Rt,#29,#1 VMSR FPSCR_nzevqe,Rs VMSR P0,Rp VPST VSBCT.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqe LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
uint32x4_t [_arm_]vsbcq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, unsigned * carry, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm *carry -> Rt p -> Rp	VMRS Rs,FPSCR_nzcvqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzcvqc,Rs VMSR P0,Rp VPST VSBCT.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
int8x16_t [_arm_]vsubq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VSUB.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vsubq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VSUB.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vsubq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VSUB.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vsubq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VSUB.I8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vsubq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VSUB.I16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [arm_]vsubq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VSUB.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vsubq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VSUB.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vsubq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VSUB.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vsubq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VSUB.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vsubq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VSUB.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vsubq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VSUB.I16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [arm_]vsubq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VSUB.I32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [_arm_]vsubq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VSUB.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]vsubq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VSUB.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [_arm_]vsubq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VSUB.F16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [_arm_]vsubq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VSUB.F32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [_arm_]vsubq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I8 Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vsubq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vsubq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vsubq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.18 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vsubq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [arm_]vsubq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vsubq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.18 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vsubq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vsubq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vsubq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vsubq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vsubq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [arm_]vsubq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vsubq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [arm_]vsubq_m[_n_f16](float16x8_t inactive, float16x8_t a, float16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.F16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [_arm_]vsubq_m[_n_f32](float32x4_t inactive, float32x4_t a, float32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.F32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vsubq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vsubq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p) int32x4 t [ arm ]vsubq x[ s32](int32x4 t a, int32x4 t	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vsubq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p) int8x16_t [arm_]vsubq_x[_n_s8](int8x16_t a, int8_t b,	a -> Qn b -> Qm p -> Rp a -> Qn	VMSR P0,Rp VPST VSUBT.I32 Qd,Qn,Qm VMSR P0,Rp	Qd -> result  Qd -> result	MVE MVE
intox10_t [aim_]vsubq_x[_n_s6](int16x8_t a, int16_t  int16x8_t [arm_]vsubq_x[_n_s16](int16x8_t a, int16_t	b -> Rm p -> Rp a -> Qn	VMSR FU,RP VPST VSUBT.I8 Qd,Qn,Rm VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VSUBT.I16 Qd,Qn,Rm	Za > resuit	

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [arm_]vsubq_x[_n_s32](int32x4_t a, int32_t	a -> On	VMSR P0,Rp	Od -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VSUBT.I32 Qd,Qn,Rm	0.1 >14	MVE
uint8x16_t [arm_]vsubq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
unitox10_t b, inve_preu10_t p)	p -> Rp	VSUBT.I8 Qd,Qn,Qm		
uint16x8_t [arm_]vsubq_x[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
uint32x4_t [arm_]vsubq_x[_u32](uint32x4_t a,	p -> Rp a -> Qn	VSUBT.I16 Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
uint32x4_t [aim_]vsubq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	b -> Om	VPST	Qu -> resun	IVIVE
	p -> Rp	VSUBT.I32 Qd,Qn,Qm		
uint8x16_t [arm_]vsubq_x[_n_u8](uint8x16_t a, uint8_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VSUBT.I8 Qd,Qn,Rm		
uint16x8_t [arm_]vsubq_x[_n_u16](uint16x8_t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
uint16_t b, mve_pred16_t p)	b -> Rm	VPST	Qu'y resuit	111.12
	p -> Rp	VSUBT.I16 Qd,Qn,Rm		
uint32x4_t [arm_]vsubq_x[_n_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VSUBT.I32 Qd,Qn,Rm		
float16x8_t [arm_]vsubq_x[_f16](float16x8_t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST	( - · · · · · · · · · · · · · · · · · ·	
	p -> Rp	VSUBT.F16 Qd,Qn,Qm		
float32x4_t [_arm_]vsubq_x[_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VSUBT.F32 Qd,Qn,Qm		
float16x8 t [ arm ]vsubq x [ n f16](float16x8 t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
float16_t b, mve_pred16_t p)	b -> Rm	VPST		·
	p -> Rp	VSUBT.F16 Qd,Qn,Rm		
float32x4_t [_arm_]vsubq_x[_n_f32](float32x4_t a, float32 t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Qd -> result	MVE
noat32_t b, nive_pred10_t p)	p -> Rn	VSUBT.F32 Qd,Qn,Rm		
float16x8_t [arm_]vcaddq_rot90[_f16](float16x8_t a,	a -> Qn	VCADD.F16 Qd,Qn,Qm,#90	Qd -> result	MVE/NEON
float16x8_t b)	b -> Qm			
float32x4_t [_arm_]vcaddq_rot90[_f32](float32x4_t a,	a -> Qn	VCADD.F32 Qd,Qn,Qm,#90	Qd -> result	MVE/NEON
float32x4_t b) int8x16_t [arm_]vcaddq_rot90[_s8](int8x16_t a,	b -> Qm a -> On	VCADD.I8 Qd,Qn,Qm,#90	Od -> result	MVE
int8x16_t b)	b -> Qm	Veribbilo Qu,Qii,Qiii,ii>0	Qu > resuit	III V E
int16x8_t [arm_]vcaddq_rot90[_s16](int16x8_t a,	a -> Qn	VCADD.I16 Qd,Qn,Qm,#90	Qd -> result	MVE
int16x8_t b)	b -> Qm	VC4 DD 122 0 1 0 0 100	01 1:	NOW
int32x4_t [arm_]vcaddq_rot90[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VCADD.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
uint8x16_t [arm_]vcaddq_rot90[_u8](uint8x16_t a,	a -> On	VCADD.I8 Qd,Qn,Qm,#90	Od -> result	MVE
uint8x16_t b)	b -> Qm		`	
uint16x8_t [arm_]vcaddq_rot90[_u16](uint16x8_t a,	a -> Qn	VCADD.I16 Qd,Qn,Qm,#90	Qd -> result	MVE
uint16x8_t b) uint32x4_t [arm_]vcaddq_rot90[_u32](uint32x4_t a,	b -> Qm a -> On	VCADD.I32 Qd,Qn,Qm,#90	Od -> result	MVE
uint32x4_t [arm_]vcaddq_rot90[_u52](uint32x4_t a, uint32x4_t b)	b -> Qm	VCADD.132 Qd,Qli,Qlii,#90	Qu -> resuit	MVE
float16x8_t [arm_]vcaddq_rot270[_f16](float16x8_t a,	a -> Qn	VCADD.F16 Qd,Qn,Qm,#270	Qd -> result	MVE/NEON
float16x8_t b)	b -> Qm			
float32x4_t [_arm_]vcaddq_rot270[_f32](float32x4_t a,	a -> Qn	VCADD.F32 Qd,Qn,Qm,#270	Qd -> result	MVE/NEON
float32x4_t b) int8x16_t [arm_]vcaddq_rot270[_s8](int8x16_t a,	b -> Qm a -> Qn	VCADD.I8 Qd,Qn,Qm,#270	Od -> result	MVE
int8x16 t b)	b -> Qm	VCADD.18 Qu,Qii,Qiii,#270	Qu -> resuit	IVI V E
int16x8_t [arm_]vcaddq_rot270[_s16](int16x8_t a,	a -> Qn	VCADD.I16 Qd,Qn,Qm,#270	Qd -> result	MVE
int16x8_t b)	b -> Qm			
int32x4_t [arm_]vcaddq_rot270[_s32](int32x4_t a,	a -> Qn	VCADD.I32 Qd,Qn,Qm,#270	Qd -> result	MVE
int32x4_t b) uint8x16_t [arm_]vcaddq_rot270[_u8](uint8x16_t a,	b -> Qm a -> Qn	VCADD.I8 Qd,Qn,Qm,#270	Qd -> result	MVE
uint8x16_t b)	b -> Qm	Veribb.io Qu,Qii,Qiii,#270	Qu -> result	WYL
uint16x8_t [_arm_]vcaddq_rot270[_u16](uint16x8_t a,	a -> Qn	VCADD.I16 Qd,Qn,Qm,#270	Qd -> result	MVE
uint16x8_t b)	b -> Qm		1	1.00
uint32x4_t [_arm_]vcaddq_rot270[_u32](uint32x4_t a,	a -> Qn	VCADD.I32 Qd,Qn,Qm,#270	Qd -> result	MVE
uint32x4_t b) float16x8_t [_arm_]vcaddq_rot90_m[_f16](float16x8_t	b -> Qm inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn	VPST VPST	Qu > resuit	
	b -> Qm	VCADDT.F16 Qd,Qn,Qm,#90	1	
0.004.6	p -> Rp	VIMOR PO P	01	Marie
float32x4_t [arm_]vcaddq_rot90_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
macave, noac32x+_t a, noac32x+_t b, mvc_prearo_t p)	b -> Qm	VCADDT.F32 Qd,Qn,Qm,#90	1	
	p -> Rp			

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vcaddq_rot90_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
int16x8_t [arm_]vcaddq_rot90_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I16 Qd,Qn,Qm,#90	Qd -> result	MVE
int32x4_t [arm_]vcaddq_rot90_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
uint8x16_t [_arm_]vcaddq_rot90_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
uint16x8_t [arm_]vcaddq_rot90_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I16 Qd,Qn,Qm,#90	Qd -> result	MVE
uint32x4_t [arm_]vcaddq_rot90_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
float16x8_t [arm_]vcaddq_rot270_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.F16 Qd,Qn,Qm,#270	Qd -> result	MVE
float32x4_t [arm_]vcaddq_rot270_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.F32 Qd,Qn,Qm,#270	Qd -> result	MVE
int8x16_t [arm_]vcaddq_rot270_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I8 Qd,Qn,Qm,#270	Qd -> result	MVE
int16x8_t [arm_]vcaddq_rot270_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I16 Qd,Qn,Qm,#270	Qd -> result	MVE
int32x4_t [arm_]vcaddq_rot270_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#270	Qd -> result	MVE
uint8x16_t [_arm_]vcaddq_rot270_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I8 Qd,Qn,Qm,#270	Qd -> result	MVE
uint16x8_t [_arm_]vcaddq_rot270_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I16 Qd,Qn,Qm,#270	Qd -> result	MVE
uint32x4_t [_arm_]vcaddq_rot270_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#270	Qd -> result	MVE
float16x8_t [arm_]vcaddq_rot90_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.F16 Qd,Qn,Qm,#90	Qd -> result	MVE
float32x4_t [_arm_]vcaddq_rot90_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.F32 Qd,Qn,Qm,#90	Qd -> result	MVE
int8x16_t [_arm_]vcaddq_rot90_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
int16x8_t [_arm_]vcaddq_rot90_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I16 Qd,Qn,Qm,#90	Qd -> result	MVE
int32x4_t [arm_]vcaddq_rot90_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
uint8x16_t [arm_]vcaddq_rot90_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
uint16x8_t [_arm_]vcaddq_rot90_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I16 Qd,Qn,Qm,#90	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vcaddq_rot90_x[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Od -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Q	1
	p -> Rp	VCADDT.I32 Qd,Qn,Qm,#90		
float16x8_t [_arm_]vcaddq_rot270_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
a, noatroxo_t b, nive_predio_t p)	p -> Rp	VCADDT.F16 Qd,Qn,Qm,#270		
float32x4_t [arm_]vcaddq_rot270_x[_f32](float32x4_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
a, float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCADDT.F32 Qd,Qn,Qm,#270		
int8x16_t [arm_]vcaddq_rot270_x[_s8](int8x16_t a,	a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
int8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VCADDT.I8 Qd,Qn,Qm,#270		
int16x8_t [arm_]vcaddq_rot270_x[_s16](int16x8_t a,	a -> Qn	VMSR P0,Rp	Od -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCADDT.I16 Qd,Qn,Qm,#270		
int32x4_t [_arm_]vcaddq_rot270_x[_s32](int32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCADDT.I32 Qd,Qn,Qm,#270		
uint8x16_t [arm_]vcaddq_rot270_x[_u8](uint8x16_t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST	Qu > resuit	III V E
	p -> Rp	VCADDT.I8 Qd,Qn,Qm,#270		
uint16x8_t [arm_]vcaddq_rot270_x[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
nint22v4 + [ arm ]voodda_rat270_v[ n22](nint22v4 + a	p -> Rp a -> Qn	VCADDT.I16 Qd,Qn,Qm,#270 VMSR P0,Rp	Qd -> result	MVE
uint32x4_t [_arm_]vcaddq_rot270_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	b -> Qm	VMSK PO,KP VPST	Qu -> resuit	MIVE
unito 2x 1_t o, in to_prod to_t p)	p -> Rp	VCADDT.I32 Qd,Qn,Qm,#270		
float16x8_t [arm_]vcmlaq[_f16](float16x8_t a,	a -> Qda	VCMLA.F16 Qda,Qn,Qm,#0	Qda -> result	MVE/NEON
float16x8_t b, float16x8_t c)	b -> Qn			
G - 22 4 - 1 - 1 - 1 - 1 - 1 - 1 - 22 - 22 - 1 - 22 - 22 - 1 - 22 - 22 - 1 - 22 - 22 - 1 - 22 - 1 - 22 - 1 - 22 - 1 - 22 - 1 - 22 - 1 - 22 - 1 - 22	c -> Qm	1/07/07 4 7722 01 0 0 0 110	0.1	) (I I A I E O V
float32x4_t [arm_]vcmlaq[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn	VCMLA.F32 Qda,Qn,Qm,#0	Qda -> result	MVE/NEON
Hoat32x4_t b, Hoat32x4_t c)	c -> Qm			
float16x8_t [arm_]vcmlaq_rot90[_f16](float16x8_t a,	a -> Qda	VCMLA.F16 Qda,Qn,Qm,#90	Qda -> result	MVE/NEON
float16x8_t b, float16x8_t c)	b -> Qn			
	c -> Qm			
float32x4_t [_arm_]vcmlaq_rot90[_f32](float32x4_t a,	a -> Qda	VCMLA.F32 Qda,Qn,Qm,#90	Qda -> result	MVE/NEON
float32x4_t b, float32x4_t c)	b -> Qn c -> Qm			
float16x8_t [arm_]vcmlaq_rot180[_f16](float16x8_t a,	a -> Qda	VCMLA.F16 Qda,Qn,Qm,#180	Qda -> result	MVE/NEON
float16x8_t b, float16x8_t c)	b -> Qn	V CIVILI II 10 Qua, QII, QIII, II 100	Qua > resurt	W V E/ I V EO I V
	c -> Qm			
float32x4_t [arm_]vcmlaq_rot180[_f32](float32x4_t a,	a -> Qda	VCMLA.F32 Qda,Qn,Qm,#180	Qda -> result	MVE/NEON
float32x4_t b, float32x4_t c)	b -> Qn			
float16x8 t [ arm ]vcmlaq rot270[ f16](float16x8 t a,	c -> Qm a -> Qda	VCMLA.F16 Qda,Qn,Qm,#270	Oda -> result	MVE/NEON
float16x8_t b, float16x8_t c)	b -> Qua	VCWLA.I 10 Qua,Qii,Qiii,#270	Qua -> resuit	WIVE/INDOIN
induitorio_t o, induitorio_t o)	c -> Qm			
float32x4_t [arm_]vcmlaq_rot270[_f32](float32x4_t a,	a -> Qda	VCMLA.F32 Qda,Qn,Qm,#270	Qda -> result	MVE/NEON
float32x4_t b, float32x4_t c)	b -> Qn			
fl416-9 4 [ ]1 f161/fl416-9 4 -	c -> Qm	VMCD DO D.	0.1	MVE
float16x8_t [_arm_]vcmlaq_m[_f16](float16x8_t a, float16x8_t b, float16x8_t c, mve_pred16_t p)	a -> Qda b -> Qn	VMSR P0,Rp VPST	Qda -> result	MVE
noutono_t o, noutrono_t e, nive_preuro_t p)	c -> Qm	VCMLAT.F16 Qda,Qn,Qm,#0		
	p -> Rp			
float32x4_t [arm_]vcmlaq_m[_f32](float32x4_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
float32x4_t b, float32x4_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Qm p -> Rp	VCMLAT.F32 Qda,Qn,Qm,#0		
float16x8 t [ arm ]vcmlaq rot90 m[ f16](float16x8 t	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
a, float16x8_t b, float16x8_t c, mve_pred16_t p)	b -> Qua	VPST	Zan > icani	
	c -> Qm	VCMLAT.F16 Qda,Qn,Qm,#90		1
	p -> Rp			<u> </u>
float32x4_t [_arm_]vcmlaq_rot90_m[_f32](float32x4_t	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
a, float32x4_t b, float32x4_t c, mve_pred16_t p)	b -> Qn c -> Qm	VPST VCMLAT.F32 Qda,Qn,Qm,#90		1
	p -> Rp	V CIVILA 1.132 Qua,Qii,Qiii,#90		
float16x8_t [arm_]vcmlaq_rot180_m[_f16](float16x8_t	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
a, float16x8_t b, float16x8_t c, mve_pred16_t p)	b -> Qn	VPST		1
-	c -> Qm	VCMLAT.F16 Qda,Qn,Qm,#180		1
	p -> Rp			1
float32x4_t [_arm_]vcmlaq_rot180_m[_f32](float32x4_t	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
a, float32x4_t b, float32x4_t c, mve_pred16_t p)	b -> Qn c -> Qm	VPST VCMLAT.F32 Qda,Qn,Qm,#180		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [arm_]vcmlaq_rot270_m[_f16](float16x8_t	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
a, float16x8_t b, float16x8_t c, mve_pred16_t p)	b -> Qn c -> Qm	VPST VCMLAT.F16 Qda,Qn,Qm,#270		
float32x4_t [_arm_]vcmlaq_rot270_m[_f32](float32x4_t	p -> Rp a -> Oda	VMSR P0,Rp	Qda -> result	MVE
a, float32x4_t b, float32x4_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Qm	VCMLAT.F32 Qda,Qn,Qm,#270		
float16x8 t [ arm ]vcmulq[ f16](float16x8 t a,	p -> Rp a -> On	VCMUL.F16 Qd,Qn,Qm,#0	Od -> result	MVE
float16x8_t b)	b -> Qm	2,72,72	<u> </u>	·
float32x4_t [arm_]vcmulq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMUL.F32 Qd,Qn,Qm,#0	Qd -> result	MVE
float16x8_t [arm_]vcmulq_rot90[_f16](float16x8_t a,	a -> Qn	VCMUL.F16 Qd,Qn,Qm,#90	Qd -> result	MVE
float16x8_t b) float32x4_t [arm_]vcmulq_rot90[_f32](float32x4_t a,	b -> Qm a -> Qn	VCMUL.F32 Qd,Qn,Qm,#90	Qd -> result	MVE
float32x4_t b)	b -> Qm	2 / 2 / 2 /		
float16x8_t [arm_]vcmulq_rot180[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMUL.F16 Qd,Qn,Qm,#180	Qd -> result	MVE
float32x4_t [_arm_]vcmulq_rot180[_f32](float32x4_t a,	a -> Qn	VCMUL.F32 Qd,Qn,Qm,#180	Qd -> result	MVE
float32x4_t b) float16x8_t [arm_]vcmulq_rot270[_f16](float16x8_t a,	b -> Qm a -> Qn	VCMUL.F16 Qd,Qn,Qm,#270	Od -> result	MVE
float16x8_t b)	b->Qm	VG1477 F22 0 1 0 0 1070	21 1	NAME.
float32x4_t [arm_]vcmulq_rot270[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMUL.F32 Qd,Qn,Qm,#270	Qd -> result	MVE
float16x8_t [_arm_]vcmulq_m[_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VCMULT.F16 Qd,Qn,Qm,#0		
float32x4 t [ arm ]vcmulg m[ f32](float32x4 t	p -> Rp inactive -> Od	VACD DO D	0.1	MVE
inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
, , , ,	b -> Qm	VCMULT.F32 Qd,Qn,Qm,#0		
float16x8_t [arm_]vcmulq_rot90_m[_f16](float16x8_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VCMULT.F16 Qd,Qn,Qm,#90		
float32x4_t [arm_]vcmulq_rot90_m[_f32](float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VCMULT.F32 Qd,Qn,Qm,#90		
	p -> Rp			
float16x8_t [_arm_]vcmulq_rot180_m[_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VCMULT.F16 Qd,Qn,Qm,#180		
G (22.4 ) 1 (100   F (22.4 ) (22.4 )	p -> Rp	VA (CD DO D	01 . 1	MATE
float32x4_t [arm_]vcmulq_rot180_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qm	VCMULT.F32 Qd,Qn,Qm,#180		
float16x8_t [_arm_]vcmulq_rot270_m[_f16](float16x8_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VCMULT.F16 Qd,Qn,Qm,#270		
$float32x4\_t \ [\_arm\_]vcmulq\_rot270\_m[\_f32](float32x4\_t$	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VCMULT.F32 Qd,Qn,Qm,#270		
	p -> Rp			
float16x8_t [arm_]vcmulq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
_ · · _ · _ · ·	p -> Rp	VCMULT.F16 Qd,Qn,Qm,#0		
float32x4_t [arm_]vcmulq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
· · - · · · · · · · · · · · · ·	p -> Rp	VCMULT.F32 Qd,Qn,Qm,#0		
float16x8_t [arm_]vcmulq_rot90_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Qiii p -> Rp	VCMULT.F16 Qd,Qn,Qm,#90		
float32x4_t [arm_]vcmulq_rot90_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
noat32x+_t t, nive_predio_t p)	p -> QIII p -> Rp	VCMULT.F32 Qd,Qn,Qm,#90		
float16x8_t [_arm_]vcmulq_rot180_x[_f16](float16x8_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
a, float16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCMULT.F16 Qd,Qn,Qm,#180		
float32x4_t [_arm_]vcmulq_rot180_x[_f32](float32x4_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
a, float32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCMULT.F32 Qd,Qn,Qm,#180		
float16x8_t [arm_]vcmulq_rot270_x[_f16](float16x8_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
a, float16x8_t b, mve_pred16_t p)	b -> Qm	VPST		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [arm_]vcmulq_rot270_x[_f32](float32x4_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
a, float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMULT.F32 Qd,Qn,Qm,#270		
int8x16_t [_arm_]vqabsq[_s8](int8x16_t a)	a -> Qm	VQABS.S8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vqabsq[_s16](int16x8_t a) int32x4_t [arm_]vqabsq[_s32](int32x4_t a)	a -> Qm a -> Qm	VQABS.S16 Qd,Qm VQABS.S32 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
int8x16_t [arm_]vqabsq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Om	VPST	Qu -> result	WIVE
	p -> Rp	VQABST.S8 Qd,Qm		
int16x8_t [arm_]vqabsq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
int32x4 t [ arm ]vqabsq m[ s32](int32x4 t inactive,	p -> Rp	VQABST.S16 Qd,Qm	0.1	MVE
int32x4_t iarm_jvqaosq_m[_s52](int52x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
miszki-t a, mio_proato_t p)	p -> Rp	VQABST.S32 Qd,Qm		
int8x16_t [arm_]vqaddq[_n_s8](int8x16_t a, int8_t b)	a -> Qn	VQADD.S8 Qd,Qn,Rm	Qd -> result	MVE
	b -> Rm		`	
int16x8_t [arm_]vqaddq[_n_s16](int16x8_t a, int16_t b)	a -> Qn	VQADD.S16 Qd,Qn,Rm	Qd -> result	MVE
	b -> Rm			
int32x4_t [arm_]vqaddq[_n_s32](int32x4_t a, int32_t b)	a -> Qn	VQADD.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vqaddq[_n_u8](uint8x16_t a, uint8_t	b -> Rm a -> On	VQADD.U8 Qd,Qn,Rm	Od -> result	MVE
b)	b -> Rm	VQADD.U8 Qd,Qii,Riii	Qu -> resuit	MVE
uint16x8_t [arm_]vqaddq[_n_u16](uint16x8_t a,	a -> On	VQADD.U16 Qd,Qn,Rm	Od -> result	MVE
uint16_t b)	b -> Rm	, d.199.610 da',dii'itiii	Qu > result	14142
uint32x4_t [arm_]vqaddq[_n_u32](uint32x4_t a,	a -> Qn	VQADD.U32 Qd,Qn,Rm	Qd -> result	MVE
uint32_t b)	b -> Rm			
int8x16_t [arm_]vqaddq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VQADD.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
1.460.460.460.460.41	b -> Qm	VOLDE GLOOD	0.1	) (I TE AVECAV
int16x8_t [arm_]vqaddq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VQADD.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vqaddq[_s32](int32x4_t a, int32x4_t b)	b -> Qm a -> On	VQADD.S32 Qd,Qn,Qm	Od -> result	MVE/NEON
int32x4_t [atm_]vqaddq[_832](int32x4_t a, int32x4_t b)	b -> Qm	VQADD.332 Qd,Qll,Qlll	Qu -> resuit	IVI V E/INEOIN
uint8x16_t [arm_]vqaddq[_u8](uint8x16_t a, uint8x16_t	a -> On	VQADD.U8 Qd,Qn,Qm	Od -> result	MVE/NEON
b)	b -> Qm			
uint16x8_t [arm_]vqaddq[_u16](uint16x8_t a,	a -> Qn	VQADD.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t b)	b -> Qm			
uint32x4_t [arm_]vqaddq[_u32](uint32x4_t a,	a -> Qn	VQADD.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t b)	b -> Qm	LIMED DO D	0.1 1:	MATE
int8x16_t [arm_]vqaddq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
intox10_t a, into_t b, inve_picu10_t p)	b -> Rm	VQADDT.S8 Qd,Qn,Rm		
	p -> Rp	, 4.1221.50 44,41,1111		
int16x8_t [arm_]vqaddq_m[_n_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm	VQADDT.S16 Qd,Qn,Rm		
int22v4 t [ amm lygodda mf n c221/int22v4 t incetive	p -> Rp	VMSR P0,Rp	Od -> result	MVE
int32x4_t [arm_]vqaddq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSK PO,KP VPST	Qu -> resuit	MVE
int32x4_t a, int32_t b, inve_pred10_t p)	b -> Rm	VQADDT.S32 Qd,Qn,Rm		
	p -> Rp			
uint8x16_t [arm_]vqaddq_m[_n_u8](uint8x16_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm	VQADDT.U8 Qd,Qn,Rm		1
uint16x8_t [arm_]vqaddq_m[_n_u16](uint16x8_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn	VMSK PO,KP VPST	Qu -> resuit	MVE
mactive, unitroxo_t a, unitro_t o, mve_preuro_t p)	b -> Rm	VQADDT.U16 Qd,Qn,Rm		
	p -> Rp	(1)	<u> </u>	
uint32x4_t [arm_]vqaddq_m[_n_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm	VQADDT.U32 Qd,Qn,Rm		
int8x16_t [arm_]vqaddq_m[_s8](int8x16_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t iarm_jvqaddq_mi_soj(int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> On	VMSK FU,KP VPST	Qu -> Iesuit	IVI V IS
	b -> Qm	VQADDT.S8 Qd,Qn,Qm		
	p -> Rp	7 7 7 7		
int16x8_t [arm_]vqaddq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VQADDT.S16 Qd,Qn,Qm		
int32x4_t [_arm_]vqaddq_m[_s32](int32x4_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t [armjvqaddq_m[_s32](mt32x4_t mactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn	VMSK PO,KP VPST	Qu -> resuit	1V1 V ID
,,,	b -> Qm	VQADDT.S32 Qd,Qn,Qm		1
	p -> Rp	1	1	1

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [_arm_]vqaddq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQADDT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vqaddq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQADDT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vqaddq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQADDT.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vqdmladhq[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b)	inactive -> Qd a -> Qn b -> Qm	VQDMLADH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vqdmladhq[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b)	inactive -> Qd a -> Qn b -> Qm	VQDMLADH.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4\_t \ [\_arm\_]vqdmladhq [\_s32] (int32x4\_t \ inactive, \\ int32x4\_t \ a, int32x4\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLADH.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqdmladhq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLADHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqdmladhq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLADHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vqdmladhq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLADHT.S32 Qd,Qn,Qm	Qd -> result	MVE
$int8x16\_t \ [\_arm\_]vqdmladhxq[\_s8] (int8x16\_t \ inactive, \\ int8x16\_t \ a, int8x16\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLADHX.S8 Qd,Qn,Qm	Qd -> result	MVE
$int16x8\_t \ [\_arm\_]vqdmladhxq[\_s16] (int16x8\_t \ inactive, \\ int16x8\_t \ a, int16x8\_t \ b)$	inactive -> Qd a -> Qn b -> Om	VQDMLADHX.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4\_t \ [\_arm\_]vqdmladhxq[\_s32](int32x4\_t \ inactive, \\ int32x4\_t \ a, int32x4\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLADHX.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vqdmladhxq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLADHXT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vqdmladhxq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLADHXT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vqdmladhxq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLADHXT.S32 Qd,Qn,Qm	Qd -> result	MVE
$int8x16\_t \ [\_arm\_]vqrdmladhq[\_s8] (int8x16\_t \ inactive, \\ int8x16\_t \ a, \ int8x16\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLADH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vqrdmladhq[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b)	inactive -> Qd a -> Qn b -> Qm	VQRDMLADH.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4\_t \ [\_arm\_]vqrdmladhq[\_s32](int32x4\_t \ inactive, \\ int32x4\_t \ a, int32x4\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLADH.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vqrdmladhq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLADHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vqrdmladhq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLADHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vqrdmladhq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLADHT.S32 Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vqrdmladhxq[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b)	inactive -> Qd a -> Qn	VQRDMLADHX.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqrdmladhxq[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b)	b -> Qm inactive -> Qd a -> Qn b -> Om	VQRDMLADHX.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4\_t \ [\_arm\_]vqrdmladhxq[\_s32](int32x4\_t \ inactive, \\ int32x4\_t \ a, \ int32x4\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLADHX.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqrdmladhxq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLADHXT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqrdmladhxq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLADHXT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vqrdmladhxq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLADHXT.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqdmlahq[_n_s8](int8x16_t a, int8x16_t b, int8_t c)	a -> Qda b -> Qn c -> Rm	VQDMLAH.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vqdmlahq[_n_s16](int16x8_t a, int16x8_t b, int16_t c)	a -> Qda b -> Qn c -> Rm	VQDMLAH.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vqdmlahq[_n_s32](int32x4_t a, int32x4_t b, int32_t c)	a -> Qda b -> Qn c -> Rm	VQDMLAH.S32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [arm_]vqdmlahq_m[_n_s8](int8x16_t a, int8x16_t b, int8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLAHT.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vqdmlahq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLAHT.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [armlvqdmlahq_m[_n_s32](int32x4_t a, int32x4_t b, int32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm	VMSR P0,Rp VPST VQDMLAHT.S32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [arm_]vqrdmlahq[_n_s8](int8x16_t a, int8x16_t b, int8_t c)	p -> Rp a -> Qda b -> Qn c -> Rm	VQRDMLAH.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vqrdmlahq[_n_s16](int16x8_t a, int16x8_t b, int16_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLAH.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vqrdmlahq[_n_s32](int32x4_t a, int32x4_t b, int32_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLAH.S32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [arm_]vqrdmlahq_m[_n_s8](int8x16_t a, int8x16_t b, int8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLAHT.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vqrdmlahq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLAHT.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vqrdmlahq_m[_n_s32](int32x4_t a, int32x4_t b, int32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLAHT.S32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [arm_]vqdmlashq[_n_s8](int8x16_t a, int8x16_t b, int8_t c)	a -> Qda b -> Qn c -> Rm	VQDMLASH.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vqdmlashq[_n_s16](int16x8_t a, int16x8_t b, int16_t c)	a -> Qda b -> Qn c -> Rm	VQDMLASH.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vqdmlashq[_n_s32](int32x4_t a, int32x4_t b, int32_t c)	a -> Qda b -> Qn	VQDMLASH.S32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [arm_]vqdmlashq_m[_n_s8](int8x16_t a, int8x16_t b, int8_t c, mve_pred16_t p)	c -> Rm a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLASHT.S8 Qda,Qn,Rm	Qda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vqdmlashq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLASHT.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vqdmlashq_m[_n_s32](int32x4_t a, int32x4_t b, int32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLASHT.S32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [arm_]vqrdmlashq[_n_s8](int8x16_t a, int8x16_t b, int8_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLASH.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vqrdmlashq[_n_s16](int16x8_t a, int16x8_t b, int16_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLASH.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vqrdmlashq[_n_s32](int32x4_t a, int32x4_t b, int32_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLASH.S32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [arm_]vqrdmlashq_m[_n_s8](int8x16_t a, int8x16_t b, int8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLASHT.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vqrdmlashq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLASHT.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vqrdmlashq_m[_n_s32](int32x4_t a, int32x4_t b, int32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLASHT.S32 Qda,Qn,Rm	Qda -> result	MVE
$int8x16\_t \ [\_arm\_]vqdmlsdhq[\_s8](int8x16\_t \ inactive, \\ int8x16\_t \ a, int8x16\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLSDH.S8 Qd,Qn,Qm	Qd -> result	MVE
$int16x8\_t \ [\_arm\_]vqdmlsdhq[\_s16](int16x8\_t \ inactive, \\ int16x8\_t \ a, int16x8\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLSDH.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4\_t \ [\_arm\_]vqdmlsdhq[\_s32](int32x4\_t \ inactive, \\ int32x4\_t \ a, int32x4\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLSDH.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqdmlsdhq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqdmlsdhq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vqdmlsdhq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHT.S32 Qd,Qn,Qm	Qd -> result	MVE
$int8x16\_t \ [\_arm\_]vqdmlsdhxq[\_s8](int8x16\_t \ inactive, \\ int8x16\_t \ a, int8x16\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLSDHX.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqdmlsdhxq[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b)	inactive -> Qd a -> Qn b -> Qm	VQDMLSDHX.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4\_t \ [\_arm\_]vqdmlsdhxq[\_s32](int32x4\_t \ inactive, \\ int32x4\_t \ a, int32x4\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLSDHX.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqdmlsdhxq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHXT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqdmlsdhxq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHXT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vqdmlsdhxq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHXT.S32 Qd,Qn,Qm	Qd -> result	MVE
$int8x16\_t \ [\_arm\_]vqrdmlsdhq[\_s8](int8x16\_t \ inactive, \\ int8x16\_t \ a, int8x16\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqrdmlsdhq[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b)	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDH.S16 Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
$int32x4\_t \ [\_arm\_]vqrdmlsdhq[\_s32](int32x4\_t \ inactive, \\ int32x4\_t \ a, int32x4\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDH.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqrdmlsdhq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLSDHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqrdmlsdhq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLSDHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vqrdmlsdhq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLSDHT.S32 Qd,Qn,Qm	Qd -> result	MVE
$int8x16\_t \ [\_arm\_]vqrdmlsdhxq[\_s8](int8x16\_t \ inactive, \\ int8x16\_t \ a, int8x16\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDHX.S8 Qd,Qn,Qm	Qd -> result	MVE
$int16x8\_t \ [\_arm\_]vqrdmlsdhxq[\_s16](int16x8\_t \ inactive, \\ int16x8\_t \ a, int16x8\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDHX.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4\_t \ [\_arm\_]vqrdmlsdhxq[\_s32](int32x4\_t \ inactive, \\ int32x4\_t \ a, int32x4\_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDHX.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqrdmlsdhxq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLSDHXT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqrdmlsdhxq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLSDHXT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vqrdmlsdhxq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLSDHXT.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vqdmulhq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VQDMULH.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vqdmulhq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQDMULH.S16 Qd,Qn,Rm	Qd -> result	MVE/NEON
int32x4_t [arm_]vqdmulhq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VQDMULH.S32 Qd,Qn,Rm	Qd -> result	MVE/NEON
int8x16_t [arm_]vqdmulhq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULHT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vqdmulhq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULHT.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [arm_]vqdmulhq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULHT.S32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vqdmulhq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VQDMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vqdmulhq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQDMULH.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vqdmulhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQDMULH.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vqdmulhq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VQDMULHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqdmulhq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMULHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vqdmulhq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMULHT.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqrdmulhq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VQRDMULH.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vqrdmulhq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQRDMULH.S16 Qd,Qn,Rm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [_arm_]vqrdmulhq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VQRDMULH.S32 Qd,Qn,Rm	Qd -> result	MVE/NEON
int8x16_t [arm_]vqrdmulhq_m[_n_s8](int8x16_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VQRDMULHT.S8 Qd,Qn,Rm		
	p -> Rp	VQKDWOEITI.58 Qd,Qii,Kiii		
int16x8_t [arm_]vqrdmulhq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
macuve, meroxo_t a, mero_t b, mve_preuro_t p)	b -> Rm	VQRDMULHT.S16 Qd,Qn,Rm		
int32x4_t [arm_]vqrdmulhq_m[_n_s32](int32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn	VPST	Qu -> resuit	MVL
	b -> Rm p -> Rp	VQRDMULHT.S32 Qd,Qn,Rm		
int8x16_t [arm_]vqrdmulhq[_s8](int8x16_t a, int8x16_t	a -> Qn	VQRDMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
b) int16x8_t [arm_]vqrdmulhq[_s16](int16x8_t a,	b -> Qm a -> On	VQRDMULH.S16 Qd,Qn,Qm	Od -> result	MVE/NEON
int16x8_t b)	b->Qm		`	
int32x4_t [arm_]vqrdmulhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQRDMULH.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vqrdmulhq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VQRDMULHT.S8 Qd,Qn,Qm		
	p -> Rp	2		
int16x8_t [_arm_]vqrdmulhq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, introxo_t a, introxo_t b, inve_predio_t p	b -> Qm	VQRDMULHT.S16 Qd,Qn,Qm		
int32x4_t [arm_]vgrdmulhq_m[_s32](int32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn	VPST	Qu > result	11112
	b -> Qm p -> Rp	VQRDMULHT.S32 Qd,Qn,Qm		
int32x4_t [_arm_]vqdmullbq[_n_s16](int16x8_t a,	a -> Qn	VQDMULLB.S16 Qd,Qn,Rm	Qd -> result	MVE
int16_t b) int64x2_t [arm_]vqdmullbq[_n_s32](int32x4_t a,	b -> Rm a -> On	VQDMULLB.S32 Qd,Qn,Rm	Od -> result	MVE
int32_t b)	b->Rm	7 7 7 7		
int32x4_t [arm_]vqdmullbq_m[_n_s16](int32x4_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Rm	VQDMULLBT.S16 Qd,Qn,Rm		
int64x2_t [arm_]vqdmullbq_m[_n_s32](int64x2_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm p -> Rp	VQDMULLBT.S32 Qd,Qn,Rm		
int32x4_t [_arm_]vqdmullbq[_s16](int16x8_t a,	a -> Qn b -> Om	VQDMULLB.S16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t b) int64x2_t [arm_]vqdmullbq[_s32](int32x4_t a,	a -> Qn	VQDMULLB.S32 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t b) int32x4_t [_arm_]vqdmullbq_m[_s16](int32x4_t	b -> Qm inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST	Qu -> resuit	MVL
	b -> Qm p -> Rp	VQDMULLBT.S16 Qd,Qn,Qm		
int64x2_t [arm_]vqdmullbq_m[_s32](int64x2_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VODMULLBT.S32 Od,On,Om		
	p -> Rp	2,72,72		
int32x4_t [arm_]vqdmulltq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQDMULLT.S16 Qd,Qn,Rm	Qd -> result	MVE
int64x2_t [arm_]vqdmulltq[_n_s32](int32x4_t a, int32_t	a -> Qn	VQDMULLT.S32 Qd,Qn,Rm	Qd -> result	MVE
b) int32x4_t [arm_]vqdmulltq_m[_n_s16](int32x4_t	b -> Rm inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn	VPST	(2 / 22	
	b -> Rm p -> Rp	VQDMULLTT.S16 Qd,Qn,Rm		
int64x2_t [arm_]vqdmulltq_m[_n_s32](int64x2_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VQDMULLTT.S32 Qd,Qn,Rm		
100204 AT 1000 hadronllast 1000 de 0 1 1000 de	p -> Rp	2 . 2 .	01 2 1	MVE
int32x4_t [_arm_]vqdmulltq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQDMULLT.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [arm_]vqdmulltq[_s32](int32x4_t a, int32x4_t	a -> Qn	VQDMULLT.S32 Qd,Qn,Qm	Qd -> result	MVE
b) int32x4_t [arm_]vqdmulltq_m[_s16](int32x4_t inactive,	b -> Qm inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VQDMULLTT.S16 Qd,Qn,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64x2_t [_arm_]vqdmulltq_m[_s32](int64x2_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VQDMULLTT.S32 Qd,Qn,Qm		
	p -> Rp	V Q D W C D E 1 1.552 Qu, Q II, Q III		
int8x16_t [arm_]vqnegq[_s8](int8x16_t a)	a -> Qm	VQNEG.S8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vqnegq[_s16](int16x8_t a) int32x4_t [arm_]vqnegq[_s32](int32x4_t a)	a -> Qm a -> Qm	VQNEG.S16 Qd,Qm VQNEG.S32 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
int8x16_t [arm_]vqnegq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
int16x8_t [arm_]vqnegq_m[_s16](int16x8_t inactive,	p -> Rp inactive -> Qd	VQNEGT.S8 Qd,Qm VMSR P0,Rp	Od -> result	MVE
int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VQNEGT.S16 Qd,Qm	Qu > resuit	IVI V E
int32x4_t [arm_]vqnegq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VQNEGT.S32 Qd,Qm		
int8x16_t [arm_]vqsubq[_n_s8](int8x16_t a, int8_t b)	a -> Qn	VQSUB.S8 Qd,Qn,Rm	Qd -> result	MVE
	b -> Rm		1	
int16x8_t [arm_]vqsubq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQSUB.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vqsubq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VQSUB.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vqsubq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VQSUB.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [arm_]vqsubq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VQSUB.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [arm_]vqsubq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VQSUB.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vqsubq_m[_n_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VQSUBT.S8 Qd,Qn,Rm		
int16x8_t [arm_]vqsubq_m[_n_s16](int16x8_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn	VPST VPST	Qu -> resuit	WIVE
	b -> Rm p -> Rp	VQSUBT.S16 Qd,Qn,Rm		
int32x4_t [arm_]vqsubq_m[_n_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VPST VQSUBT.S32 Qd,Qn,Rm		
uint8x16_t [arm_]vqsubq_m[_n_u8](uint8x16_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VQSUBT.U8 Qd,Qn,Rm		
uint16x8 t [ arm ]vqsubq m[ n u16](uint16x8 t	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn	VPST	Qu -> resurt	WIVE
2	b -> Rm	VQSUBT.U16 Qd,Qn,Rm		
uint32x4_t [arm_]vqsubq_m[_n_u32](uint32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn	VPST	Qu > result	W. C.
	b -> Rm p -> Rp	VQSUBT.U32 Qd,Qn,Rm		
int8x16_t [_arm_]vqsubq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VQSUB.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vqsubq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQSUB.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vqsubq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQSUB.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vqsubq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VQSUB.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vqsubq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VQSUB.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vqsubq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VQSUB.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vqsubq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VQSUBT.S8 Qd,Qn,Qm		
int16x8_t [arm_]vqsubq_m[_s16](int16x8_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VQSUBT.S16 Qd,Qn,Qm		
int32x4_t [arm_]vqsubq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
m.52x+_t a, m.52x+_t b, mvc_preuto_t p)	b -> Qm	VQSUBT.S32 Qd,Qn,Qm		
	p -> Rp			

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [arm_]vqsubq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQSUBT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vqsubq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQSUBT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vqsubq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQSUBT.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16x2_t [_arm_]vld2q[_s8](int8_t const * addr)	addr -> Rn	VLD20.8 {Qd - Qd2},[Rn] VLD21.8 {Qd - Qd2},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1]	MVE
int16x8x2_t [_arm_]vld2q[_s16](int16_t const * addr)	addr -> Rn	VLD20.16 {Qd - Qd2},[Rn] VLD21.16 {Qd - Qd2},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1]	MVE
int32x4x2_t [_arm_]vld2q[_s32](int32_t const * addr)	addr -> Rn	VLD20.32 {Qd - Qd2},[Rn] VLD21.32 {Qd - Qd2},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1]	MVE
uint8x16x2_t [arm_]vld2q[_u8](uint8_t const * addr)	addr -> Rn	VLD20.8 {Qd - Qd2},[Rn] VLD21.8 {Qd - Qd2},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1]	MVE
uint16x8x2_t [arm_]vld2q[_u16](uint16_t const * addr)	addr -> Rn	VLD20.16 {Qd - Qd2},[Rn] VLD21.16 {Qd - Qd2},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1]	MVE
uint32x4x2_t [_arm_]vld2q[_u32](uint32_t const * addr)	addr -> Rn	VLD20.32 {Qd - Qd2},[Rn] VLD21.32 {Qd - Qd2},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1]	MVE
float16x8x2_t [arm_]vld2q[_f16](float16_t const * addr)	addr -> Rn	VLD20.16 {Qd - Qd2},[Rn] VLD21.16 {Qd - Qd2},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1]	MVE
float32x4x2_t [arm_]vld2q[_f32](float32_t const * addr)	addr -> Rn	VLD20.32 {Qd - Qd2},[Rn] VLD21.32 {Qd - Qd2},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1]	MVE
int8x16x4_t [_arm_]vld4q[_s8](int8_t const * addr)	addr -> Rn	VLD40.8 {Qd - Qd4},[Rn] VLD41.8 {Qd - Qd4},[Rn] VLD42.8 {Qd - Qd4},[Rn] VLD43.8 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
int16x8x4_t [_arm_]vld4q[_s16](int16_t const * addr)	addr -> Rn	VLD40.16 {Qd - Qd4},[Rn] VLD41.16 {Qd - Qd4},[Rn] VLD42.16 {Qd - Qd4},[Rn] VLD43.16 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
int32x4x4_t [_arm_]vld4q[_s32](int32_t const * addr)	addr -> Rn	VLD40.32 {Qd - Qd4},[Rn] VLD41.32 {Qd - Qd4},[Rn] VLD42.32 {Qd - Qd4},[Rn] VLD43.32 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
uint8x16x4_t [_arm_]vld4q[_u8](uint8_t const * addr)	addr -> Rn	VLD40.8 {Qd - Qd4},[Rn] VLD41.8 {Qd - Qd4},[Rn] VLD42.8 {Qd - Qd4},[Rn] VLD43.8 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8x4_t [_arm_]vld4q[_u16](uint16_t const * addr)	addr -> Rn	VLD40.16 {Qd - Qd4},[Rn] VLD41.16 {Qd - Qd4},[Rn] VLD42.16 {Qd - Qd4},[Rn] VLD43.16 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
uint32x4x4_t [_arm_]vld4q[_u32](uint32_t const * addr)	addr -> Rn	VLD40.32 {Qd - Qd4},[Rn] VLD41.32 {Qd - Qd4},[Rn] VLD42.32 {Qd - Qd4},[Rn] VLD43.32 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
float16x8x4_t [_arm_]vld4q[_f16](float16_t const * addr)	addr -> Rn	VLD40.16 {Qd - Qd4},[Rn] VLD41.16 {Qd - Qd4},[Rn] VLD42.16 {Qd - Qd4},[Rn] VLD43.16 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
float32x4x4_t [arm_]vld4q[_f32](float32_t const * addr)	addr -> Rn	VLD40.32 {Qd - Qd4},[Rn] VLD41.32 {Qd - Qd4},[Rn] VLD42.32 {Qd - Qd4},[Rn] VLD43.32 {Qd - Qd4},[Rn] VLD43.32 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
int8x16_t [arm_]vldrbq_s8(int8_t const * base)	base -> Rn	VLDRB.8 Qd,[Rn]	Qd -> result	MVE
int16x8_t [_arm_]vldrbq_s16(int8_t const * base)	base -> Rn	VLDRB.S16 Qd,[Rn]	Qd -> result	MVE
int32x4_t [_arm_]vldrbq_s32(int8_t const * base) uint8x16_t [_arm_]vldrbq_u8(uint8_t const * base)	base -> Rn base -> Rn	VLDRB.S32 Qd,[Rn] VLDRB.8 Qd,[Rn]	Qd -> result Od -> result	MVE MVE
uint16x8_t [arm_]vldrbq_u16(uint8_t const * base)	base -> Rn	VLDRB.U16 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [arm_]vldrbq_u32(uint8_t const * base)	base -> Rn	VLDRB.U32 Qd,[Rn]	Qd -> result	MVE
int8x16_t [arm_]vldrbq_z_s8(int8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.8 Qd,[Rn]	Qd -> result	MVE
int16x8_t [arm_]vldrbq_z_s16(int8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.S16 Qd,[Rn]	Qd -> result	MVE
int32x4_t [arm_]vldrbq_z_s32(int8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.S32 Qd,[Rn]	Qd -> result	MVE
uint8x16_t [_arm_]vldrbq_z_u8(uint8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.8 Qd,[Rn]	Qd -> result	MVE
uint16x8_t [arm_]vldrbq_z_u16(uint8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.U16 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [arm_]vldrbq_z_u32(uint8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.U32 Qd,[Rn]	Qd -> result	MVE
int16x8_t [arm_]vldrhq_s16(int16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE
int32x4_t [_arm_]vldrhq_s32(int16_t const * base)	base -> Rn	VLDRH.S32 Qd,[Rn]	Qd -> result	MVE
uint16x8_t [_arm_]vldrhq_u16(uint16_t const * base) uint32x4_t [_arm_]vldrhq_u32(uint16_t const * base)	base -> Rn base -> Rn	VLDRH.16 Qd,[Rn] VLDRH.U32 Qd,[Rn]	Qd -> result Qd -> result	MVE MVE
float16x8_t [arm_]vldrhq_f16(float16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE
int16x8_t [arm_]vldrhq_z_s16(int16_t const * base,	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VLDRHT.S16 Qd,[Rn]		) AUE
int32x4_t [arm_]vldrhq_z_s32(int16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRHT.S32 Qd,[Rn]	Qd -> result	MVE
uint16x8_t [arm_]vldrhq_z_u16(uint16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRHT.U16 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [_arm_]vldrhq_z_u32(uint16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRHT.U32 Qd,[Rn]	Qd -> result	MVE
float16x8_t [_arm_]vldrhq_z_fl6(float16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRHT.F16 Qd,[Rn]	Qd -> result	MVE
int32x4_t [_arm_]vldrwq_s32(int32_t const * base)	base -> Rn	VLDRW.32 Qd,[Rn]	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [arm_]vldrwq_u32(uint32_t const * base)	base -> Rn	VLDRW.32 Qd,[Rn]	Qd -> result	MVE
float32x4_t [arm_]vldrwq_f32(float32_t const * base)	base -> Rn	VLDRW.32 Qd,[Rn]	Qd -> result	MVE
int32x4_t [arm_]vldrwq_z_s32(int32_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRWT.32 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [_arm_]vldrwq_z_u32(uint32_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
float32x4_t [_arm_]vldrwq_z_f32(float32_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VLDRWT.32 Qd,[Rn] VMSR P0,Rp VPST	Qd -> result	MVE
	рукр	VLDRWT.32 Qd,[Rn]		
int8x16_t [_arm_]vld1q[_s8](int8_t const * base)	base -> Rn	VLDRB.8 Qd,[Rn]	Qd -> result	MVE/NEON
int16x8_t [_arm_]vld1q[_s16](int16_t const * base) int32x4 t [ arm ]vld1q[ s32](int32 t const * base)	base -> Rn base -> Rn	VLDRH.16 Qd,[Rn] VLDRW.32 Qd,[Rn]	Qd -> result Qd -> result	MVE/NEON MVE/NEON
uint8x16_t [arm_]vld1q[_u8](uint8_t const * base)	base -> Rn	VLDRB.8 Qd,[Rn]	Qd -> result	MVE/NEON MVE/NEON
uint16x8_t [arm_]vld1q[_u16](uint16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE/NEON
uint32x4_t [arm_]vld1q[_u32](uint32_t const * base)	base -> Rn	VLDRW.32 Qd,[Rn]	Qd -> result	MVE/NEON
float16x8_t [arm_]vld1q[_f16](float16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE/NEON
float32x4_t [arm_]vld1q[_f32](float32_t const * base)	base -> Rn	VLDRW.32 Qd,[Rn]	Qd -> result	MVE/NEON
int8x16_t [arm_]vld1q_z[_s8](int8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
		VLDRBT.8 Qd,[Rn]		1.00
int16x8_t [arm_]vld1q_z[_s16](int16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
inve_pred1o_t p)	p -> Kp	VLDRHT.16 Qd,[Rn]		
int32x4_t [_arm_]vld1q_z[_s32](int32_t const * base,	base -> Rn	VMSR P0,Rp VPST	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VLDRWT.32 Qd,[Rn]		
uint8x16_t [arm_]vld1q_z[_u8](uint8_t const * base,	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
	1 D	VLDRBT.8 Qd,[Rn]	0.1	MVE
uint16x8_t [_arm_]vld1q_z[_u16](uint16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t [_arm_]vld1q_z[_u32](uint32_t const * base,	base -> Rn	VLDRHT.16 Qd,[Rn] VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VLDRWT.32 Qd,[Rn]	Qu -> result	MYL
float16x8_t [arm_]vld1q_z[_f16](float16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
G - 22 4 - 1 - 2 111 - 1 (22)(G - 22		VLDRHT.16 Qd,[Rn]		) am
float32x4_t [_arm_]vld1q_z[_f32](float32_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRWT.32 Qd,[Rn]	Qd -> result	MVE
int16x8_t [_arm_]vldrhq_gather_offset[_s16](int16_t const * base, uint16x8_t offset)	base -> Rn offset -> Qm	VLDRH.U16 Qd,[Rn,Qm]	Qd -> result	MVE
int32x4_t [_arm_]vldrhq_gather_offset[_s32](int16_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRH.S32 Qd,[Rn,Qm]	Qd -> result	MVE
uint16x8_t [arm_]vldrhq_gather_offset[_u16](uint16_t	base -> Rn offset -> Qm	VLDRH.U16 Qd,[Rn,Qm]	Qd -> result	MVE
const * base, uint16x8_t offset) uint32x4_t [_arm_]vldrhq_gather_offset[_u32](uint16_t	base -> Rn	VLDRH.U32 Qd,[Rn,Qm]	Qd -> result	MVE
const * base, uint32x4_t offset) float16x8_t [_arm_]vldrhq_gather_offset[_f16](float16_t	offset -> Qm base -> Rn	VLDRH.F16 Qd,[Rn,Qm]	Qd -> result	MVE
const * base, uint16x8_t offset) int16x8_t [arm_]vldrhq_gather_offset_z[_s16](int16_t	offset -> Qm base -> Rn	VMSR P0,Rp	Qd -> result	MVE
const * base, uint16x8_t offset, mve_pred16_t p)	offset -> Qm p -> Rp	VPST VLDRHT.U16 Qd,[Rn,Qm]		
int32x4_t [_arm_]vldrhq_gather_offset_z[_s32](int16_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VLDRHT.S32 Qd,[Rn,Qm]	01 : 1:	MVE
uint16x8_t [arm_]vldrhq_gather_offset_z[_u16](uint16_t const * base, uint16x8 t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRHT.U16 Qd,[Rn,Qm]	Qd -> result	MVE
uint32x4_t [_arm_]vldrhq_gather_offset_z[_u32](uint16_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRHT.U32 Qd,[Rn,Qm]	Qd -> result	MVE
float16x8_t [_arm_]vldrhq_gather_offset_z[_f16](float16_t const * base, uint16x8_t offset, mve_pred16_t p)	base -> Rn offset -> Qm	VMSR P0,Rp VPST VLDRHT.F16 Qd,[Rn,Qm]	Qd -> result	MVE
int16x8_t [arm_]vldrhq_gather_shifted_offset[_s16](int16_t const	p -> Rp base -> Rn offset -> Qm	VLDRH.U16 Qd,[Rn,Qm,UXTW #1]	Qd -> result	MVE
* base, uint16x8_t offset) int32x4_t [_arm_]vldrhq_gather_shifted_offset[_s32](int16_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRH.S32 Qd,[Rn,Qm,UXTW #1]	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [_arm_]vldrhq_gather_shifted_offset[_u16](uint16_t const * base, uint16x8_t offset)	base -> Rn offset -> Qm	VLDRH.U16 Qd,[Rn,Qm,UXTW #1]	Qd -> result	MVE
uint32x4_t [_arm_]vldrhq_gather_shifted_offset[_u32](uint16_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRH.U32 Qd,[Rn,Qm,UXTW #1]	Qd -> result	MVE
float16x8_t [_arm_]vldrhq_gather_shifted_offset[_f16](float16_t const * base, uint16x8_t offset)	base -> Rn offset -> Qm	VLDRH.F16 Qd,[Rn,Qm,UXTW #1]	Qd -> result	MVE
int16x8_t init16x8_t offset) int16x8_t [_arm_]vldrhq_gather_shifted_offset_z[_s16](int16_t const * base, uint16x8_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRHT.U16 Qd,[Rn,Qm,UXTW #1]	Qd -> result	MVE
int32x4_t [_arm_]vldrhq_gather_shifted_offset_z[_s32](int16_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRHT.S32 Qd,[Rn,Qm,UXTW #1]	Qd -> result	MVE
uint16x8_t [_arm_ vldrhq_gather_shifted_offset_z[_u16](uint16_t const * base, uint16x8_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRHT.U16 Qd,[Rn,Qm,UXTW #1]	Qd -> result	MVE
uint32x4_t [arm_]vldrhq_gather_shifted_offset_z[_u32](uint16_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRHT.U32 Qd,[Rn,Qm,UXTW #1]	Qd -> result	MVE
float16x8_t [arm_]vldrhq_gather_shifted_offset_z[_f16](float16_t const * base, uint16x8_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRHT.F16 Qd,[Rn,Qm,UXTW #1]	Qd -> result	MVE
int8x16_t [_arm_]vldrbq_gather_offset[_s8](int8_t const * base, uint8x16_t offset)	base -> Rn offset -> Qm	VLDRB.U8 Qd,[Rn,Qm]	Qd -> result	MVE
int16x8_t [_arm_]vldrbq_gather_offset[_s16](int8_t const * base, uint16x8_t offset)	base -> Rn offset -> Qm	VLDRB.S16 Qd,[Rn,Qm]	Qd -> result	MVE
int32x4_t [_arm_]vldrbq_gather_offset[_s32](int8_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRB.S32 Qd,[Rn,Qm]	Qd -> result	MVE
uint8x16_t [_arm_]vldrbq_gather_offset[_u8](uint8_t const * base, uint8x16_t offset)	base -> Rn offset -> Qm	VLDRB.U8 Qd,[Rn,Qm]	Qd -> result	MVE
uint16x8_t [_arm_]vldrbq_gather_offset[_u16](uint8_t const * base, uint16x8_t offset)	base -> Rn offset -> Qm	VLDRB.U16 Qd,[Rn,Qm]	Qd -> result	MVE
uint32x4_t [_arm_]vldrbq_gather_offset[_u32](uint8_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRB.U32 Qd,[Rn,Qm]	Qd -> result	MVE
int8x16_t [_arm_]vldrbq_gather_offset_z[_s8](int8_t const * base, uint8x16_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRBT.U8 Qd,[Rn,Qm]	Qd -> result	MVE
int16x8_t [_arm_]vldrbq_gather_offset_z[_s16](int8_t const * base, uint16x8_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRBT.S16 Qd,[Rn,Qm]	Qd -> result	MVE
int32x4_t [_arm_]vldrbq_gather_offset_z[_s32](int8_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRBT.S32 Qd,[Rn,Qm]	Qd -> result	MVE
uint8x16_t [_arm_]vldrbq_gather_offset_z[_u8](uint8_t const * base, uint8x16_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRBT.U8 Qd,[Rn,Qm]	Qd -> result	MVE
uint16x8_t [_arm_]vldrbq_gather_offset_z[_u16](uint8_t const * base, uint16x8_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRBT.U16 Qd,[Rn,Qm]	Qd -> result	MVE
uint32x4_t [_arm_]vldrbq_gather_offset_z[_u32](uint8_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRBT.U32 Qd,[Rn,Qm]	Qd -> result	MVE
int32x4_t [_arm_]vldrwq_gather_offset[_s32](int32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRW.U32 Qd,[Rn,Qm]	Qd -> result	MVE
uint32x4_t [_arm_]vldrwq_gather_offset[_u32](uint32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRW.U32 Qd,[Rn,Qm]	Qd -> result	MVE
float32x4_t [_arm_]vldrwq_gather_offset[_f32](float32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRW.U32 Qd,[Rn,Qm]	Qd -> result	MVE
int32x4_t [_arm_]vldrwq_gather_offset_z[_s32](int32_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Rn,Qm]	Qd -> result	MVE
uint32x4_t [_arm_]vldrwq_gather_offset_z[_u32](uint32_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Rn,Qm]	Qd -> result	MVE
float32x4_t [_arm_]vldrwq_gather_offset_z[_f32](float32_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Rn,Qm]	Qd -> result	MVE
int32x4_t [arm_]vldrwq_gather_shifted_offset[_s32](int32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRW.U32 Qd,[Rn,Qm,UXTW #2]	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t	base -> Rn offset -> Om	VLDRW.U32	Qd -> result	MVE
[_arm_]vldrwq_gather_shifted_offset[_u32](uint32_t const * base, uint32x4_t offset)	Ì	Qd,[Rn,Qm,UXTW #2]		
float32x4_t [_arm_]vldrwq_gather_shifted_offset[_f32](float32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRW.U32 Qd,[Rn,Qm,UXTW #2]	Qd -> result	MVE
int32x4_t [_arm_]vldrwq_gather_shifted_offset_z[_s32](int32_t	base -> Rn offset -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
const * base, uint32x4_t offset, mve_pred16_t p)	p -> Rp	VLDRWT.U32 Qd,[Rn,Qm,UXTW #2]		
uint32x4_t [arm_]vldrwq_gather_shifted_offset_z[_u32](uint32_t	base -> Rn offset -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
const * base, uint32x4_t offset, mve_pred16_t p)	p -> Rp	VLDRWT.U32 Qd,[Rn,Qm,UXTW #2]		
float32x4_t [_arm_]vldrwq_gather_shifted_offset_z[_f32](float32_t	base -> Rn offset -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
const * base, uint32x4_t offset, mve_pred16_t p)	p -> Rp	VLDRWT.U32 Qd,[Rn,Qm,UXTW #2]		
int32x4_t [_arm_]vldrwq_gather_base_s32(uint32x4_t addr, const int offset)	addr -> Qn offset in +/-	VLDRW.U32 Qd,[Qn,#offset]	Qd -> result	MVE
	4*[0127]			
uint32x4_t [arm_]vldrwq_gather_base_u32(uint32x4_t addr, const int offset)	addr -> Qn offset in +/- 4*[0127]	VLDRW.U32 Qd,[Qn,#offset]	Qd -> result	MVE
float32x4_t [_arm_]vldrwq_gather_base_f32(uint32x4_t addr, const int offset)	addr -> Qn offset in +/-	VLDRW.U32 Qd,[Qn,#offset]	Qd -> result	MVE
int32x4 t [ arm ]vldrwq gather base z s32(uint32x4 t	4*[0127] addr -> On	VMSR P0,Rp	Od -> result	MVE
addr, const int offset, mve_pred16_t p)	offset in +/-	VPST	Qu => resuit	WIVE
	4*[0127] p -> Rp	VLDRWT.U32 Qd,[Qn,#offset]		
uint32x4_t	addr -> Qn offset in +/-	VMSR P0,Rp VPST	Qd -> result	MVE
[arm_]vldrwq_gather_base_z_u32(uint32x4_t addr, const int offset, mve_pred16_t p)	4*[0127] p -> Rp	VLDRWT.U32 Qd,[Qn,#offset]		
float32x4_t	addr -> Qn	VMSR P0,Rp	Qd -> result	MVE
[_arm_]vldrwq_gather_base_z_f32(uint32x4_t addr, const int offset, mve_pred16_t p)	offset in +/- 4*[0127] p -> Rp	VPST VLDRWT.U32 Qd,[Qn,#offset]		
int32x4_t [arm_]vldrwq_gather_base_wb_s32(uint32x4_t * addr,	*addr -> Qn offset in +/-	VLDRW.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
const int offset) uint32x4 t	4*[0127] *addr -> Qn	VLDRW.U32 Qd,[Qn,#offset]!	Qd -> result	MVE
[arm_]vldrwq_gather_base_wb_u32(uint32x4_t * addr, const int offset)	offset in +/- 4*[0127]		Qn -> *addr	
float32x4_t [arm_]vldrwq_gather_base_wb_f32(uint32x4_t * addr, const int offset)	*addr -> Qn offset in +/- 4*[0127]	VLDRW.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
int32x4_t	*addr -> Qn	VMSR P0,Rp	Qd -> result	MVE
[_arm_]vldrwq_gather_base_wb_z_s32(uint32x4_t * addr, const int offset, mve_pred16_t p)	offset in +/- 4*[0127] p -> Rp	VPST VLDRWT.U32 Qd,[Qn,#offset]!	Qn -> *addr	
uint32x4_t	*addr -> Qn	VMSR P0,Rp	Qd -> result	MVE
[_arm_]vldrwq_gather_base_wb_z_u32(uint32x4_t * addr, const int offset, mve_pred16_t p)	offset in +/- 4*[0127] p -> Rp	VPST VLDRWT.U32 Qd,[Qn,#offset]!	Qn -> *addr	
float32x4_t	*addr -> Qn	VMSR P0,Rp	Qd -> result	MVE
[_arm_]vldrwq_gather_base_wb_z_f32(uint32x4_t * addr, const int offset, mve_pred16_t p)	offset in +/- 4*[0127] p -> Rp	VPST VLDRWT.U32 Qd,[Qn,#offset]!	Qn -> *addr	
int64x2_t [_arm_]vldrdq_gather_offset[_s64](int64_t const * base, uint64x2_t offset)	base -> Rn offset -> Qm	VLDRD.U64 Qd,[Rn,Qm]	Qd -> result	MVE
uint64x2_t [_arm_]vldrdq_gather_offset[_u64](uint64_t const * base, uint64x2_t offset)	base -> Rn offset -> Qm	VLDRD.U64 Qd,[Rn,Qm]	Qd -> result	MVE
int64x2_t [_arm_]vldrdq_gather_offset_z[_s64](int64_t const * base, uint64x2_t offset, mve_pred16_t p)	base -> Rn offset -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
uint64x2_t	p -> Rp base -> Rn	VLDRDT.U64 Qd,[Rn,Qm] VMSR P0,Rp	Qd -> result	MVE
[arm_]vldrdq_gather_offset_z[_u64](uint64_t const * base, uint64x2_t offset, mve_pred16_t p)	offset -> Qm p -> Rp	VPST VLDRDT.U64 Qd,[Rn,Qm]		
int64x2_t [arm_]vldrdq_gather_shifted_offset[_s64](int64_t const	base -> Rn offset -> Qm	VLDRD.U64 Qd,[Rn,Qm,UXTW #3]	Qd -> result	MVE
* base, uint64x2_t offset) uint64x2_t [arm]vldrdq_gather_shifted_offset[_u64](uint64_t	base -> Rn offset -> Qm	VLDRD.U64 Qd,[Rn,Qm,UXTW #3]	Qd -> result	MVE
const * base, uint64x2_t offset)	\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	- 1		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64x2_t [_arm_]vldrdq_gather_shifted_offset_z[_s64](int64_t const * base, uint64x2_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRDT.U64 Qd,[Rn,Qm,UXTW #3]	Qd -> result	MVE
uint64x2_t [_arm_]vldrdq_gather_shifted_offset_z[_u64](uint64_t const * base, uint64x2_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRDT.U64 Qd,[Rn,Qm,UXTW #3]	Qd -> result	MVE
int64x2_t [arm_]vldrdq_gather_base_s64(uint64x2_t addr, const int offset)	addr -> Qn offset in +/- 8*[0127]	VLDRD.64 Qd,[Qn,#offset]	Qd -> result	MVE
uint64x2_t [arm_]vldrdq_gather_base_u64(uint64x2_t addr, const int offset)	addr -> Qn offset in +/- 8*[0127]	VLDRD.64 Qd,[Qn,#offset]	Qd -> result	MVE
int64x2_t [arm_]vldrdq_gather_base_z_s64(uint64x2_t addr, const int offset, mve_pred16_t p)	addr -> Qn offset in +/- 8*[0127] p -> Rp	VMSR P0,Rp VPST VLDRDT.U64 Qd,[Qn,#offset]	Qd -> result	MVE
uint64x2_t [arm_]vldrdq_gather_base_z_u64(uint64x2_t addr, const int offset, mve_pred16_t p)	addr -> Qn offset in +/- 8*[0127] p -> Rp	VMSR P0,Rp VPST VLDRDT.U64 Qd,[Qn,#offset]	Qd -> result	MVE
int64x2_t [arm_]vldrdq_gather_base_wb_s64(uint64x2_t * addr, const int offset)	*addr -> Qn offset in +/- 8*[0127]	VLDRD.64 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
uint64x2_t [_arm_lvldrdq_gather_base_wb_u64(uint64x2_t * addr, const int offset)	*addr -> Qn offset in +/- 8*[0127]	VLDRD.64 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
int64x2_t [_arm_lvldrdq_gather_base_wb_z_s64(uint64x2_t * addr, const int offset, mve_pred16_t p)	*addr -> Qn offset in +/- 8*[0127] p -> Rp	VMSR P0,Rp VPST VLDRDT.U64 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
uint64x2_t [_arm_]vldrdq_gather_base_wb_z_u64(uint64x2_t * addr, const int offset, mve_pred16_t p)	*addr -> Qn offset in +/- 8*[0127] p -> Rp	VMSR P0,Rp VPST VLDRDT.U64 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
void [_arm_]vst2q[_s8](int8_t * addr, int8x16x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2	VST20.8 {Qd - Qd2},[Rn] VST21.8 {Qd - Qd2},[Rn]	void -> result	MVE
void [_arm_]vst2q[_s16](int16_t * addr, int16x8x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Od2	VST20.16 {Qd - Qd2},[Rn] VST21.16 {Qd - Qd2},[Rn]	void -> result	MVE
void [_arm_]vst2q[_s32](int32_t * addr, int32x4x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2	VST20.32 {Qd - Qd2},[Rn] VST21.32 {Qd - Qd2},[Rn]	void -> result	MVE
void [_arm_]vst2q[_u8](uint8_t * addr, uint8x16x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2	VST20.8 {Qd - Qd2},[Rn] VST21.8 {Qd - Qd2},[Rn]	void -> result	MVE
void [_arm_]vst2q[_u16](uint16_t * addr, uint16x8x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2	VST20.16 {Qd - Qd2},[Rn] VST21.16 {Qd - Qd2},[Rn]	void -> result	MVE
void [_arm_]vst2q[_u32](uint32_t * addr, uint32x4x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Od2	VST20.32 {Qd - Qd2},[Rn] VST21.32 {Qd - Qd2},[Rn]	void -> result	MVE
void [_arm_]vst2q[_f16](float16_t * addr, float16x8x2_t value)	addr -> Rn   value.val[0] ->   Qd   value.val[1] ->   Qd2	VST20.16 {Qd - Qd2},[Rn] VST21.16 {Qd - Qd2},[Rn]	void -> result	MVE
void [_arm_]vst2q[_f32](float32_t * addr, float32x4x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2	VST20.32 {Qd - Qd2},[Rn] VST21.32 {Qd - Qd2},[Rn]	void -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void [_arm_]vst4q[_s8](int8_t * addr, int8x16x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.8 {Qd - Qd4},[Rn] VST41.8 {Qd - Qd4},[Rn] VST42.8 {Qd - Qd4},[Rn] VST43.8 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_s16](int16_t * addr, int16x8x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.16 {Qd - Qd4},[Rn] VST41.16 {Qd - Qd4},[Rn] VST42.16 {Qd - Qd4},[Rn] VST43.16 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_s32](int32_t * addr, int32x4x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.32 {Qd - Qd4},[Rn] VST41.32 {Qd - Qd4},[Rn] VST42.32 {Qd - Qd4},[Rn] VST43.32 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_u8](uint8_t * addr, uint8x16x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.8 {Qd - Qd4},[Rn] VST41.8 {Qd - Qd4},[Rn] VST42.8 {Qd - Qd4},[Rn] VST43.8 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_u16](uint16_t * addr, uint16x8x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.16 {Qd - Qd4},[Rn] VST41.16 {Qd - Qd4},[Rn] VST42.16 {Qd - Qd4},[Rn] VST43.16 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_u32](uint32_t * addr, uint32x4x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.32 {Qd - Qd4},[Rn] VST41.32 {Qd - Qd4},[Rn] VST42.32 {Qd - Qd4},[Rn] VST42.32 {Qd - Qd4},[Rn] VST43.32 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_f16](float16_t * addr, float16x8x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.16 {Qd - Qd4},[Rn] VST41.16 {Qd - Qd4},[Rn] VST42.16 {Qd - Qd4},[Rn] VST43.16 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_f32](float32_t * addr, float32x4x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.32 {Qd - Qd4},[Rn] VST41.32 {Qd - Qd4},[Rn] VST42.32 {Qd - Qd4},[Rn] VST43.32 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vstrbq[_s8](int8_t * base, int8x16_t value)	base -> Rn value -> Qd	VSTRB.8 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrbq[_s16](int8_t * base, int16x8_t value)	base -> Rn value -> Qd	VSTRB.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrbq[_s32](int8_t * base, int32x4_t value)	base -> Rn value -> Qd	VSTRB.32 Qd,[Rn]	void -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void [_arm_]vstrbq[_u8](uint8_t * base, uint8x16_t value)	base -> Rn value -> Qd	VSTRB.8 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrbq[_u16](uint8_t * base, uint16x8_t value)	base -> Rn value -> Qd	VSTRB.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrbq[_u32](uint8_t * base, uint32x4_t value)	base -> Rn value -> Qd	VSTRB.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrbq_p[_s8](int8_t * base, int8x16_t value, mve_pred16_t p)	base -> Rn value -> Qd	VMSR P0,Rp VPST	void -> result	MVE
void [arm_]vstrbq_p[_s16](int8_t * base, int16x8_t value, mve_pred16_t p)	p -> Rp base -> Rn value -> Qd	VSTRBT.8 Qd,[Rn]  VMSR P0,Rp  VPST	void -> result	MVE
void [_arm_lvstrbq_p[_s32](int8_t * base, int32x4_t	p -> Rp base -> Rn	VSTRBT.16 Qd,[Rn] VMSR P0,Rp	void -> result	MVE
value, mve_pred16_t p)	value -> Qd p -> Rp	VPST VSTRBT.32 Qd,[Rn]	voia > result	
void [_arm_]vstrbq_p[_u8](uint8_t * base, uint8x16_t value, mve_pred16_t p)	base -> Rn value -> Qd	VMSR P0,Rp VPST	void -> result	MVE
void [_arm_]vstrbq_p[_u16](uint8_t * base, uint16x8_t value, mve_pred16_t p)	p -> Rp base -> Rn value -> Qd	VSTRBT.8 Qd,[Rn]  VMSR P0,Rp  VPST	void -> result	MVE
void [_arm_]vstrbq_p[_u32](uint8_t * base, uint32x4_t value, mve_pred16_t p)	p -> Rp base -> Rn value -> Qd	VSTRBT.16 Qd,[Rn]  VMSR P0,Rp  VPST	void -> result	MVE
void [_arm_]vstrhq[_s16](int16_t * base, int16x8_t value)	p -> Rp base -> Rn value -> Od	VSTRBT.32 Qd,[Rn] VSTRH.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrhq[_s32](int16_t * base, int32x4_t value)	base -> Rn value -> Qd	VSTRH.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrhq[_u16](uint16_t * base, uint16x8_t value)	base -> Rn value -> Qd	VSTRH.16 Qd,[Rn]	void -> result	MVE
void [arm_]vstrhq[_u32](uint16_t * base, uint32x4_t value)	base -> Rn value -> Qd	VSTRH.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrhq[_f16](float16_t * base, float16x8_t value)	base -> Rn value -> Qd	VSTRH.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrhq_p[_s16](int16_t * base, int16x8_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrhq_p[_s32](int16_t * base, int32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrhq_p[_u16](uint16_t * base, uint16x8_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrhq_p[_u32](uint16_t * base, uint32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrhq_p[_f16](float16_t * base, float16x8_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrwq[_s32](int32_t * base, int32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE
void [arm_]vstrwq[_u32](uint32_t * base, uint32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE
void [arm_]vstrwq[_f32](float32_t * base, float32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrwq_p[_s32](int32_t * base, int32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrwq_p[_u32](uint32_t * base, uint32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrwq_p[_f32](float32_t * base, float32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vst1q[_s8](int8_t * base, int8x16_t value)	base -> Rn value -> Qd	VSTRB.8 Qd,[Rn]	void -> result	MVE/NEON
void [_arm_]vst1q[_s16](int16_t * base, int16x8_t value)	base -> Rn value -> Qd	VSTRH.16 Qd,[Rn]	void -> result	MVE/NEON
void [_arm_]vst1q[_s32](int32_t * base, int32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE/NEON
void [_arm_]vst1q[_u8](uint8_t * base, uint8x16_t value)	base -> Rn value -> Qd	VSTRB.8 Qd,[Rn]	void -> result	MVE/NEON
void [arm_]vst1q[_u16](uint16_t * base, uint16x8_t value)	base -> Rn value -> Qd	VSTRH.16 Qd,[Rn]	void -> result	MVE/NEON
void [_arm_]vst1q[_u32](uint32_t * base, uint32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void [_arm_]vst1q[_f16](float16_t * base, float16x8_t value)	base -> Rn value -> Qd	VSTRH.16 Qd,[Rn]	void -> result	MVE/NEON
void [_arm_]vst1q[_f32](float32_t * base, float32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE/NEON
void [_arm_]vst1q_p[_s8](int8_t * base, int8x16_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRBT.8 Qd,[Rn]	void -> result	MVE
void [_arm_]vst1q_p[_s16](int16_t * base, int16x8_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vst1q_p[_s32](int32_t * base, int32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vst1q_p[_u8](uint8_t * base, uint8x16_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRBT.8 Qd,[Rn]	void -> result	MVE
void [_arm_]vst1q_p[_u16](uint16_t * base, uint16x8_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vst1q_p[_u32](uint32_t * base, uint32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vst1q_p[_f16](float16_t * base, float16x8_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vst1q_p[_f32](float32_t * base, float32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset[_s8](int8_t * base, uint8x16_t offset, int8x16_t value)	base -> Rn offset -> Qm value -> Qd	VSTRB.8 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset[_s16](int8_t * base, uint16x8_t offset, int16x8_t value)	base -> Rn offset -> Qm value -> Qd	VSTRB.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset[_s32](int8_t * base, uint32x4_t offset, int32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRB.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset[_u8](uint8_t * base, uint8x16_t offset, uint8x16_t value)	base -> Rn offset -> Qm value -> Qd	VSTRB.8 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset[_u16](uint8_t * base, uint16x8_t offset, uint16x8_t value)	base -> Rn offset -> Qm value -> Qd	VSTRB.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset[_u32](uint8_t * base, uint32x4_t offset, uint32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRB.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset_p[_s8](int8_t * base, uint8x16_t offset, int8x16_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRBT.8 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset_p[_s16](int8_t * base, uint16x8_t offset, int16x8_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRBT.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset_p[_s32](int8_t * base, uint32x4_t offset, int32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRBT.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset_p[_u8](uint8_t * base, uint8x16_t offset, uint8x16_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRBT.8 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset_p[_u16](uint8_t * base, uint16x8_t offset, uint16x8_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRBT.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrbq_scatter_offset_p[_u32](uint8_t * base, uint32x4_t offset, uint32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRBT.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset[_s16](int16_t * base, uint16x8_t offset, int16x8_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset[_s32](int16_t * base, uint32x4_t offset, int32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.32 Qd,[Rn,Qm]	void -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void [arm_]vstrhq_scatter_offset[_u16](uint16_t * base, uint16x8_t offset, uint16x8_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset[_u32](uint16_t * base, uint32x4_t offset, uint32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset[_f16](float16_t * base, uint16x8_t offset, float16x8_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset_p[_s16](int16_t * base, uint16x8_t offset, int16x8_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset_p[_s32](int16_t * base, uint32x4_t offset, int32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset_p[_u16](uint16_t * base, uint16x8_t offset, uint16x8_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset_p[_u32](uint16_t * base, uint32x4_t offset, uint32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset_p[_f16](float16_t * base, uint16x8_t offset, float16x8_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_shifted_offset[_s16](int16_t * base, uint16x8_t offset, int16x8_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.16 Qd,[Rn,Qm,UXTW #1]	void -> result	MVE
void [_arm_]vstrhq_scatter_shifted_offset[_s32](int16_t * base, uint32x4_t offset, int32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.32 Qd,[Rn,Qm,UXTW #1]	void -> result	MVE
void [_arm_]vstrhq_scatter_shifted_offset[_u16](uint16_t * base, uint16x8_t offset, uint16x8_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.16 Qd,[Rn,Qm,UXTW #1]	void -> result	MVE
void [_arm_]vstrhq_scatter_shifted_offset[_u32](uint16_t * base, uint32x4_t offset, uint32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.32 Qd,[Rn,Qm,UXTW #1]	void -> result	MVE
void [_arm_]vstrhq_scatter_shifted_offset[_f16](float16_t * base, uint16x8_t offset, float16x8_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.16 Qd,[Rn,Qm,UXTW #1]	void -> result	MVE
void [_arm_]vstrhq_scatter_shifted_offset_p[_s16](int16_t * base, uint16x8_t offset, int16x8_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm,UXTW #1]	void -> result	MVE
void [_arm_]vstrhq_scatter_shifted_offset_p[_s32](int16_t * base, uint32x4_t offset, int32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn,Qm,UXTW #1]	void -> result	MVE
void [_arm_]vstrhq_scatter_shifted_offset_p[_u16](uint16_t * base, uint16x8_t offset, uint16x8_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm,UXTW #1]	void -> result	MVE
void [_arm_]vstrhq_scatter_shifted_offset_p[_u32](uint16_t * base, uint32x4_t offset, uint32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn,Qm,UXTW #1]	void -> result	MVE
void [_arm_]vstrhq_scatter_shifted_offset_p[_f16](float16_t * base, uint16x8_t offset, float16x8_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm,UXTW	void -> result	MVE
void [arm_]vstrwq_scatter_base[_s32](uint32x4_t addr, const int offset, int32x4_t value)	addr -> Qn offset in +/- 4*[0127] value -> Qd	VSTRW.U32 Qd,[Qn,#offset]	void -> result	MVE
void [arm_]vstrwq_scatter_base[_u32](uint32x4_t addr, const int offset, uint32x4_t value)	addr -> Qn offset in +/- 4*[0127] value -> Qd	VSTRW.U32 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrwq_scatter_base[_f32](uint32x4_t addr, const int offset, float32x4_t value)	addr -> Qn offset in +/- 4*[0127] value -> Qd	VSTRW.U32 Qd,[Qn,#offset]	void -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void [_arm_]vstrwq_scatter_base_p[_s32](uint32x4_t addr, const int offset, int32x4_t value, mve_pred16_t p)	addr -> Qn offset in +/- 4*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrwq_scatter_base_p[_u32](uint32x4_t addr, const int offset, uint32x4_t value, mve_pred16_t p)	addr -> Qn offset in +/- 4*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrwq_scatter_base_p[_f32](uint32x4_t addr, const int offset, float32x4_t value, mve_pred16_t p)	addr -> Qn offset in +/- 4*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrwq_scatter_base_wb[_s32](uint32x4_t * addr, const int offset, int32x4_t value)	*addr -> Qn offset in +/- 4*[0127] value -> Qd	VSTRW.U32 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrwq_scatter_base_wb[_u32](uint32x4_t * addr, const int offset, uint32x4_t value)	*addr -> Qn offset in +/- 4*[0127] value -> Qd	VSTRW.U32 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrwq_scatter_base_wb[_f32](uint32x4_t * addr, const int offset, float32x4_t value)	*addr -> Qn offset in +/- 4*[0127] value -> Qd	VSTRW.U32 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrwq_scatter_base_wb_p[_s32](uint32x4_t * addr, const int offset, int32x4_t value, mve_pred16_t p)	*addr -> Qn offset in +/- 4*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]!	void -> result	MVE
void [arm_]vstrwq_scatter_base_wb_p[_u32](uint32x4_t * addr, const int offset, uint32x4_t value, mve_pred16_t p)	*addr -> Qn offset in +/- 4*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrwq_scatter_base_wb_p[_f32](uint32x4_t * addr, const int offset, float32x4_t value, mve_pred16_t p)	*addr -> Qn offset in +/- 4*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrwq_scatter_offset[_s32](int32_t * base, uint32x4_t offset, int32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRW.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrwq_scatter_offset[_u32](uint32_t * base, uint32x4_t offset, uint32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRW.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrwq_scatter_offset[_f32](float32_t * base, uint32x4_t offset, float32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRW.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrwq_scatter_offset_p[_s32](int32_t * base, uint32x4_t offset, int32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrwq_scatter_offset_p[_u32](uint32_t * base, uint32x4_t offset, uint32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrwq_scatter_offset_p[_f32](float32_t * base, uint32x4_t offset, float32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrwq_scatter_shifted_offset[_s32](int32_t * base, uint32x4_t offset, int32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRW.32 Qd,[Rn,Qm,UXTW #2]	void -> result	MVE
void [arm_]vstrwq_scatter_shifted_offset[_u32](uint32_t * base, uint32x4_t offset, uint32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRW.32 Qd,[Rn,Qm,UXTW #2]	void -> result	MVE
void [_arm_]vstrwq_scatter_shifted_offset[_f32](float32_t * base, uint32x4_t offset, float32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRW.32 Qd,[Rn,Qm,UXTW #2]	void -> result	MVE
void [_arm_]vstrwq_scatter_shifted_offset_p[_s32](int32_t * base, uint32x4_t offset, int32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm,UXTW #2]	void -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void [arm_]vstrwq_scatter_shifted_offset_p[_u32](uint32_t * base, uint32x4_t offset, uint32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm,UXTW #2]	void -> result	MVE
void [_arm_]vstrwq_scatter_shifted_offset_p[_f32](float32_t * base, uint32x4_t offset, float32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm,UXTW #2]	void -> result	MVE
void [_arm_]vstrdq_scatter_base[_s64](uint64x2_t addr, const int offset, int64x2_t value)	addr -> Qn offset in +/- 8*[0127] value -> Qd	VSTRD.U64 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrdq_scatter_base[_u64](uint64x2_t addr, const int offset, uint64x2_t value)	addr -> Qn offset in +/- 8*[0127] value -> Qd	VSTRD.U64 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrdq_scatter_base_p[_s64](uint64x2_t addr, const int offset, int64x2_t value, mve_pred16_t p)	addr -> Qn offset in +/- 8*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrdq_scatter_base_p[_u64](uint64x2_t addr, const int offset, uint64x2_t value, mve_pred16_t p)	addr -> Qn offset in +/- 8*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrdq_scatter_base_wb[_s64](uint64x2_t * addr, const int offset, int64x2_t value)	*addr -> Qn offset in +/- 8*[0127] value -> Qd	VSTRD.U64 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrdq_scatter_base_wb[_u64](uint64x2_t * addr, const int offset, uint64x2_t value)	*addr -> Qn offset in +/- 8*[0127] value -> Qd	VSTRD.U64 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrdq_scatter_base_wb_p[_s64](uint64x2_t * addr, const int offset, int64x2_t value, mve_pred16_t p)	*addr -> Qn offset in +/- 8*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrdq_scatter_base_wb_p[_u64](uint64x2_t * addr, const int offset, uint64x2_t value, mve_pred16_t p)	*addr -> Qn offset in +/- 8*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrdq_scatter_offset[_s64](int64_t * base, uint64x2_t offset, int64x2_t value)	base -> Rn offset -> Qm value -> Qd	VSTRD.64 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrdq_scatter_offset[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value)	base -> Rn offset -> Qm value -> Qd	VSTRD.64 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrdq_scatter_offset_p[_s64](int64_t * base, uint64x2_t offset, int64x2_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrdq_scatter_offset_p[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrdq_scatter_shifted_offset[_s64](int64_t * base, uint64x2_t offset, int64x2_t value)	base -> Rn offset -> Qm value -> Qd	VSTRD.64 Qd,[Rn,Qm,UXTW #3]	void -> result	MVE
void [_arm_]vstrdq_scatter_shifted_offset[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value)	base -> Rn offset -> Qm value -> Qd	VSTRD.64 Qd,[Rn,Qm,UXTW #3]	void -> result	MVE
void [_arm_]vstrdq_scatter_shifted_offset_p[_s64](int64_t * base, uint64x2_t offset, int64x2_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm,UXTW #3]	void -> result	MVE
void [arm_]vstrdq_scatter_shifted_offset_p[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm,UXTW #3]	void -> result	MVE
int64_t [arm_]vaddlvaq[_s32](int64_t a, int32x4_t b)	a -> [RdaHi,RdaLo] b -> Qm	VADDLVA.S32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64 t [ arm ]vaddlvaq[ u32](uint64 t a, uint32x4 t	a ->	VADDLVA.U32	[RdaHi,RdaLo]	MVE
b)	[RdaHi,RdaLo] b -> Qm	RdaLo,RdaHi,Qm	-> result	
int64_t [arm_]vaddlvaq_p[_s32](int64_t a, int32x4_t b,	a ->	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
mve_pred16_t p)	[RdaHi,RdaLo] b -> Om	VPST VADDLVAT.S32	-> result	
	p -> QIII	RdaLo,RdaHi,Qm		
uint64_t [arm_]vaddlvaq_p[_u32](uint64_t a,	a ->	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
uint32x4_t b, mve_pred16_t p)	[RdaHi,RdaLo]	VPST	-> result	
	b -> Qm	VADDLVAT.U32		
int64 t [ arm ]vaddlvq[ s32](int32x4 t a)	p -> Rp a -> Om	RdaLo,RdaHi,Qm VADDLV.S32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo]	MVE
10 10 10	`	, , , ,	-> result	·
uint64_t [arm_]vaddlvq[_u32](uint32x4_t a)	a -> Qm	VADDLV.U32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vaddlvq_p[_s32](int32x4_t a,	a -> Qm	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
mve_pred16_t p)	p -> Rp	VPST VADDLVT.S32	-> result	
		RdaLo,RdaHi,Qm		
uint64_t [arm_]vaddlvq_p[_u32](uint32x4_t a,	a -> Qm	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
mve_pred16_t p)	p -> Rp	VPST	-> result	
		VADDLVT.U32		
int32_t [arm_]vaddvaq[_s8](int32_t a, int8x16_t b)	a -> Rda	RdaLo,RdaHi,Qm VADDVA.S8 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvaq[_s8](int32_t a, int8x16_t b)	a -> Rda b -> Qm		Rda -> resuit	MVE
int32_t [arm_]vaddvaq[_s16](int32_t a, int16x8_t b)	a -> Rda b -> Qm	VADDVA.S16 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvaq[_s32](int32_t a, int32x4_t b)	a -> Rda b -> Om	VADDVA.S32 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vaddvaq[_u8](uint32_t a, uint8x16_t b)	a -> Rda b -> Qm	VADDVA.U8 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vaddvaq[_u16](uint32_t a, uint16x8_t b)	a -> Rda	VADDVA.U16 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vaddvaq[_u32](uint32_t a, uint32x4_t b)	b -> Qm a -> Rda	VADDVA.U32 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvaq_p[_s8](int32_t a, int8x16_t b,	b -> Qm a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mis2_t [aim_]vaddvaq_p[_s8](mis2_t a, mi8x10_t b, mve_pred16_t p)	b -> Qm	VMSK FO,KP VPST	Kua -> iesuit	WIVE
mve_preare_c p/	p -> Rp	VADDVAT.S8 Rda,Qm		
int32_t [arm_]vaddvaq_p[_s16](int32_t a, int16x8_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm	VPST		
int32_t [arm_]vaddvaq_p[_s32](int32_t a, int32x4_t b,	p -> Rp a -> Rda	VADDVAT.S16 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
mis2_t [aim_]vaddvaq_p[_s32](mis2_t a, mis2x4_t b, mve_pred16_t p)	b -> Qm	VMSK FO,KP VPST	Kua -> iesuit	WIVE
	p -> Rp	VADDVAT.S32 Rda,Qm		
uint32_t [arm_]vaddvaq_p[_u8](uint32_t a, uint8x16_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
uint32_t [arm_]vaddvaq_p[_u16](uint32_t a, uint16x8_t	p -> Rp a -> Rda	VADDVAT.U8 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VMSK PO,KP VPST	Rua -> resuit	MIVE
c,	p -> Rp	VADDVAT.U16 Rda,Qm		
uint32_t [arm_]vaddvaq_p[_u32](uint32_t a, uint32x4_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
int32_t [arm_]vaddvq[_s8](int8x16_t a)	p -> Rp a -> Qm	VADDVAT.U32 Rda,Qm VADDV.S8 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvq[_s6](int6x16_t a)	a -> Qm	VADDV.S6 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvq[_s32](int32x4_t a)	a -> Qm	VADDV.S32 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vaddvq[_u8](uint8x16_t a)	a -> Qm	VADDV.U8 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vaddvq[_u16](uint16x8_t a)	a -> Qm	VADDV.U16 Rda,Qm	Rda -> result	MVE
uint32_t [_arm_]vaddvq[_u32](uint32x4_t a)	a -> Qm	VADDV.U32 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvq_p[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Rda -> result	MVE
prouto_t p/	l b > 1.h	VADDVT.S8 Rda,Qm		
int32_t [arm_]vaddvq_p[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	p -> <b>R</b> p	VPST VADDVT.S16 Rda,Qm		
int32_t [arm_]vaddvq_p[_s32](int32x4_t a,	a -> Qm	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Nuu > 103uit	
wint22 to a man broaddyn mf 01/1010 to	a > Om:	VADDVT.S32 Rda,Qm	D.do >14	MVE
uint32_t [arm_]vaddvq_p[_u8](uint8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Rda -> result	MVE
prouto_t p/	l b > 1.h	VADDVT.U8 Rda,Qm		
uint32_t [arm_]vaddvq_p[_u16](uint16x8_t a,	a -> Qm	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VADDVT.U16 Rda,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32_t [_arm_]vaddvq_p[_u32](uint32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VADDVT.U32 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaq[_s8](int32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVA.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaq[_s16](int32_t a, int16x8_t b, int16x8_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVA.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaq[_s32](int32_t a, int32x4_t b, int32x4_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVA.S32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq[_u8](uint32_t a, uint8x16_t b, uint8x16_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVA.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq[_u16](uint32_t a, uint16x8_t b, uint16x8_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVA.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq[_u32](uint32_t a, uint32x4_t b, uint32x4_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVA.U32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [_arm_]vmladavaq_p[_s8](int32_t a, int8x16_t b, int8x16_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm	VMSR P0,Rp VPST VMLADAVAT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaq_p[_s16](int32_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	p -> Rp a -> Rda b -> Qn c -> Qm	VMSR P0,Rp VPST VMLADAVAT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaq_p[_s32](int32_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	p -> Rp a -> Rda b -> Qn c -> Qm	VMSR P0,Rp VPST VMLADAVAT.S32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq_p[_u8](uint32_t a, uint8x16_t b, uint8x16_t c, mve_pred16_t p)	p -> Rp a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAT.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq_p[_u16](uint32_t a, uint16x8_t b, uint16x8_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAT.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq_p[_u32](uint32_t a, uint32x4_t b, uint32x4_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm	VMSR P0,Rp VPST VMLADAVAT.U32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq[_s8](int8x16_t a, int8x16_t b)	p -> Rp a -> Qn b -> Qm	VMLADAV.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMLADAV.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMLADAV.S32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [arm_]vmladavq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMLADAV.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [arm_]vmladavq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMLADAV.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [arm_]vmladavq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMLADAV.U32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq_p[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [_arm_]vmladavq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVT.S32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavq_p[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVT.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavq_p[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVT.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavq_p[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVT.U32 Rda,Qn,Qm	Rda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32_t [arm_]vmladavaxq[_s8](int32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVAX.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaxq[_s16](int32_t a, int16x8_t b, int16x8_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVAX.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaxq[_s32](int32_t a, int32x4_t b, int32x4_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVAX.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaxq_p[_s8](int32_t a, int8x16_t b, int8x16_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAXT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaxq_p[_s16](int32_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAXT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [_arm_]vmladavaxq_p[_s32](int32_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAXT.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavxq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMLADAVX.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavxq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMLADAVX.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Om	VMLADAVX.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [_arm_]vmladavxq_p[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVXT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [_arm_]vmladavxq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVXT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [_arm_]vmladavxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVXT.S32 Rda,Qn,Qm	Rda -> result	MVE
int64_t [arm_]vmlaldavaq[_s16](int64_t a, int16x8_t b, int16x8_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVA.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavaq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVA.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vmlaldavaq[_u16](uint64_t a, uint16x8_t b, uint16x8_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVA.U16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vmlaldavaq[_u32](uint64_t a, uint32x4_t b, uint32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVA.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavaq_p[_s16](int64_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavaq_p[_s32](int64_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vmlaldavaq_p[_u16](uint64_t a, uint16x8_t b, uint16x8_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAT.U16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vmlaldavaq_p[_u32](uint64_t a, uint32x4_t b, uint32x4_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAT.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMLALDAV.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMLALDAV.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64_t [arm_]vmlaldavq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMLALDAV.U16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vmlaldavq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMLALDAV.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo]	MVE
int64_t [_arm_]vmlaldavq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vmlaldavq_p[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVT.U16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vmlaldavq_p[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVT.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavaxq[_s16](int64_t a, int16x8_t b, int16x8_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVAX.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavaxq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVAX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavaxq_p[_s16](int64_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAXT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavaxq_p[_s32](int64_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAXT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavxq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMLALDAVX.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMLALDAVX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavxq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVXT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVXT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int8x16_t [arm_]vmlaq[_n_s8](int8x16_t a, int8x16_t b, int8_t c)	a -> Qda b -> Qn c -> Rm	VMLA.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vmlaq[_n_s16](int16x8_t a, int16x8_t b, int16_t c)	a -> Qda b -> Qn c -> Rm	VMLA.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vmlaq[_n_s32](int32x4_t a, int32x4_t b, int32_t c)	a -> Qda b -> Qn c -> Rm	VMLA.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vmlaq[_n_u8](uint8x16_t a, uint8x16_t b, uint8_t c)	a -> Qda b -> Qn c -> Rm	VMLA.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [_arm_]vmlaq[_n_u16](uint16x8_t a, uint16x8_t b, uint16_t c)	a -> Qda b -> Qn c -> Rm	VMLA.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vmlaq[_n_u32](uint32x4_t a, uint32x4_t b, uint32_t c)	a -> Qda b -> Qn c -> Rm	VMLA.U32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [_arm_]vmlaq_m[_n_s8](int8x16_t a, int8x16_t b, int8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAT.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vmlaq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAT.S16 Qda,Qn,Rm	Qda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [arm_]vmlaq_m[_n_s32](int32x4_t a, int32x4_t b, int32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm	VMSR P0,Rp VPST VMLAT.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vmlaq_m[_n_u8](uint8x16_t a, uint8x16_t b, uint8_t c, mve_pred16_t p)	p -> Rp a -> Qda b -> Qn c -> Rm	VMSR P0,Rp VPST VMLAT.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [arm_]vmlaq_m[_n_u16](uint16x8_t a, uint16x8_t b, uint16_t c, mve_pred16_t p)	p -> Rp a -> Qda b -> Qn c -> Rm	VMSR P0,Rp VPST VMLAT.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [arm_]vmlaq_m[_n_u32](uint32x4_t a, uint32x4_t b, uint32_t c, mve_pred16_t p)	p -> Rp a -> Qda b -> Qn c -> Rm	VMSR P0,Rp VPST VMLAT.U32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [arm_]vmlasq[_n_s8](int8x16_t a, int8x16_t b, int8_t c)	p -> Rp a -> Qda b -> Qn c -> Rm	VMLAS.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vmlasq[_n_s16](int16x8_t a, int16x8_t b, int16_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [_arm_]vmlasq[_n_s32](int32x4_t a, int32x4_t b, int32_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [arm_]vmlasq[_n_u8](uint8x16_t a, uint8x16_t b, uint8_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [arm_]vmlasq[_n_u16](uint16x8_t a, uint16x8_t b, uint16_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vmlasq[_n_u32](uint32x4_t a, uint32x4_t b, uint32_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.U32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [_arm_]vmlasq_m[_n_s8](int8x16_t a, int8x16_t b, int8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vmlasq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [_arm_]vmlasq_m[_n_s32](int32x4_t a, int32x4_t b, int32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vmlasq_m[_n_u8](uint8x16_t a, uint8x16_t b, uint8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [arm_]vmlasq_m[_n_u16](uint16x8_t a, uint16x8_t b, uint16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [arm_]vmlasq_m[_n_u32](uint32x4_t a, uint32x4_t b, uint32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.U32 Qda,Qn,Rm	Qda -> result	MVE
int32_t [arm_]vmlsdavaq[_s8](int32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn c -> Qm	VMLSDAVA.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavaq[_s16](int32_t a, int16x8_t b, int16x8_t c)	a -> Rda b -> Qn c -> Qm	VMLSDAVA.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavaq[_s32](int32_t a, int32x4_t b, int32x4_t c)	a -> Rda b -> Qn c -> Qm	VMLSDAVA.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavaq_p[_s8](int32_t a, int8x16_t b, int8x16_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVAT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavaq_p[_s16](int32_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVAT.S16 Rda,Qn,Qm	Rda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32_t [arm_]vmlsdavaq_p[_s32](int32_t a, int32x4_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, int32x4_t c, mve_pred16_t p)	b -> Qn c -> Qm p -> Rp	VPST VMLSDAVAT.S32 Rda,Qn,Qm		
int32_t [arm_]vmlsdavq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMLSDAV.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMLSDAV.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMLSDAV.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavq_p[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
o, mvc_predro_t p)	p -> Rp	VMLSDAVT.S8 Rda,Qn,Qm		
int32_t [arm_]vmlsdavq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavq_p[_s32](int32x4_t a, int32x4_t	a -> Qn	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMLSDAVT.S32 Rda,Qn,Qm		
int32_t [arm_]vmlsdavaxq[_s8](int32_t a, int8x16_t b,	a -> Rda	VMLSDAVAX.S8 Rda,Qn,Qm	Rda -> result	MVE
int8x16_t c)	b -> Qn c -> Qm			
int32_t [_arm_]vmlsdavaxq[_s16](int32_t a, int16x8_t b, int16x8_t c)	a -> Rda b -> Qn c -> Qm	VMLSDAVAX.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavaxq[_s32](int32_t a, int32x4_t b,	a -> Rda	VMLSDAVAX.S32 Rda,Qn,Qm	Rda -> result	MVE
int32x4_t c)	b -> Qn c -> Qm			
int32_t [_arm_]vmlsdavaxq_p[_s8](int32_t a, int8x16_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, int8x16_t c, mve_pred16_t p)	b -> Qn c -> Qm p -> Rp	VPST VMLSDAVAXT.S8 Rda,Qn,Qm		
int32_t [arm_]vmlsdavaxq_p[_s16](int32_t a, int16x8_t	$a \rightarrow Rda$	VMSR P0,Rp	Rda -> result	MVE
b, int16x8_t c, mve_pred16_t p)	b -> Qn c -> Qm	VPST VMLSDAVAXT.S16		
	p -> Rp	Rda,Qn,Qm		
int32_t [_arm_]vmlsdavaxq_p[_s32](int32_t a, int32x4_t	a -> Rda	VMSR P0,Rp VPST	Rda -> result	MVE
b, int32x4_t c, mve_pred16_t p)	b -> Qn c -> Qm	VMLSDAVAXT.S32		
: 422 4 [ ]	p -> Rp	Rda,Qn,Qm	D.d. v	MVE
int32_t [_arm_]vmlsdavxq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMLSDAVX.S8 Rda,Qn,Qm	Rda -> result	
int32_t [arm_]vmlsdavxq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMLSDAVX.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMLSDAVX.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavxq_p[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
o, mvc_predio_t p)	p -> Rp	VMLSDAVXT.S8 Rda,Qn,Qm		
int32_t [arm_]vmlsdavxq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
	p -> Rp	VMLSDAVXT.S16 Rda,Qn,Qm		
int32_t [arm_]vmlsdavxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
int64_t [arm_]vmlsldavaq[_s16](int64_t a, int16x8_t b,	p -> Rp a ->	VMLSDAVXT.S32 Rda,Qn,Qm VMLSLDAVA.S16	[RdaHi,RdaLo]	MVE
int16x8_t c)	[RdaHi,RdaLo] b -> Qn	RdaLo,RdaHi,Qn,Qm	-> result	WVL
int64 t [ arm ]vmlsldavaq[ s32](int64 t a, int32x4 t b,	c -> Qm a ->	VMLSLDAVA.S32	[RdaHi,RdaLo]	MVE
int32x4_t c)	[RdaHi,RdaLo] b -> Qn	RdaLo,RdaHi,Qn,Qm	-> result	WIVE
int64_t [arm_]vmlsldavaq_p[_s16](int64_t a, int16x8_t	c -> Qm a ->	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
b, int16x8_t c, mve_pred16_t p)	[RdaHi,RdaLo]	VPST	-> result	
	b -> Qn c -> Qm p -> Rp	VMLSLDAVAT.S16 RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlsldavaq_p[_s32](int64_t a, int32x4_t	a ->	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
b, int32x4_t c, mve_pred16_t p)	[RdaHi,RdaLo] b -> Qn	VPST VMLSLDAVAT.S32	-> result	
	c -> Qm	RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlsldavq[_s16](int16x8_t a, int16x8_t	p -> Rp a -> Qn	VMLSLDAV.S16	[RdaHi,RdaLo]	MVE
b) int64_t [arm_]vmlsldavq[_s32](int32x4_t a, int32x4_t	b -> Qm a -> Qn	RdaLo,RdaHi,Qn,Qm VMLSLDAV.S32	-> result [RdaHi,RdaLo]	MVE
b)	b -> Qm	RdaLo,RdaHi,Qn,Qm	-> result	

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64_t [arm_]vmlsldavq_p[_s16](int16x8_t a, int16x8_t	a -> Qn	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMLSLDAVT.S16 RdaLo,RdaHi,On,Om	-> result	
int64_t [arm_]vmlsldavq_p[_s32](int32x4_t a, int32x4_t	a -> Qn	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
b, mve_pred16_t p)	b -> Qm	VPST VMLSLDAVT.S32	-> result	
	p -> Rp	RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlsldavaxq[_s16](int64_t a, int16x8_t b,	a ->	VMLSLDAVAX.S16	[RdaHi,RdaLo]	MVE
int16x8_t c)	[RdaHi,RdaLo] b -> On	RdaLo,RdaHi,Qn,Qm	-> result	
	c -> Qm			
int64_t [arm_]vmlsldavaxq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo]	VMLSLDAVAX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
11.022.4_(0)	b -> Qn	realDo,realTi,Qii,Qiii	> resure	
int64_t [arm_]vmlsldavaxq_p[_s16](int64_t a, int16x8_t	c -> Qm a ->	VMSR P0,Rp	[PdoH; PdoLo]	MVE
b, int16x8_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo]	VMSK PU,KP VPST	[RdaHi,RdaLo] -> result	MVE
, <u> </u>	b -> Qn	VMLSLDAVAXT.S16		
	c -> Qm p -> Rp	RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlsldavaxq_p[_s32](int64_t a, int32x4_t	a ->	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
b, int32x4_t c, mve_pred16_t p)	[RdaHi,RdaLo] b -> On	VPST VMLSLDAVAXT.S32	-> result	
	c -> Qm	RdaLo,RdaHi,Qn,Qm		
int64 t [ arm ]vmlsldavxq[ s16](int16x8 t a, int16x8 t	p -> Rp a -> Qn	VMLSLDAVX.S16	[RdaHi,RdaLo]	MVE
b)	b -> Qm	RdaLo,RdaHi,Qn,Qm	-> result	IVI V E
int64_t [arm_]vmlsldavxq[_s32](int32x4_t a, int32x4_t	a -> Qn	VMLSLDAVX.S32	[RdaHi,RdaLo]	MVE
b) int64_t [arm_]vmlsldavxq_p[_s16](int16x8_t a,	b -> Qm a -> Qn	RdaLo,RdaHi,Qn,Qm VMSR P0,Rp	-> result [RdaHi,RdaLo]	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST	-> result	
	p -> Rp	VMLSLDAVXT.S16 RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlsldavxq_p[_s32](int32x4_t a,	a -> Qn	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMLSLDAVXT.S32	-> result	
	p -> <b>K</b> p	RdaLo,RdaHi,Qn,Qm		
int8x16_t [arm_]vhaddq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VHADD.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vhaddq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VHADD.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [arm_]vhaddq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VHADD.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vhaddq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VHADD.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vhaddq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VHADD.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [arm_]vhaddq[_n_u32](uint32x4_t a,	a -> Qn	VHADD.U32 Qd,Qn,Rm	Qd -> result	MVE
uint32_t b) int8x16_t [arm_]vhaddq[_s8](int8x16_t a, int8x16_t b)	b -> Rm a -> Qn	VHADD.S8 Qd,Qn,Qm	Od -> result	MVE/NEON
int16x8_t [_arm_]vhaddq[_s16](int16x8_t a, int16x8_t b)	b -> Qm	VHADD.S16 Qd,Qn,Qm	0.1 >16	MATE/ATEON
	a -> Qn b -> Qm	2,72,72	Qd -> result	MVE/NEON
int32x4_t [arm_]vhaddq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VHADD.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vhaddq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VHADD.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vhaddq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VHADD.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vhaddq[_u32](uint32x4_t a,	a -> Qn	VHADD.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t b) int8x16_t [_arm_]vhaddq_m[_n_s8](int8x16_t inactive,	b -> Qm inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm p -> Rp	VHADDT.S8 Qd,Qn,Rm		
int16x8_t [arm_]vhaddq_m[_n_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VHADDT.S16 Qd,Qn,Rm		
	p -> Rp			
int32x4_t [arm_]vhaddq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
mt32x+_t a, mt32_t b, mvc_pred10_t p)	a -> Qn b -> Rm	VHADDT.S32 Qd,Qn,Rm		
	p -> Rp			

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [arm_]vhaddq_m[_n_u8](uint8x16_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VHADDT.U8 Qd,Qn,Rm		
uint16x8_t [arm_]vhaddq_m[_n_u16](uint16x8_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> On	VMSK FO,KP VPST	Qu -> resuit	IVIVE
	b -> Rm p -> Rp	VHADDT.U16 Qd,Qn,Rm		
uint32x4_t [arm_]vhaddq_m[_n_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VHADDT.U32 Qd,Qn,Rm		
	p -> Rm p -> Rp	VHADD1:032 Qu,Qii,Kiii		
int8x16_t [arm_]vhaddq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VHADDT.S8 Qd,Qn,Qm		
	p -> Rp	VIIADD1.38 Qu,Qii,Qiii		
int16x8_t [arm_]vhaddq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VHADDT.S16 Qd,Qn,Qm		
int32x4_t [arm_]vhaddq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VHADDT.S32 Qd,Qn,Qm		
uint8x16_t [arm_]vhaddq_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VHADDT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vhaddq_m[_u16](uint16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn	VPST	<b>Q</b> 2	
•	b -> Qm	VHADDT.U16 Qd,Qn,Qm		
uint32x4_t [arm_]vhaddq_m[_u32](uint32x4_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn	VPST	Qu > resurt	111 1 12
	b -> Qm	VHADDT.U32 Qd,Qn,Qm		
int8x16_t [arm_]vhaddq_x[_n_s8](int8x16_t a, int8_t b,	p -> Rp a -> On	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	b -> Rm	VPST	Qu -> resuit	WIVE
	p -> Rp	VHADDT.S8 Qd,Qn,Rm		
int16x8_t [arm_]vhaddq_x[_n_s16](int16x8_t a, int16_t	a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm p -> Rp	VHADDT.S16 Qd,Qn,Rm		
int32x4_t [arm_]vhaddq_x[_n_s32](int32x4_t a, int32_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST		
uint8x16_t [arm_]vhaddq_x[_n_u8](uint8x16_t a,	p -> Rp a -> Qn	VHADDT.S32 Qd,Qn,Rm VMSR P0,Rp	Od -> result	MVE
uint8_t b, mve_pred16_t p)	b -> Rm	VPST	Qu' y Tesun	11112
	p -> Rp	VHADDT.U8 Qd,Qn,Rm		
uint16x8_t [arm_]vhaddq_x[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Qd -> result	MVE
unitro_t b, inve_predio_t p)	p -> Rn	VHADDT.U16 Qd,Qn,Rm		
uint32x4_t [arm_]vhaddq_x[_n_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VHADDT.U32 Qd,Qn,Rm		
int8x16 t [ arm ]vhaddq x[ s8](int8x16 t a, int8x16 t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST	<b>Q</b> 2	
116.0 . 1 11.11 1.16/6 . 16.0	p -> Rp	VHADDT.S8 Qd,Qn,Qm	0.1	NOTE
int16x8_t [arm_]vhaddq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
b, inve_predio_t p)	p -> Rp	VHADDT.S16 Qd,Qn,Qm		
int32x4_t [arm_]vhaddq_x[_s32](int32x4_t a, int32x4_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VHADDT.S32 Qd,Qn,Qm		
uint8x16_t [arm_]vhaddq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
uint16x8 t [ arm ]vhaddq x[ u16](uint16x8 t a,	p -> Rp a -> Qn	VHADDT.U8 Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
uint16x8_t [arm_]vnaddq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR PU,RP VPST	Qu -> resuit	IVI V E
_ · _ · _ · _ · .	p -> Rp	VHADDT.U16 Qd,Qn,Qm		
uint32x4_t [arm_]vhaddq_x[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
		VPST	1	1
uint32x4_t [arm_]vhaddq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	b -> Qm n -> Rn			
	b -> Qm p -> Rp a -> Qn	VHADDT.U32 Qd,Qn,Qm VHCADD.S8 Qd,Qn,Qm,#90	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	p -> Rp	VHADDT.U32 Qd,Qn,Qm	Qd -> result  Qd -> result	MVE MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [arm_]vhcaddq_rot90[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VHCADD.S32 Qd,Qn,Qm,#90	Qd -> result	MVE
int8x16_t [_arm_]vhcaddq_rot90_m[_s8](int8x16_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn	VPST VHCADDT.S8 Qd,Qn,Qm,#90		
	b -> Qm p -> Rp	VHCADD1.38 Qu,Qii,Qiii,#90		
int16x8_t [arm_]vhcaddq_rot90_m[_s16](int16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VHCADDT.S16 Qd,Qn,Qm,#90		
	p -> Rp	2 . 2 . 2 .		
int32x4_t [arm_]vhcaddq_rot90_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, misza-t a, misza-t b, mve_pred10_t p)	b -> Qm	VHCADDT.S32 Qd,Qn,Qm,#90		
int8x16_t [arm_]vhcaddq_rot90_x[_s8](int8x16_t a,	p -> Rp a -> Qn	VMSR P0,Rp	Od -> result	MVE
int8x16_t [armjvncaddq_10t90_x[_s8](int8x10_t a, int8x16_t b, mve_pred16_t p)	b -> Qm	VPST	Qu -> result	WIVE
	p -> Rp	VHCADDT.S8 Qd,Qn,Qm,#90	0.1	NA TO
int16x8_t [arm_]vhcaddq_rot90_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VHCADDT.S16 Qd,Qn,Qm,#90		
int32x4_t [arm_]vhcaddq_rot90_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VHCADDT.S32 Qd,Qn,Qm,#90		
int8x16_t [arm_]vhcaddq_rot270[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VHCADD.S8 Qd,Qn,Qm,#270	Qd -> result	MVE
int16x8_t [arm_]vhcaddq_rot270[_s16](int16x8_t a,	a -> Qn	VHCADD.S16 Qd,Qn,Qm,#270	Qd -> result	MVE
int16x8_t b)	b -> Qm			
int32x4_t [arm_]vhcaddq_rot270[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VHCADD.S32 Qd,Qn,Qm,#270	Qd -> result	MVE
int8x16_t [arm_]vhcaddq_rot270_m[_s8](int8x16_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VHCADDT.S8 Qd,Qn,Qm,#270		
	p -> Rp	VIIC/IDD1.50 Qu,Qii,Qiii,#270		
int16x8_t [arm_]vhcaddq_rot270_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
macuve, mtroxo_t a, mtroxo_t b, mve_preuro_t p)	b -> Qm	VHCADDT.S16 Qd,Qn,Qm,#270		
	p -> Rp	Andap po p	0.1	NA TO
int32x4_t [arm_]vhcaddq_rot270_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
· · - · - · - · ·	b -> Qm	VHCADDT.S32 Qd,Qn,Qm,#270		
int8x16_t [arm_]vhcaddq_rot270_x[_s8](int8x16_t a,	p -> Rp a -> Qn	VMSR P0,Rp	Od -> result	MVE
int8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
int16x8_t [arm_]vhcaddq_rot270_x[_s16](int16x8_t a,	p -> Rp a -> Qn	VHCADDT.S8 Qd,Qn,Qm,#270 VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Qu'y resuit	111,2
int32x4_t [arm_]vhcaddq_rot270_x[_s32](int32x4_t a,	p -> Rp a -> Qn	VHCADDT.S16 Qd,Qn,Qm,#270 VMSR P0,Rp	Qd -> result	MVE
int32x4_t tarmfynedddq_f0t270_x[_332](int32x4_t a, int32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Qu => resuit	WYL
int8x16_t [arm_]vhsubq[_n_s8](int8x16_t a, int8_t b)	p -> Rp	VHCADDT.S32 Qd,Qn,Qm,#270 VHSUB.S8 Qd,Qn,Rm	Qd -> result	MVE
mtox10_t [arm_]viisubq[_n_so](mtox10_t a, mto_t b)	a -> Qn b -> Rm	VH3UB.36 Qu,Qii,Kiii	Qu -> result	WIVE
int16x8_t [arm_]vhsubq[_n_s16](int16x8_t a, int16_t b)	a -> Qn	VHSUB.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4 t [ arm ]vhsubq[ n s32](int32x4 t a, int32 t b)	b -> Rm a -> Qn	VHSUB.S32 Od,On,Rm	Qd -> result	MVE
	b -> Rm			
uint8x16_t [arm_]vhsubq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VHSUB.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [arm_]vhsubq[_n_u16](uint16x8_t a,	a -> Qn	VHSUB.U16 Qd,Qn,Rm	Qd -> result	MVE
uint16_t b) uint32x4 t [ arm ]vhsubq[ n u32](uint32x4 t a,	b -> Rm a -> Qn	VHSUB.U32 Qd,Qn,Rm	Qd -> result	MVE
uint32_t b)	b -> Rm	11150B.032 Qu,Qii,Niii		
int8x16_t [arm_]vhsubq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VHSUB.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vhsubq[_s16](int16x8_t a, int16x8_t b)	b -> Qm a -> Qn	VHSUB.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vhsubq[_s32](int32x4_t a, int32x4_t b)	b -> Qm	VHSUB.S32 Qd,Qn,Qm	Od -> result	MVE/NEON
mt32x4_t [atm_jvnsubq[_s32j(mt32x4_t a, mt32x4_t b)	a -> Qn b -> Qm	v Haub.aa2 Qu,Qii,Qm	Qu -> resuit	IVI V E/INEUN
uint8x16_t [arm_]vhsubq[_u8](uint8x16_t a, uint8x16_t	a -> Qn	VHSUB.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
b) uint16x8_t [arm_]vhsubq[_u16](uint16x8_t a,	b -> Qm a -> Qn	VHSUB.U16 Qd,Qn,Qm	Od -> result	MVE/NEON
uint16x8_t b)	b -> Qm	7,7,7	Ì	
uint32x4_t [arm_]vhsubq[_u32](uint32x4_t a,	a -> Qn	VHSUB.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [_arm_]vhsubq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHSUBT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [_arm_]vhsubq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHSUBT.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vhsubq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHSUBT.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vhsubq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHSUBT.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vhsubq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHSUBT.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vhsubq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHSUBT.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [_arm_]vhsubq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHSUBT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vhsubq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHSUBT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vhsubq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHSUBT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vhsubq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHSUBT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vhsubq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHSUBT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vhsubq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHSUBT.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vhsubq_x[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHSUBT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [_arm_]vhsubq_x[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p) int32x4_t [_arm_]vhsubq_x[_n_s32](int32x4_t a, int32_t	a -> Qn b -> Rm p -> Rp a -> Qn	VMSR P0,Rp VPST VHSUBT.S16 Qd,Qn,Rm VMSR P0,Rp	Qd -> result  Qd -> result	MVE MVE
b, mve_pred16_t p)  uint8x16_t [arm_]vhsubq_x[_n_u8](uint8x16_t a,	b -> Rm p -> Rp a -> On	VMSR PO,RP  VPST  VHSUBT.S32 Qd,Qn,Rm  VMSR PO.Rp	Qd -> result	MVE
uint8_t b, mve_pred16_t p)  uint16x8_t [_arm_]vhsubq_x[_n_u16](uint16x8_t a,	b -> Rm p -> Rp a -> Qn	VPST VHSUBT.U8 Qd,Qn,Rm VMSR P0,Rp	Qd -> result	MVE
uint16_t b, mve_pred16_t p) uint32x4_t [_arm_]vhsubq_x[_n_u32](uint32x4_t a,	b -> Rm p -> Rp a -> Qn	VPST VHSUBT.U16 Qd,Qn,Rm VMSR P0,Rp	Qd -> result	MVE
uint32_t b, mve_pred16_t p)  int8x16_t [_arm_]vhsubq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	b -> Rm p -> Rp a -> Qn b > Qm	VPST VHSUBT.U32 Qd,Qn,Rm VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)  int16x8_t [_arm_]vhsubq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp a -> Qn b -> Qm p -> Rp	VPST VHSUBT.S8 Qd,Qn,Qm VMSR P0,Rp VPST VHSUBT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vhsubq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm p -> Rp	VHSUBT.S16 Qd,Qn,Qm  VMSR P0,Rp  VPST  VHSUBT.S32 Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [arm_]vhsubq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VHSUBT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vhsubq_x[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
1.00 4.7	p -> Rp	VHSUBT.U16 Qd,Qn,Qm	0.1	) am
uint32x4_t [arm_]vhsubq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
unit32x4_t b, nive_pred10_t p)	p -> QIII	VHSUBT.U32 Qd,Qn,Qm		
int8x16_t [arm_]vrhaddq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VRHADD.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
	b -> Qm			
int16x8_t [arm_]vrhaddq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VRHADD.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vrhaddq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VRHADD.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vrhaddq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VRHADD.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vrhaddq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VRHADD.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vrhaddq[_u32](uint32x4_t a,	a -> Qn	VRHADD.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t b)	b -> Qm	VACO DO D	0.1	MATE
int8x16_t [arm_]vrhaddq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
mtox10_t a, mtox10_t b, mve_pred10_t p)	b -> Qm p -> Rp	VRHADDT.S8 Qd,Qn,Qm		
int16x8_t [arm_]vrhaddq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VRHADDT.S16 Qd,Qn,Qm		
int32x4_t [_arm_]vrhaddq_m[_s32](int32x4_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> On	VNSK FO,KP VPST	Qu -> resuit	MIVE
	b -> Qm	VRHADDT.S32 Qd,Qn,Qm		
	p -> Rp			
uint8x16_t [_arm_]vrhaddq_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VRHADDT.U8 Qd,Qn,Qm		
	p -> Rp	TRITIDD 1.00 Qu,Qii,Qiii		
uint16x8_t [arm_]vrhaddq_m[_u16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VRHADDT.U16 Qd,Qn,Qm		
uint32x4_t [arm_]vrhaddq_m[_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VRHADDT.U32 Qd,Qn,Qm		
int8x16_t [arm_]vrhaddq_x[_s8](int8x16_t a, int8x16_t	p -> Rp a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve pred16 t p)	b -> Qm	VPST	Qu > result	WYE
	p -> Rp	VRHADDT.S8 Qd,Qn,Qm		
int16x8_t [arm_]vrhaddq_x[_s16](int16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VRHADDT.S16 Qd,Qn,Qm		
int32x4_t [arm_]vrhaddq_x[_s32](int32x4_t a,	a -> Qn	VMSR P0,Rp	Od -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VRHADDT.S32 Qd,Qn,Qm		L Norm
uint8x16_t [arm_]vrhaddq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
umox10_t o, mvc_picu10_t p)	p -> Qm	VRHADDT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vrhaddq_x[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
win420w4 4.f. comp. br-4-44 f 202( ' - 202 4 -	p -> Rp	VRHADDT.U16 Qd,Qn,Qm	04 > 1:	MVE
uint32x4_t [_arm_]vrhaddq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
amoza (_t o, mvc_prodro_t p)	p -> Rp	VRHADDT.U32 Qd,Qn,Qm		
float16x8_t [arm_]vfmaq[_n_f16](float16x8_t a,	a -> Qda	VFMA.F16 Qda,Qn,Rm	Qda -> result	MVE/NEON
float16x8_t b, float16_t c)	b -> Qn			
float32x4_t [arm_]vfmaq[_n_f32](float32x4_t a,	c -> Rm a -> Qda	VFMA.F32 Qda,Qn,Rm	Qda -> result	MVE/NEON
float32x4_t [arm_]vrmaq[_n_132](float32x4_t a, float32x4_t b, float32_t c)	b -> Qua b -> Qn	VI MA.1 32 Qua,Qii,Kiii	Qua -> resuit	IVI V E/INEOIN
	c -> Rm			
float16x8_t [arm_]vfmaq_m[_n_f16](float16x8_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
float16x8_t b, float16_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Rm p -> Rp	VFMAT.F16 Qda,Qn,Rm		
float32x4_t [arm_]vfmaq_m[_n_f32](float32x4_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
float32x4_t b, float32_t c, mve_pred16_t p)	b -> Qn	VPST	2 > 100011	
	c -> Rm	VFMAT.F32 Qda,Qn,Rm		
	p -> Rp			

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [arm_]vfmaq[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Qm	VFMA.F16 Qda,Qn,Qm	Qda -> result	MVE/NEON
float32x4_t [_arm_]vfmaq[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VFMA.F32 Qda,Qn,Qm	Qda -> result	MVE/NEON
float16x8_t [arm_]vfmaq_m[_f16](float16x8_t a, float16x8_t b, float16x8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm	VMSR P0,Rp VPST VFMAT.F16 Qda,Qn,Qm	Qda -> result	MVE
float32x4_t [arm_]vfmaq_m[_f32](float32x4_t a, float32x4_t b, float32x4_t c, mve_pred16_t p)	p -> Rp a -> Qda b -> Qn c -> Qm	VMSR P0,Rp VPST VFMAT.F32 Qda,Qn,Qm	Qda -> result	MVE
float16x8_t [arm_]vfmasq[_n_f16](float16x8_t a, float16x8_t b, float16_t c)	p -> Rp a -> Qda b -> Qn c -> Rm	VFMAS.F16 Qda,Qn,Rm	Qda -> result	MVE
float32x4_t [arm_]vfmasq[_n_f32](float32x4_t a, float32x4_t b, float32_t c)	a -> Qda b -> Qn c -> Rm	VFMAS.F32 Qda,Qn,Rm	Qda -> result	MVE
float16x8_t [arm_]vfmasq_m[_n_f16](float16x8_t a, float16x8_t b, float16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VFMAST.F16 Qda,Qn,Rm	Qda -> result	MVE
float32x4_t [_arm_]vfmasq_m[_n_f32](float32x4_t a, float32x4_t b, float32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VFMAST.F32 Qda,Qn,Rm	Qda -> result	MVE
float16x8_t [arm_]vfmsq[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Qm	VFMS.F16 Qda,Qn,Qm	Qda -> result	MVE/NEON
float32x4_t [_arm_]vfmsq[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VFMS.F32 Qda,Qn,Qm	Qda -> result	MVE/NEON
float16x8_t [arm_]vfmsq_m[_f16](float16x8_t a, float16x8_t b, float16x8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VFMST.F16 Qda,Qn,Qm	Qda -> result	MVE
float32x4_t [arm_]vfmsq_m[_f32](float32x4_t a, float32x4_t b, float32x4_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VFMST.F32 Qda,Qn,Qm	Qda -> result	MVE
int64_t [arm_]vrmlaldavhaq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VRMLALDAVHA.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]vrmlaldavhaq[_u32](uint64_t a, uint32x4_t b, uint32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn	VRMLALDAVHA.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vrmlaldavhaq_p[_s32](int64_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	c -> Qm a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VRMLALDAVHAT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]vrmlaldavhaq_p[_u32](uint64_t a, uint32x4_t b, uint32x4_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VRMLALDAVHAT.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vrmlaldavhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VRMLALDAVH.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]vrmlaldavhq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VRMLALDAVH.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vrmlaldavhq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMLALDAVHT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vrmlaldavhq_p[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMLALDAVHT.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vrmlaldavhaxq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VRMLALDAVHAX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64_t [arm_]vrmlaldavhaxq_p[_s32](int64_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VRMLALDAVHAXT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vrmlaldavhxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VRMLALDAVHX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vrmlaldavhxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMLALDAVHXT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vrmlsldavhaq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VRMLSLDAVHA.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vrmlsldavhaq_p[_s32](int64_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VRMLSLDAVHAT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vrmlsldavhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VRMLSLDAVH.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vrmlsldavhq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMLSLDAVHT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vrmlsldavhaxq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VRMLSLDAVHAX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vrmlsldavhaxq_p[_s32](int64_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VRMLSLDAVHAXT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vrmlsldavhxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VRMLSLDAVHX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_tarm_]vrmlsldavhxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMLSLDAVHXT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int8x16_t [arm_]vrmulhq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VRMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vrmulhq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VRMULH.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vrmulhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VRMULH.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [arm_]vrmulhq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VRMULH.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vrmulhq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VRMULH.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vrmulhq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VRMULH.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vrmulhq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMULHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vrmulhq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMULHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vrmulhq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMULHT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vrmulhq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMULHT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vrmulhq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMULHT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vrmulhq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMULHT.U32 Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vrmulhq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t [_arm_]vrmulhq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VRMULHT.S8 Qd,Qn,Qm  VMSR P0,Rp  VPST	Qd -> result	MVE
int32x4_t [_arm_]vrmulhq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VRMULHT.S16 Qd,Qn,Qm  VMSR P0,Rp  VPST  VPMULHT S22 Qd On Om	Qd -> result	MVE
uint8x16_t [_arm_]vrmulhq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm p -> Rp	VRMULHT.S32 Qd,Qn,Qm  VMSR P0,Rp  VPST  VRMULHT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vrmulhq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMULHT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vrmulhq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRMULHT.U32 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vcvtaq_s16_f16(float16x8_t a)	a -> Qm	VCVTA.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vcvtaq_s32_f32(float32x4_t a)	a -> Qm	VCVTA.S32.F32 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vcvtaq_u16_f16(float16x8_t a)	a -> Qm	VCVTA.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vcvtaq_u32_f32(float32x4_t a)	a -> Qm	VCVTA.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vcvtaq_m[_s16_f16](int16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.S16.F16 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vcvtaq_m[_s32_f32](int32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.S32.F32 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vcvtaq_m[_u16_f16](uint16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.U16.F16 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vcvtaq_m[_u32_f32](uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.U32.F32 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vcvtaq_x_s16_f16(float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.S16.F16 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vcvtaq_x_s32_f32(float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.S32.F32 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vcvtaq_x_u16_f16(float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.U16.F16 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vcvtaq_x_u32_f32(float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.U32.F32 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vcvtnq_s16_f16(float16x8_t a)	a -> Qm	VCVTN.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vcvtnq_s32_f32(float32x4_t a)	a -> Qm	VCVTN.S32.F32 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vcvtnq_u16_f16(float16x8_t a)	a -> Qm	VCVTN.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vcvtnq_u32_f32(float32x4_t a)	a -> Qm	VCVTN.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vcvtnq_m[_s16_f16](int16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTNT.S16.F16 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vcvtnq_m[_s32_f32](int32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTNT.S32.F32 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vcvtnq_m[_u16_f16](uint16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTNT.U16.F16 Od,Om	Qd -> result	MVE
uint32x4_t [_arm_]vcvtnq_m[_u32_f32](uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t [arm_]vcvtnq_x_s16_f16(float16x8_t a, mve_pred16_t p)	p -> Rp a -> Qm p -> Rp	VCVTNT.U32.F32 Qd,Qm  VMSR P0,Rp  VPST  VCVTNT.S16.F16 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vcvtnq_x_s32_f32(float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VCVTNT.S10.F10 Qd,Qml  VMSR P0,Rp  VPST  VCVTNT.S32.F32 Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vcvtnq_x_u16_f16(float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VCVTNT.332.F32 Qd,Qmi  VMSR P0,Rp  VPST  VCVTNT.U16.F16 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vcvtnq_x_u32_f32(float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VCVTNT.U10.F10 Qd,Qm  VMSR P0,Rp  VPST  VCVTNT.U32.F32 Qd,Qm	Qd -> result	MVE
int16x8_t [_arm_]vcvtpq_s16_f16(float16x8_t a) int32x4_t [_arm_]vcvtpq_s32_f32(float32x4_t a)	a -> Qm a -> Qm	VCVTP.S16.F16 Qd,Qm VCVTP.S32.F32 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vcvtpq_u16_f16(float16x8_t a)	a -> Qm	VCVTP.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vcvtpq_u32_f32(float32x4_t a)	a -> Qm	VCVTP.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vcvtpq_m[_s16_f16](int16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VCVTPT.S16.F16 Qd,Qm		
int32x4_t [arm_]vcvtpq_m[_s32_f32](int32x4_t	inactive -> Od	VMSR P0,Rp	Od -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Qu > resuit	III V E
	p -> Rp	VCVTPT.S32.F32 Qd,Qm		
uint16x8_t [arm_]vcvtpq_m[_u16_f16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
uint32x4_t [arm_]vcvtpq_m[_u32_f32](uint32x4_t	p -> Rp inactive -> Qd	VCVTPT.U16.F16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Om	VMSK FO,KP VPST	Qu -> resuit	IVI V E
materie, notes ziviet a, mve_prouto_v p)	p -> Rp	VCVTPT.U32.F32 Qd,Qm		
int16x8_t [arm_]vcvtpq_x_s16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
20 00 00 00 00 00 00 00 00 00 00 00 00 0		VCVTPT.S16.F16 Qd,Qm	0.1	NO.
int32x4_t [_arm_]vcvtpq_x_s32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VCVTPT.S32.F32 Qd,Qm		
uint16x8_t [_arm_]vcvtpq_x_u16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu's resuit	111.12
	1 1	VCVTPT.U16.F16 Qd,Qm		
uint32x4_t [arm_]vcvtpq_x_u32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
16 0 4 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	. 0	VCVTPT.U32.F32 Qd,Qm	0.1	MUTATION
int16x8_t [arm_]vcvtmq_s16_f16(float16x8_t a) int32x4_t [arm_]vcvtmq_s32_f32(float32x4_t a)	a -> Qm	VCVTM.S16.F16 Qd,Qm VCVTM.S32.F32 Qd,Qm	Qd -> result Od -> result	MVE/NEON MVE/NEON
uint16x8_t [arm_]vcvtmq_s32_132(110at32x4_t a) uint16x8_t [arm_]vcvtmq_u16_f16(float16x8_t a)	a -> Qm a -> Qm	VCVTM.S32.F32 Qd,Qm VCVTM.U16.F16 Qd,Qm	Qd -> result	MVE/NEON MVE/NEON
uint32x4_t [_arm_]vcvtmq_u32_f32(float32x4_t a)	a -> Qm	VCVTM.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vcvtmq_m[_s16_f16](int16x8_t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		·
	p -> Rp	VCVTMT.S16.F16 Qd,Qm		
int32x4_t [arm_]vcvtmq_m[_s32_f32](int32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
uint16x8_t [arm_]vcvtmq_m[_u16_f16](uint16x8_t	p -> Rp inactive -> Qd	VCVTMT.S32.F32 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Om	VPST	Qu -> resuit	IVIVE
	p -> Rp	VCVTMT.U16.F16 Qd,Qm		
uint32x4_t [arm_]vcvtmq_m[_u32_f32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
1.160.15	p -> Rp	VCVTMT.U32.F32 Qd,Qm	0.1	Nam.
int16x8_t [_arm_]vcvtmq_x_s16_f16(float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
mvc_picuro_t p)	p -> Kp	VCVTMT.S16.F16 Qd,Qm		
int32x4 t [ arm ]vcvtmq x s32 f32(float32x4 t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		·
		VCVTMT.S32.F32 Qd,Qm		
uint16x8_t [arm_]vcvtmq_x_u16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VCVTMT.U16.F16 Qd,Qm		
uint32x4_t [arm_]vcvtmq_x_u32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve pred16 t p)	p -> Rp	VMSK FO,KP VPST	Qu → resuit	171 7 12
	r T	VCVTMT.U32.F32 Qd,Qm		
float16x8_t [arm_]vcvtbq_f16_f32(float16x8_t a,	a -> Qd	VCVTB.F16.F32 Qd,Qm	Qd -> result	MVE
float32x4_t b)	b -> Qm	VOLUMB FOR FILE S. C.		NO.
float16x8 t [_arm_]vcvtbq_f32_f16(float16x8_t a)	a -> Qm	VCVTB.F32.F16 Qd,Qm	Qd -> result	MVE
float16x8_t [arm_]vcvtbq_m_f16_f32(float16x8_t a, float32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
nouts2x4_t o, invo_prouto_t p)	p -> QIII	VCVTBT.F16.F32 Qd,Qm		
float32x4_t [arm_]vcvtbq_m_f32_f16(float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCVTBT.F32.F16 Qd,Qm		
float32x4_t [_arm_]vcvtbq_x_f32_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VCVTBT.F32.F16 Qd,Qm		
float16x8_t [arm_]vcvttq_f16_f32(float16x8_t a,	a -> Qd	VCVTB1.F32.F16 Qd,Qm VCVTT.F16.F32 Od.Om	Od -> result	MVE
float32x4_t b)	b -> Qm	7.5 7 1 1.1 10.1 32 Qu,QIII	Qu -> icsuit	141 4 1
float32x4_t [arm_]vcvttq_f32_f16(float16x8_t a)	a -> Qm	VCVTT.F32.F16 Qd,Qm	Qd -> result	MVE
float16x8_t [_arm_]vcvttq_m_f16_f32(float16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
0.004.1	p -> Rp	VCVTTT.F16.F32 Qd,Qm		\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
float32x4_t [_arm_]vcvttq_m_f32_f16(float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST VCVTTT.F32.F16 Qd,Qm		
	p -> Rp	v C v 1 1 1 .F32.F10 Qa,Qm	1	

Boat 56.4.1_am_jeveq_x_f2_f2_f16(bat16x8_1a)   a> Om   WSR_RNRp   Qd > result   MVE_NFON   Veve_NFT_NFT_NFT_NFT_NFT_NFT_NFT_NFT_NFT_NFT	Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
		-	VPST	Qd -> result	MVE
	fl-416-9 4 [ ]4-[ f16 -16](416-9 4)	0		0.1 >	MVENIEON
float 168.1  am]vevaq_ml_f16_s16[float 168.1   anpvetaq_ml_f16_s16[float 168.1   anpvetaq_ml_f18_s2_s2][float 324.1   anpvetaq_ml_f18_s2_s2][float 324.1   anpvetaq_ml_f18_s2_s2][float 324.1   anpvetaq_ml_f18_s2_s2][float 324.1   anpvetaq_ml_f18_s2_s2][float 324.1   anpvetaq_ml_f18_s2_s2][float 324.1   anpvetaq_ml_f18_s16[im168_s1 a   a > 0mpvetaq_ml_f18_s16[im168_s1 a   a > 0mpvetaq_ml_f18_s16[im168_s1 a   a > 0mpvetaq_ml_f18_s18[im168_s1 a   a > 0mpvetaq_ml_f18_s18[im					
Double   D				Qu > resuit	141 4 12
float 16x8_famm_lvevtq_m[_f32_s32](float32x4_t)					
float324_f  amm_jvevtq_mf_f32_a32](float324_f  amm_jvevtq_mf_f		inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
Most	flt22-4 + [ ]t [ f22 -22](flt22-4 +			0.1 >16	MAZE
float244_tamm_jvevtq_xt_nfl_6_16_0tif(int16x8_t a, a)	inactive, int32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Qu -> resuit	MVE
inactive, uinf32x4_1_s, mev_pred16_tp)	float32x4_t [arm_]vcvtq_m[_f32_u32](float32x4_t	inactive -> Qd		Qd -> result	MVE
float168.8   _amm_ vevtq_x _file_s16 (int168.1 a, a)		a -> Qm	VPST		
mve_pred16_t p)					
New_pred16_tp   New_pred16_t		-		Qd -> result	MVE
a > 0m	mve_pred16_t p)	p -> Rp			
P > Rp	0.160.150.150.160.160.160.160.160.160.160.160.160.16			0.1	) a m
New   New		•	, 1	Qd -> result	MVE
## Processor   ## P	lilve_pred16_t p)	p -> <b>K</b> p			
P > Rp	float32v4 t [ arm ]vevta v[ f32 s32](int32v4 t a	a -> Om		Od -> result	MVF
New   New		•		Qu -> result	MAL
	e_prearo_r.p/	Prin			
Note	float32x4_t [arm_]vcvtq_x[_f32_u32](uint32x4_t a,	a -> Qm		Qd -> result	MVE
1 <= imm6 <	- <b>-</b> - <b>- .</b> ,	p -> Rp	VCVTT.F32.U32 Qd,Qm		
16			VCVT.F16.S16 Qd,Qm,imm6	Qd -> result	MVE/NEON
Total 16x8_t   _arm_ vcvtq_n[_f16_u16](uint16x8_t a, const int imm6)	int imm6)				
1   2   2   2   2   3   3   3   4   1   2   3   3   3   3   3   3   3   3   3	float16-9 t f and loants of f16 -161/-int16-9 t		VOVT E16 U16 O4 O··· imme	0.1 >16	MATERIEON
It = imm6 \corr   32		1 <= imm6 <=	VCV1.F16.U16 Qd,Qm,imm6	Qd -> result	MVE/NEON
1 <= imm6 <= 32		1 <= imm6 <=	VCVT.F32.S32 Qd,Qm,imm6	Qd -> result	MVE/NEON
Inactive   Float   Inactive   I		1 <= imm6 <=	VCVT.F32.U32 Qd,Qm,imm6	Qd -> result	MVE/NEON
Inactive   Inactive		inactive -> Qd a -> Qm 1 <= imm6 <= 16	VPST	Qd -> result	MVE
Inactive, uint16x8_t a, const int imm6, mve_pred16_t p)	float16x8 t [ arm ]vcvtq m n[ f16 u16](float16x8 t		VMSR P0 Rn	Od -> result	MVE
16				(	
float32x4_t [_arm_]vcvtq_m_n[_f32_s32](float32x4_t inactive, int32x4_t a, const int imm6, mve_pred16_t p)		16	VCVTT.F16.U16 Qd,Qm,imm6		
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	float32v4 t [ arm lyoyta m n[ f22 s22](float22v4 t		VMSP PO Pp	Od > rogult	MVE
float32x4_t [_arm_]vcvtq_m_n[_f32_u32](float32x4_t inactive, uint32x4_t a, const int imm6, mve_pred16_t p)		a -> Qm 1 <= imm6 <=	VPST	Qu -> resuit	MVE
$ \begin{array}{c} inactive, uint32x4\_t\ a, const\ int\ imm6, \ mve\_pred16\_t\ p) \\ 1 <= imm6 <= \\ 32 \\ p >> Rp \\ \\ float16x8\_t\ [\_arm\_]vcvtq\_x\_n[\_f16\_s16](int16x8\_t\ a, \\ const\ int\ imm6, \ mve\_pred16\_t\ p) \\ 1 <= imm6 <= \\ 16 \\ p >> Rp \\ \\ float16x8\_t\ [\_arm\_]vcvtq\_x\_n[\_f16\_u16](uint16x8\_t\ a, \\ const\ int\ imm6, \ mve\_pred16\_t\ p) \\ 1 <= imm6 <= \\ 16 \\ p >> Rp \\ \\ float32x4\_t\ [\_arm\_]vcvtq\_x\_n[\_f16\_u16](uint16x8\_t\ a, \\ const\ int\ imm6, \ mve\_pred16\_t\ p) \\ 1 <= imm6 <= \\ 16 \\ p >> Rp \\ \\ float32x4\_t\ [\_arm\_]vcvtq\_x\_n[\_f32\_s32](int32x4\_t\ a, \\ const\ int\ imm6, \ mve\_pred16\_t\ p) \\ 32 \\ \end{array} \begin{array}{c} a > Qm \\ VMSR\ P0, Rp \\ VPST \\ VCVTT.F16.U16\ Qd, Qm, imm6 \\ p >> Rp \\ \\ VMSR\ P0, Rp \\ VPST \\ VCVTT.F16.U16\ Qd, Qm, imm6 \\ \\ VPST \\ VVST \\ VVST \\ VVST \\ VCVTT.F32.S32\ Qd, Qm, imm6 \\ \\ VVST \\ VCVTT.F32.S32\ Qd, Qm, imm6 \\ \\ VVST \\ VCVTT.F32.S32\ Qd, Qm, imm6 \\ \\ \end{array} \begin{array}{c} A > Qm \\ VMSR\ P0, Rp \\ VMSR\ P0, Rp \\ VPST \\ VCVTT.F32.S32\ Qd, Qm, imm6 \\ \\ VCVTT.F32.S32\ Qd, Qm, imm6 \\ \\ VCVTT.F32.S32\ Qd, Qm, imm6 \\ \\ \end{array}$					
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		-		Qd -> result	MVE
32	inactive, uint32x4_t a, const int imm6, mve_pred16_t p)				
Float16x8_t [_arm_]vcvtq_x_n[_f16_s16](int16x8_t a, const int imm6, mve_pred16_t p)		32	VCV11.F32.U32 Qd,Qm,imm6		
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	float16x8 t [ arm ]vcvtq x n[ f16 s16](int16x8 t a		VMSR P0 Rn	Od -> result	MVF.
p -> Rp		1 <= imm6 <=	VPST	Qu > resurt	1,1,1,1
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$			VC V 1 1.1·10.510 Qu,QIII,IIIIII0		
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	float16x8 t[ arm ]vcvta x n[ f16 u16](uint16x8 ta		VMSR P0.Rp	Od -> result	MVE
float32x4_t [_arm_]vcvtq_x_n[_f32_s32](int32x4_t a, const int imm6, mve_pred16_t p)  a -> Qm		1 <= imm6 <= 16	VPST	Qu' > Tesuit	
const int imm6, mve_pred16_t p)	floot22v4 t f _ own ]t		VMCD DO D.:	04 1	MVE
32 VCVTT.F32.S32 Qd,Qm,imm6				Qu -> resuit	IVI V E
	const int minio, inve_picuro_t p)				
p -> Rp		p -> Rp	2.52.522 (3,011,1111110		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [arm_]vcvtq_x_n[_f32_u32](uint32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
const int imm6, mve_pred16_t p)	1 <= imm6 <= 32 p -> Rp	VPST VCVTT.F32.U32 Qd,Qm,imm6		
int16x8_t [arm_]vcvtq_s16_f16(float16x8_t a)	a -> Qm	VCVT.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vcvtq_s32_f32(float32x4_t a)	a -> Qm	VCVT.S32.F32 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vcvtq_u16_f16(float16x8_t a)	a -> Qm	VCVT.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vcvtq_u32_f32(float32x4_t a)	a -> Qm	VCVT.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vcvtq_m[_s16_f16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VCVTT.S16.F16 Qd,Qm		
int32x4 t [ arm ]vcvtq m[ s32 f32](int32x4 t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
float32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Qu	
uint16x8_t [arm_]vcvtq_m[_u16_f16](uint16x8_t	p -> Rp inactive -> Qd	VCVTT.S32.F32 Qd,Qm VMSR P0,Rp	Od -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST	Qu -> resuit	IVIVE
	p -> Rp	VCVTT.U16.F16 Qd,Qm		
uint32x4_t [arm_]vcvtq_m[_u32_f32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
int16v0 t [ ome ]vorto v a16 f16/f1ast16v0 t a	p -> Rp	VCVTT.U32.F32 Qd,Qm	Od > monule	MVE
int16x8_t [arm_]vcvtq_x_s16_f16(float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
inve_preuro_t p)	p > Kp	VCVTT.S16.F16 Od,Om		
int32x4_t [arm_]vcvtq_x_s32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
1.150.15		VCVTT.S32.F32 Qd,Qm	0.1	No.
uint16x8_t [arm_]vcvtq_x_u16_f16(float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
nive_pieuro_t p)	p -> Kp	VCVTT.U16.F16 Qd,Qm		
uint32x4_t [arm_]vcvtq_x_u32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
	_	VCVTT.U32.F32 Qd,Qm		
int16x8_t [arm_]vcvtq_n_s16_f16(float16x8_t a, const int imm6)	a -> Qm 1 <= imm6 <=	VCVT.S16.F16 Qd,Qm,imm6	Qd -> result	MVE/NEON
int minio)	16			
int32x4_t [arm_]vcvtq_n_s32_f32(float32x4_t a, const	a -> Qm	VCVT.S32.F32 Qd,Qm,imm6	Qd -> result	MVE/NEON
int imm6)	1 <= imm6 <=			
1.150.15	32	VOLUME VILLE DE LA COLLA	0.1	MENTON
uint16x8_t [arm_]vcvtq_n_u16_f16(float16x8_t a, const int imm6)	a -> Qm 1 <= imm6 <=	VCVT.U16.F16 Qd,Qm,imm6	Qd -> result	MVE/NEON
in mino)	16			
uint32x4_t [arm_]vcvtq_n_u32_f32(float32x4_t a, const	a -> Qm	VCVT.U32.F32 Qd,Qm,imm6	Qd -> result	MVE/NEON
int imm6)	1 <= imm6 <=			
int16x8 t [ arm ]vcvtq m n[ s16 f16](int16x8 t	32 inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, const int imm6, mve_pred16_t p)	a -> Om	VMSK PO,KP VPST	Qu -> resuit	MVE
	1 <= imm6 <=	VCVTT.S16.F16 Qd,Qm,imm6		
	16			
:-,22-4 4 [ ]	p -> Rp	VAACD DO D.	0.1 >14	MVE
int32x4_t [_arm_]vcvtq_m_n[_s32_f32](int32x4_t inactive, float32x4_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, noat52x4_t a, const int imino, inve_pred10_t p)	1 <= imm6 <=	VCVTT.S32.F32 Qd,Qm,imm6		
	32			
	p -> Rp			
uint16x8_t [_arm_]vcvtq_m_n[_u16_f16](uint16x8_t inactive, float16x8_t a, const int imm6, mve_pred16_t p)	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
inactive, floatfox8_t a, const int immo, mve_pred1o_t p)	a -> Qm 1 <= imm6 <=	VCVTT.U16.F16 Qd,Qm,imm6		
	16	ve v 11.010.110 Qu,Qin,iiiiii0		
	p -> Rp			
uint32x4_t [arm_]vcvtq_m_n[_u32_f32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, const int imm6, mve_pred16_t p)	a -> Qm 1 <= imm6 <=	VPST VCVTT.U32.F32 Qd,Qm,imm6		
	32	VC V I 1.032.132 Qu,Qiii,iiiiiiio		
	p -> Rp			
int16x8_t [arm_]vcvtq_x_n_s16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
const int imm6, mve_pred16_t p)	1 <= imm6 <=	VPST		
	16 p -> Rp	VCVTT.S16.F16 Qd,Qm,imm6		
int32x4_t [arm_]vcvtq_x_n_s32_f32(float32x4_t a,	a -> Om	VMSR P0,Rp	Qd -> result	MVE
const int imm6, mve_pred16_t p)	1 <= imm6 <=	VPST	2 × 103uit	
	32	VCVTT.S32.F32 Qd,Qm,imm6		
140.45	p -> Rp	VACE DO D	01	Mare
uint16x8_t [arm_]vcvtq_x_n_u16_f16(float16x8_t a, const int imm6, mve_pred16_t p)	a -> Qm 1 <= imm6 <=	VMSR P0,Rp VPST	Qd -> result	MVE
const int mino, mve_picu1o_t p)	1 <= 1mm6 <= 16	VCVTT.U16.F16 Qd,Qm,imm6		
	p -> Rp	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2		

Float16x8_t _ arm_wrdpq_m[f16](float16x8_t inactive > Qd a > Qm	Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
12	uint32x4_t [arm_]vcvtq_x_n_u32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
Goat16.8.1   _am_  wmdq   72  Glora(23.4.1   a)   a > 0m	const int imm6, mve_pred16_t p)	32			
		1			_
		_ `			_
Doi:120.4_1_am_lyradq_m[_f32](float)224_1 inactive of a composite of the				Qa -> result	MVE
	noatrox8_t a, nive_pred16_t p)				
	float32x4 t [ arm ]vrndq m[ f32](float32x4 t inactive			Od -> result	MVE
Deat1688_1_arm_lyrndq_xl_f16(float168_1 a, a > 0 m				Qu'y resuit	
	2,	p -> Rp			
New North Content of the Content o	float16x8_t [arm_]vrndq_x[_f16](float16x8_t a,			Qd -> result	MVE
doad23.4.t   _amm_lymdq_xs_[32](fload224.t a, a   a > Qm	mve_pred16_t p)	p -> Rp	VPST		
Deat16.8_1_mm_lymdnng_[16](flotat168.8_1 a)   a > 0m					
NRINTZIF32 Qd.Qm	· ·	`		Qd -> result	MVE
	mve_pred16_t p)	p -> Rp			
	0 16 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	. 0		0.1 1:	MATE
		_			
a > 0m					
D > Rp				Qu -> resuit	NIVE
	mactive, noatroxo_t a, mvc_picuro_t p)	-			
Inactive,	float32x4 t [ arm ]vrndng m[ f32](float32x4 t			Od -> result	MVE
P. S. Rp   VRINTNTF32 Qd.Qm   Qd > result   MVE		`		Qu'y resuit	
Section   Sect	float16x8_t [arm_]vrndnq_x[_f16](float16x8_t a,			Qd -> result	MVE
	mve_pred16_t p)	p -> Rp			
Doct			VRINTNT.F16 Qd,Qm		
VRINTNTB32 Qd,Qm   Qd > result   MVE				Qd -> result	MVE
	mve_pred16_t p)	p -> Rp			
	Charles On Francis and Francis On the One of			0.1	) (T ) (T )
nactive, float16x8_t a, mve_pred16_t p   p					
Description				Qa -> result	MVE
	mactive, moatroxs_t a, mve_predro_t p)		1 12		
a - Q m	float32x4 t [ arm ]vrndmq m[ f32](float32x4 t			Od -> result	MVE
		a -> Qm	VPST	Qu > resuit	MY E
mve_pred16_t p	float16x8 t [ arm ]vrndmq x[ f16](float16x8 t a.			Od -> result	MVE
Float32x4_t [_arm_ vmdmq_x[32](float32x4_t a, mve_pred16_t p)					
Float16x8_t [_arm_]vrndpq[_f16](float16x8_t a)   a -> Qm   VRINTP.F15Q dQ,Qm   Qd -> result   MVE/NE	float32x4_t [arm_]vrndmq_x[_f32](float32x4_t a,	a -> Qm		Qd -> result	MVE
Roat16x8_t [_arm_]vmdpq[_f16](float16x8_t a)   a -> Qm   VRINTP.F16 Qd,Qm   Qd -> result   MVE	mve_pred16_t p)	-		`	
Float32x4_t[_arm_]vrndpq_[f32](float32x4_t a)   a -> Qm   VRINTP.F32 Qd,Qm   Qd -> result   MVE/NE					
Float16x8_t [_arm_]vrmdpq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)		_			
inactive, float16x8_t a, mve_pred16_t p)					MVE/NEON
P -> Rp				Qd -> result	MVE
Float32x4_t [_arm_]vrndpq_m[_f32](float32x4_t inactive -> Qd a -> Qm vRINTPT.F32 Qd,Qm p -> Rp vRINTPT.F32 Qd,Qm vRINTPTT.F32 Qd,Qm vRINTPTT.F32 Qd,Qm vRI	inactive, float16x8_t a, mve_pred16_t p)				
inactive, float32x4_t a, mve_pred16_t p)	floot22v4 t [	F :F		Od > magnife	MVE
P -> Rp				Qu -> resuit	IVI V L
Float16x8_t [_arm_]vrndpq_x[_f16](float16x8_t a, mve_pred16_t p)		`			
Description	float16x8_t [arm_]vrndpq_x[_f16](float16x8_t a,			Qd -> result	MVE
VRINTPT.F16 Qd,Qm	- t 1 I- t- 3	-	VPST	1	
Description			VRINTPT.F16 Qd,Qm		
VRINTPT.F32 Qd,Qm				Qd -> result	MVE
Float16x8_t [_arm_]vrndaq[_f16](float16x8_t a)   a -> Qm   VRINTA.F16 Qd,Qm   Qd -> result   MVE	mve_pred16_t p)	p -> Rp			
Float32x4_t[_arm_]vrndaq_[f32](float32x4_t a)   a -> Qm   VRINTA.F32 Qd,Qm   Qd -> result   MVE/NE	fl-416-0 4 [ ]. 1 [ fl63/fl dc 0 : )			01	MUT
float16x8_t [_arm_]vrndaq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)					
The float 16x8_t a, mve_pred16_t p					
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$			, 1	Qu -> resuit	IVI V L
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	noarrono_t a, mro_prouro_t p)				
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	float32x4_t [arm_]vrndaq_m[ f32](float32x4_t inactive.			Qd -> result	MVE
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$			VPST		
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$			VRINTAT.F32 Qd,Qm		
VRINTAT.F16 Qd,Qm	float16x8_t [arm_]vrndaq_x[_f16](float16x8_t a,			Qd -> result	MVE
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	mve_pred16_t p)	p -> Rp			
$\begin{array}{ccc} mve\_pred16\_t \ p) & & p \rightarrow Rp & VPST \\ & & VRINTAT.F32 \ Qd,Qm & & & \end{array}$		_			
VRINTAT.F32 Qd,Qm		-		Qd -> result	MVE
	mve_pred16_t p)	p -> Rp			
thoutless the new lyendral fleithoutless to be a compared to the first term of the f	float16v0 tf arm lymdyaf f161/float16v0 to	0 > 0==		Od > manula	MVE
					MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [_arm_]vrndxq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTXT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vrndxq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VKINTXT.F10 Qd,Qfif VMSR P0,Rp VPST VRINTXT.F32 Qd,Qm	Qd -> result	MVE
float16x8_t [_arm_]vrndxq_x[_f16](float16x8_t a, mve_pred16_t p)	p -> Rp a -> Qm p -> Rp	VKINTXT.F32 Qd,Qfii VMSR P0,Rp VPST VRINTXT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vrndxq_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VKINTXT.F10 Qd,Qfii VMSR P0,Rp VPST VRINTXT.F32 Qd,Qm	Qd -> result	MVE
int8x16_t [_arm_]vandq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vandq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vandq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vandq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vandq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vandq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vandq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [arm_]vandq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vandq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vandq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vandq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vandq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vandq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vandq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vandq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vandq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vandq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vandq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vandq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vandq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vandq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vandq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
float16x8_t [_arm_]vandq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VANDT Qd,Qn,Qm VMSR P0,Rp VPST	Qd -> result	MVE
float32x4_t [_arm_]vandq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VANDT Qd,Qn,Qm  VMSR P0,Rp  VPST	Qd -> result	MVE
int8x16_t [_arm_]vbicq[_s8](int8x16_t a, int8x16_t b)	p -> Rp a -> Qn b -> Qm	VANDT Qd,Qn,Qm VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vbicq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vbicq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vbicq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vbicq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vbicq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vbicq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]vbicq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vbicq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vbicq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vbicq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vbicq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vbicq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vbicq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vbicq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vbicq_m[f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vbicq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Od,On,Om	Qd -> result	MVE
int16x8_t [arm_]vbicq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vbicq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vbicq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vbicq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vbicq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VBICT Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [arm_]vbicq_x[_f16](float16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
float32x4_t [arm_]vbicq_x[_f32](float32x4_t a,	p -> Rp a -> On	VBICT Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Qu > resuit	III V E
	p -> Rp	VBICT Qd,Qn,Qm		
int16x8_t [arm_]vbicq[_n_s16](int16x8_t a, const	a -> Qda	VBIC.I16 Qda,#imm	Qda -> result	MVE
int16_t imm)	imm in AdvSIMDExpa			
	ndImm			
int32x4_t [arm_]vbicq[_n_s32](int32x4_t a, const	a -> Qda	VBIC.I32 Qda,#imm	Qda -> result	MVE
int32_t imm)	imm in			
	AdvSIMDExpa ndImm			
uint16x8_t [arm_]vbicq[_n_u16](uint16x8_t a, const	a -> Qda	VBIC.I16 Qda,#imm	Qda -> result	MVE
uint16_t imm)	imm in		<b>Q</b>	1
	AdvSIMDExpa			
uint32x4_t [arm_]vbicq[_n_u32](uint32x4_t a, const	ndImm	VBIC.I32 Qda,#imm	Odo > monult	MVE
uint32x4_t [arm_]voicq[_n_u32](uint32x4_t a, const uint32_t imm)	a -> Qda imm in	VBIC.132 Qda,#1mm	Qda -> result	MVE
umis2_t mini)	AdvSIMDExpa			
	ndImm			
int16x8_t [arm_]vbicq_m_n[_s16](int16x8_t a, const	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int16_t imm, mve_pred16_t p)	imm in AdvSIMDExpa	VPST VBICT.I16 Qda,#imm		
	ndImm	VBIC1:110 Qua,#IIIIII		
	p -> Rp			
int32x4_t [arm_]vbicq_m_n[_s32](int32x4_t a, const	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32_t imm, mve_pred16_t p)	imm in	VPST		
	AdvSIMDExpa ndImm	VBICT.I32 Qda,#imm		
	p -> Rp			
uint16x8_t [arm_]vbicq_m_n[_u16](uint16x8_t a, const	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint16_t imm, mve_pred16_t p)	imm in	VPST		
	AdvSIMDExpa ndImm	VBICT.I16 Qda,#imm		
	p -> Rp			
uint32x4_t [arm_]vbicq_m_n[_u32](uint32x4_t a, const	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint32_t imm, mve_pred16_t p)	imm in	VPST		
	AdvSIMDExpa ndImm	VBICT.I32 Qda,#imm		
	p -> Rp			
int8x16_t [arm_]vbrsrq[_n_s8](int8x16_t a, int32_t b)	a -> Qn	VBRSR.8 Qd,Qn,Rm	Qd -> result	MVE
	b -> Rm			
int16x8_t [arm_]vbrsrq[_n_s16](int16x8_t a, int32_t b)	a -> Qn b -> Rm	VBRSR.16 Qd,Qn,Rm	Qd -> result	MVE
int32x4 t [ arm ]vbrsrq[ n s32](int32x4 t a, int32 t b)	a -> Qn	VBRSR.32 Qd,Qn,Rm	Od -> result	MVE
	b -> Rm	2,2,	`	
uint8x16_t [arm_]vbrsrq[_n_u8](uint8x16_t a, int32_t	a -> Qn	VBRSR.8 Qd,Qn,Rm	Qd -> result	MVE
b) uint16x8_t [arm_]vbrsrq[_n_u16](uint16x8_t a, int32_t	b -> Rm a -> Qn	VBRSR.16 Qd,Qn,Rm	Qd -> result	MVE
b)	b -> Rm	, DRDR. 10 Qu,Qii,Riii	Qu -> icsuit	171 7 15
uint32x4_t [arm_]vbrsrq[_n_u32](uint32x4_t a, int32_t	a -> Qn	VBRSR.32 Qd,Qn,Rm	Qd -> result	MVE
b)	b -> Rm	Ambab 100:0		) AUT
float16x8_t [arm_]vbrsrq[_n_f16](float16x8_t a, int32_t b)	a -> Qn b -> Rm	VBRSR.16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [arm_]vbrsrq[_n_f32](float32x4_t a, int32_t	a -> Qn	VBRSR.32 Qd,Qn,Rm	Qd -> result	MVE
b)	b -> Rm	2,2,	Ç	
int8x16_t [arm_]vbrsrq_m[_n_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VBRSRT.8 Qd,Qn,Rm		
	p -> Rm	DISKT .0 QU,QII,KIII		
int16x8_t [arm_]vbrsrq_m[_n_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm p -> Rp	VBRSRT.16 Qd,Qn,Rm		
int32x4_t [_arm_]vbrsrq_m[_n_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t taimjvorsiq_mn_s32j(ms2x4_t macuve, int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn	VPST	Qu > Iosuit	1
- ••	b -> Rm	VBRSRT.32 Qd,Qn,Rm		
windows of the same less of the same les	p -> Rp	VMCD DO D	01: 1:	MVE
uint8x16_t [arm_]vbrsrq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
amorto_t a, moz_t o, mvc_picuto_t p)	b -> Rm	VBRSRT.8 Qd,Qn,Rm		
	p -> Rp			

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vbrsrq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm	VMSR P0,Rp VPST VBRSRT.16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [arm_]vbrsrq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, int32_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [arm_]vbrsrq_m[_n_f16](float16x8_t inactive, float16x8_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm	VMSR P0,Rp VPST VBRSRT.16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [arm_]vbrsrq_m[_n_f32](float32x4_t inactive, float32x4_t a, int32_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vbrsrq_x[_n_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vbrsrq_x[_n_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [arm_]vbrsrq_x[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vbrsrq_x[_n_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vbrsrq_x[_n_u16](uint16x8_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vbrsrq_x[_n_u32](uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [arm_]vbrsrq_x[_n_f16](float16x8_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [arm_]vbrsrq_x[_n_f32](float32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [_arm_]veorq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]veorq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [_arm_]veorq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]veorq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]veorq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]veorq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [_arm_]veorq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]veorq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]veorq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]veorq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]veorq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]veorq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]veorq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]veorq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [arm_]veorq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]veorq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]veorq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]veorq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm p -> Rp	VEORT Qd,Qn,Qm  VMSR P0,Rp  VPST  VEORT Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]veorq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]veorq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]veorq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]veorq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]veorq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE MVE
float32x4_t [_arm_]veorq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp a -> Qm	VMSR P0,Rp VPST VEORT Qd,Qn,Qm VMOVLB.S8 Qd,Qm	Qd -> result  Qd -> result	MVE
int16x8_t [arm_]vmovlbq[_s8](int8x16_t a) int32x4_t [arm_]vmovlbq[_s16](int16x8_t a)	a -> Qm	VMOVLB.S8 Qd,Qm	Od -> result	MVE
uint16x8_t [arm_]vmovlbq[_u8](uint8x16_t a)	a -> Om	VMOVLB.U8 Qd,Qm	Od -> result	MVE
uint32x4_t [_arm_]vmovlbq[_u16](uint16x8_t a)	a -> Qm	VMOVLB.U16 Qd,Qm	Qd -> result	MVE
int16x8_t [_arm_]vmovlbq_m[_s8](int16x8_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.S8 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vmovlbq_m[_s16](int32x4_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmovlbq_m[_u8](uint16x8_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.U8 Qd,Qm	Qd -> result	MVE
uint32x4_t [arm_]vmovlbq_m[_u16](uint32x4_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.U16 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vmovlbq_x[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.S8 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vmovlbq_x[_s16](int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vmovlbq_x[_u8](uint8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.U8 Qd,Qm	Qd -> result	MVE
uint32x4_t [arm_]vmovlbq_x[_u16](uint16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.U16 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vmovltq[_s8](int8x16_t a)	a -> Qm	VMOVLT.S8 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vmovltq[_s16](int16x8_t a)	a -> Qm	VMOVLT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmovltq[_u8](uint8x16_t a)	a -> Qm	VMOVLT.U8 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmovltq[_u16](uint16x8_t a)	a -> Qm	VMOVLT.U16 Qd,Qm	Qd -> result	MVE
int16x8_t [_arm_]vmovltq_m[_s8](int16x8_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.S8 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vmovltq_m[_s16](int32x4_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.S16 Qd,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [_arm_]vmovltq_m[_u8](uint16x8_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.U8 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmovltq_m[_u16](uint32x4_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.U16 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vmovltq_x[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.S8 Od,Om	Qd -> result	MVE
int32x4_t [arm_]vmovltq_x[_s16](int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmovltq_x[_u8](uint8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.U8 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmovltq_x[_u16](uint16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.U16 Qd,Qm	Qd -> result	MVE
int8x16_t [_arm_]vmovnbq[_s16](int8x16_t a, int16x8_t b)	a -> Qd b -> Qm	VMOVNB.I16 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vmovnbq[_s32](int16x8_t a, int32x4_t b)	a -> Qd b -> Qm	VMOVNB.I32 Qd,Qm	Qd -> result	MVE
uint8x16_t [arm_]vmovnbq[_u16](uint8x16_t a, uint16x8_t b)	a -> Qd b -> Qm	VMOVNB.I16 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vmovnbq[_u32](uint16x8_t a, uint32x4_t b)	a -> Qd b -> Qm	VMOVNB.I32 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vmovnbq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNBT.I16 Od,Om	Qd -> result	MVE
int16x8_t [arm_]vmovnbq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNBT.I32 Qd,Qm	Qd -> result	MVE
uint8x16_t [arm_]vmovnbq_m[_u16](uint8x16_t a, uint16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST VMOVNBT.I16 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vmovnbq_m[_u32](uint16x8_t a, uint32x4_t b, mve_pred16_t p)	p -> Rp a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNBT.I32 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vmovntq[_s16](int8x16_t a, int16x8_t b)	a -> Qd b -> Qm	VMOVNT.I16 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vmovntq[_s32](int16x8_t a, int32x4_t b)	a -> Qd b -> Qm	VMOVNT.I32 Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vmovntq[_u16](uint8x16_t a, uint16x8_t b)	a -> Qd b -> Qm	VMOVNT.I16 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vmovntq[_u32](uint16x8_t a, uint32x4_t b)	a -> Qd b -> Qm	VMOVNT.I32 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vmovntq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNTT.I16 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vmovntq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNTT.I32 Qd,Qm	Qd -> result	MVE
uint8x16_t [arm_]vmovntq_m[_u16](uint8x16_t a, uint16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNTT.I16 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vmovntq_m[_u32](uint16x8_t a, uint32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNTT.I32 Od,Om	Qd -> result	MVE
int8x16_t [_arm_]vmvnq[_s8](int8x16_t a) int16x8_t [_arm_]vmvnq[_s16](int16x8_t a)	a -> Qm	VMVN Qd,Qm VMVN Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
int32x4_t [_arm_]vmvnq[_s32](int32x4_t a)	a -> Qm a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vmvnq[_u8](uint8x16_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vmvnq[_u16](uint16x8_t a) uint32x4_t [arm_]vmvnq[_u32](uint32x4_t a)	a -> Qm a -> Qm	VMVN Qd,Qm VMVN Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
int8x16_t [_arm_]vmvnq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int16x8_t [_arm_]vmvnq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t [arm_]vmvnq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm p -> Rp	VMVNT Qd,Qm  VMSR P0,Rp  VPST  VMVNT Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vmvnq_m[_u8](uint8x16_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vmvnq_m[_u16](uint16x8_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Od,Om	Qd -> result	MVE
uint32x4_t [_arm_]vmvnq_m[_u32](uint32x4_t inactive, uint32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vmvnq_x[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vmvnq_x[_s16](int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int32x4_t [arm_]vmvnq_x[_s32](int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vmvnq_x[_u8](uint8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmvnq_x[_u16](uint16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmvnq_x[_u32](uint32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int16x8_t [_arm_]vmvnq_n_s16(const int16_t imm)	imm in AdvSIMDExpa ndImm	VMVN.I16 Qd,#imm	Qd -> result	MVE
int32x4_t [_arm_]vmvnq_n_s32(const int32_t imm)	imm in AdvSIMDExpa ndImm	VMVN.I32 Qd,#imm	Qd -> result	MVE
uint16x8_t [arm_]vmvnq_n_u16(const uint16_t imm)	imm in AdvSIMDExpa ndImm	VMVN.I16 Qd,#imm	Qd -> result	MVE
uint32x4_t [arm_]vmvnq_n_u32(const uint32_t imm)	imm in AdvSIMDExpa ndImm	VMVN.I32 Qd,#imm	Qd -> result	MVE
int16x8_t [arm_]vmvnq_m[_n_s16](int16x8_t inactive, const int16_t imm, mve_pred16_t p)	inactive -> Qd imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I16 Qd,#imm	Qd -> result	MVE
int32x4_t [arm_]vmvnq_m[_n_s32](int32x4_t inactive, const int32_t imm, mve_pred16_t p)	inactive -> Qd imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I32 Qd,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vmvnq_m[_n_u16](uint16x8_t inactive, const uint16_t imm, mve_pred16_t p)	inactive -> Qd imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I16 Qd,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vmvnq_m[_n_u32](uint32x4_t inactive, const uint32_t imm, mve_pred16_t p)	inactive -> Qd imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I32 Qd,#imm	Qd -> result	MVE
int16x8_t [arm_]vmvnq_x_n_s16(const int16_t imm, mve_pred16_t p)	imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I16 Qd,#imm	Qd -> result	MVE
int32x4_t [arm_]vmvnq_x_n_s32(const int32_t imm, mve_pred16_t p)	imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I32 Qd,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vmvnq_x_n_u16(const uint16_t imm, mve_pred16_t p)	imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I16 Qd,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vmvnq_x_n_u32(const uint32_t imm, mve_pred16_t p)	imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I32 Qd,#imm	Qd -> result	MVE
mve_pred16_t [arm_]vpnot(mve_pred16_t a)	a -> Rp	VMSR P0,Rp VPNOT VMRS Rt,P0	Rt -> result	MVE

	Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
Doc	int8x16_t [arm_]vpselq[_s8](int8x16_t a, int8x16_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
int		`	VPSEL Qd,Qn,Qm		
move_perd(d_t_p)	1.150.150.150.150.1150.1		VII (OD DO D	0.1	) am
Desire				Qd -> result	MVE
ind234_1_mm_lypselg_s23[ini328_d_1 a, ini328_d_1 b)	mive_pred10_t p)		VFSEL Qu,Qii,Qiii		
move_predid_sp   move	int32x4 t [ arm ]vpselq[ s32](int32x4 t a, int32x4 t b,		VMSR P0,Rp	Od -> result	MVE
inition   months		b -> Qm		`	
move_predicl_tp					
Description		-		Qd -> result	MVE
min816_1arm_lypselq_Lo8[(uim8x16_1 a, uim8x16_1 b)	mve_pred16_t p)		VPSEL Qd,Qn,Qm		
b. me_predic_t_p  b	uint8x16 t [ arm ]vpselq[ u8](uint8x16 t a, uint8x16 t		VMSR P0.Rp	Od -> result	MVE
Mint 68.8.1  arm_ ypseld_1.u5  dunt 168.1 a,   a > Qn					
b > Qm					
Description				Qd -> result	MVE
dint324.4   L_amm_lypsela(_u32 dint323.4_1 a, uint634.2   L_amm_lypsela(_u04 dint64.2_1 a, u)	uint16x8_t b, mve_pred16_t p)	-	VPSEL Qd,Qn,Qm		
uint324_1 b. mve_pred16_1p) b > 0m	nint32x4 t [ arm ]vpsela[ n32](nint32x4 t a		VMSR P0 Rn	Od -> result	MVE
				Qu' > Tesan	1,1,2
Docal Cost, 1_ arm_lypesplag_116](diotal Cost, 1_					
D - No				Qd -> result	MVE
	uint64x2_t b, mve_pred16_t p)		VPSEL Qd,Qn,Qm		
Doad 16x8_t   L. arm_lyornq_[15](lota15x4_t   a, b > Qm   PSEL_Q4Q, Qm   Qd > result   MVE	float16x8 t [ arm ]vpselg[ f16](float16x8 t a		VMSR P0 Rn	Od -> result	MVE
D > Rp				Qu -> result	MVL
fload32x4_t_b, mve_predl6_tp   b > Qm	The state of the s		20,00		
P > Rp		a -> Qn		Qd -> result	MVE
	float32x4_t b, mve_pred16_t p)		VPSEL Qd,Qn,Qm		
	float16v9 t f arm lyarnal f16l/float16v9 to		VORN Od On Om	Od > rocult	MVE
			VOKN Qu,Qii,Qiii	Qu -> resuit	MIVE
Int8x16_t [_arm_ vomq[_s8](int8x16_t a, int16x8_t b)   a > Qn   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   a > Qn   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   d > Qm   vORN Qd,Qn,Qm   Qd > result   MVE/NEON   d > Qm   vORN Qd,Qn,Qm   vORN Qd,Q		_	VORN Qd,Qn,Qm	Qd -> result	MVE
b > Qm		b -> Qm		ì	
Int16x8_t [_arm_ vomq[_s16](int16x8_t a, int16x8_t b)   a > Qn   b > Qm   VORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   VORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   VORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   VORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   VORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   VORN Qd,Qn,Qm   Qd > result   MVE/NEON   MVE/NEON   Day   D	int8x16_t [arm_]vornq[_s8](int8x16_t a, int8x16_t b)		VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
b > Qm	int16v9 t [		VORN Od On Om	Od > magualt	MVENEON
int32x4_t [_arm_ vornq[_s32 (int32x4_t a, int32x4_t b)   a -> Qn   b -> Qm   VORN Qd,Qn,Qm   Qd -> result   MVE/NEON	intlox8_t [arm_jvornq[_sloj(intlox8_t a, intlox8_t b)		VORN Qd,Qn,Qm	Qu -> result	MVE/NEON
Uint8x16_t [_arm_]vornq[u8](uint8x16_t a, uint8x16_t b   a > Qn   b > Qm   VORN Qd,Qn,Qm   Qd > result   MVE/NEON   b > Qm   VORN Qd,Qn,Qm   Qd > result   MVE/NEON   Ds > Qm   VORN Qd,Qn,Qm   Qd > result   MVE/NEON   Ds > Qm   VORN Qd,Qn,Qm   Qd > result   MVE/NEON   Ds > Qm   VORN Qd,Qn,Qm   Qd > result   MVE/NEON   Result   Res	int32x4 t[ arm ]vorng[ s32](int32x4 t a, int32x4 t b)		VORN Od.On.Om	Od -> result	MVE/NEON
b > Qm		b -> Qm	2,72,72	`	
uint16x8_t [_arm_ vornq_u16](uint16x8_t a, uint16x8_t b)		-	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
b > Qm  uint32x4_t [_arm_]vornq[_u32](uint32x4_t a, uint32x4_t b)  b > Qm  float16x8_t [_arm_]vornq_m[_f16](float16x8_t i nactive, float32x4_t a, float16x8_t b, mve_pred16_t p)  float32x4_t [_arm_]vornq_m[_f32](float32x4_t i nactive, float32x4_t b, mve_pred16_t p)  int8x16_t a, float32x4_t b, mve_pred16_t p)  int8x16_t a, int8x16_t b, mve_pred16_t p)  int16x8_t [_arm_]vornq_m[_s16](int16x8_t i nactive, int16x8_t a, int16x8_t b, mve_pred16_t p)  int32x4_t [_arm_]vornq_m[_s2](int32x4_t i nactive, inactive > Qd a > Qn b > Qd > result  MVE  MVE  MVE  MVE  MVE  MVE  MVE  MV			VORN OLO O	0.1	MVEATEON
uint32x4_t [_arm_]vornq[u32](uint32x4_t a, uint32x4_t b)         a > On b >		-	VORN Qd,Qn,Qm	Qu -> result	MVE/NEON
b   c   c   c   c   c   c   c   c   c		_	VORN Od.On.Om	Od -> result	MVE/NEON
float16x8_t a, float16x8_t b, mve_pred16_t p)					
b -> Qm				Qd -> result	MVE
P -> Rp	float16x8_t a, float16x8_t b, mve_pred16_t p)				
float32x4_t [_arm_]vornq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)			VOKN1 Qd,Qn,Qm		
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	float32x4 t[ arm ]vorng m[ f32](float32x4 t inactive		VMSR P0.Rn	Od -> result	MVE
b -> Qm				Ç= , 100an	
int8x16_t [_arm_]vornq_m[_s8](int8x16_t inactive, inactive -> Qd   a -> Qn   VPST   VORNT Qd,Qn,Qm   P -> Rp   VPST   VORNT Qd,Qn,Qm   VPST   VORNT Qd,Qn,Qm   P -> Rp   VPST   VORNT Qd,Qn,Qm   P -> Rp   VPST   VORNT Qd,Qn,Qm			VORNT Qd,Qn,Qm		
$ \begin{array}{c} int8x16\_t \ a, int8x16\_t \ b, mve\_pred16\_t \ p) \\ b \ - Q m \\ b \ - Q m \\ p \ - R p \\ \hline \\ int16x8\_t \ [\_arm\_]vornq\_m[\_s16](int16x8\_t \ inactive, \\ int16x8\_t \ a, int16x8\_t \ b, mve\_pred16\_t \ p) \\ \hline \\ int32x4\_t \ [\_arm\_]vornq\_m[\_s32](int32x4\_t \ inactive, \\ int32x4\_t \ a, int32x4\_t \ b, mve\_pred16\_t \ p) \\ \hline \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ $			VMCD DO D	01 : 1	MVE
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$				Qa -> result	MVE
p -> Rp	mioxio_t a, mioxio_t o, mvc_picuro_t p)				
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$		p -> Rp	1		
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$				Qd -> result	MVE
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	int16x8_t a, int16x8_t b, mve_pred16_t p)				
int32x4_t [_arm_]vornq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)  uint8x16_t [_arm_]vornq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)  uint16x8_t [_arm_]vornq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)  uint16x8_t a, uint16x8_t b, mve_pred16_t b, uint16x8_t b, uint16x8_t b, uint16x8_t b, uint16x8_t b, uint16x8_t b, uint16x8_t b, uint16x8_		-	VOKIVI QU,QII,QM		
$ \begin{array}{c} int32x4\_t\ a, int32x4\_t\ b, mve\_pred16\_t\ p) \\ \\ a \rightarrow Qn \\ b \rightarrow Qm \\ \\ vint8x16\_t\ [\_arm\_]vornq\_m[\_u8](uint8x16\_t\ inactive, \\ uint8x16\_t\ a, uint8x16\_t\ b, mve\_pred16\_t\ p) \\ \\ \\ uint16x8\_t\ [\_arm\_]vornq\_m[\_u16](uint16x8\_t\ inactive, \\ uint16x8\_t\ a, uint16x8\_t\ b, mve\_pred16\_t\ p) \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ $	int32x4_t [arm_]vornq_m[_s32](int32x4_t inactive.		VMSR P0,Rp	Qd -> result	MVE
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$			VPST		
$ \begin{array}{c} uint8x16\_t \ [\_arm\_]vornq\_m[\_u8](uint8x16\_t \ inactive, \\ uint8x16\_t \ a, \ uint8x16\_t \ b, \ mve\_pred16\_t \ p) \end{array} \qquad \begin{array}{c} inactive -> Qd \\ a -> Qn \\ b -> Qm \\ vORNT \ Qd,Qn,Qm \\ p -> Rp \\ uint16x8\_t \ [\_arm\_]vornq\_m[\_u16](uint16x8\_t \ inactive, \\ uint16x8\_t \ a, \ uint16x8\_t \ b, \ mve\_pred16\_t \ p) \end{array} \qquad \begin{array}{c} inactive -> Qd \\ a -> Qn \\ vORNT \ Qd,Qn,Qm \\ a -> Qn \\ vORNT \ Qd,Qn,Qm \\ vORNT \ Qd,Qn,Qm \\ b -> Qm \end{array} \qquad \begin{array}{c} Qd -> result \\ MVE \\ M$			VORNT Qd,Qn,Qm		
$\begin{array}{c} uint8x16\_t \ a, uint8x16\_t \ b, mve\_pred16\_t \ p) \\ b \rightarrow Qm \\ p \rightarrow Rp \\ \\ uint16x8\_t \ [\_arm\_]vornq\_m[\_u16](uint16x8\_t \ inactive, \\ uint16x8\_t \ a, uint16x8\_t \ b, mve\_pred16\_t \ p) \\ \end{array} \begin{array}{c} a \rightarrow Qn \\ p \rightarrow Rp \\ \\ uint16x8\_t \ a, uint16x8\_t \ b, mve\_pred16\_t \ p) \\ \end{array} \begin{array}{c} A \rightarrow Qn \\ p \rightarrow Rp \\ \\ uint16x8\_t \ a, uint16x8\_t \ b, mve\_pred16\_t \ p) \\ \end{array} \begin{array}{c} A \rightarrow Qn \\ a \rightarrow Qn \\ b \rightarrow Qm \\ \end{array} \begin{array}{c} VMSR \ PO, Rp \\ VPST \\ VORNT \ Qd, Qn, Qm \\ \end{array} \begin{array}{c} Qd \rightarrow result \\ MVE \\ \end{array}$	wint0v16 t f ama lyama and -016-int0-16 time i		VMCD DO D	041	MVE
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$				Qa -> result	WIVE
p -> Rp         VMSR P0,Rp         Qd -> result         MVE           uint16x8_t a, uint16x8_t b, mve_pred16_t p)         a -> Qn VPST b -> Qm         VORNT Qd,Qn,Qm         VORNT Qd,Qn,Qm         VORNT Qd,Qn,Qm					
uint16x8_t a, uint16x8_t b, mve_pred16_t p)       a -> Qn       VPST         b -> Qm       VORNT Qd,Qn,Qm		p -> Rp			
b -> Qm VORNT Qd,Qn,Qm				Qd -> result	MVE
	uint16x8_t a, uint16x8_t b, mve_pred16_t p)	-			
		b -> Qm p -> Rp	VOKIVI Qa,Qn,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [arm_]vornq_m[_u32](uint32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VPST VORNT Qd,Qn,Qm		
float16x8_t [arm_]vornq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
float32x4_t [_arm_]vornq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VORNT Qd,Qn,Qm VMSR P0,Rp VPST	Qd -> result	MVE
int8x16_t [_arm_]vornq_x[_s8](int8x16_t a, int8x16_t b,	p -> Rp a -> Qn	VORNT Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Qm p -> Rp	VPST VORNT Qd,Qn,Qm		
int16x8_t [arm_]vornq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORNT Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vornq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORNT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vornq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST VORNT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vornq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t [arm_]vornq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VORNT Qd,Qn,Qm VMSR P0,Rp VPST	Qd -> result	MVE
float16x8_t [arm_]vorrq[_f16](float16x8_t a, float16x8_t b)	p -> Rp a -> Qn b -> Qm	VORNT Qd,Qn,Qm VORR Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vorrq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vorrq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vorrq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vorrq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vorrq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vorrq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vorrq[_u32](uint32x4_t a, uint32x4_t b)  float16x8 t [ arm ]vorrq m[ f16](float16x8 t inactive,	a -> Qn b -> Qm inactive -> Qd	VORR Qd,Qn,Qm  VMSR P0,Rp	Qd -> result  Od -> result	MVE/NEON MVE
float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSK F0,KP VPST VORRT Qd,Qn,Qm	Qu -> resuit	WIVE
float32x4_t [_arm_]vorrq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vorrq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t [arm_]vorrq_m[_s16](int16x8_t inactive,	b -> Qm p -> Rp inactive -> Qd	VORRT Qd,Qn,Qm  VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VPST VORRT Qd,Qn,Qm		
int32x4_t [arm_]vorrq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vorrq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vorrq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vorrq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [_arm_]vorrq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vorrq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VORKT Qd,Qii,Qiii  VMSR P0,Rp  VPST  VORRT Qd,Qi,Qii	Qd -> result	MVE
int8x16_t [arm_]vorrq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vorrq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vorrq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [arm_]vorrq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vorrq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VORKT Qd,Qii,Qiii  VMSR P0,Rp  VPST  VORRT Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [arm_]vorrq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vorrq[_n_s16](int16x8_t a, const int16_t imm)	a -> Qda imm in AdvSIMDExpa ndImm	VORR.I16 Qda,#imm	Qda -> result	MVE
int32x4_t [_arm_]vorrq[_n_s32](int32x4_t a, const int32_t imm)	a -> Qda imm in AdvSIMDExpa ndImm	VORR.I32 Qda,#imm	Qda -> result	MVE
uint16x8_t [_arm_]vorrq[_n_u16](uint16x8_t a, const uint16_t imm)	a -> Qda imm in AdvSIMDExpa ndImm	VORR.I16 Qda,#imm	Qda -> result	MVE
uint32x4_t [_arm_]vorrq[_n_u32](uint32x4_t a, const uint32_t imm)	a -> Qda imm in AdvSIMDExpa ndImm	VORR.I32 Qda,#imm	Qda -> result	MVE
int16x8_t [arm_]vorrq_m_n[_s16](int16x8_t a, const int16_t imm, mve_pred16_t p)	a -> Qda imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VORRT.I16 Qda,#imm	Qda -> result	MVE
int32x4_t [arm_]vorrq_m_n[_s32](int32x4_t a, const int32_t imm, mve_pred16_t p)	a -> Qda imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VORRT.I32 Qda,#imm	Qda -> result	MVE
uint16x8_t [arm_]vorrq_m_n[_u16](uint16x8_t a, const uint16_t imm, mve_pred16_t p)	a -> Qda imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VORRT.I16 Qda,#imm	Qda -> result	MVE
uint32x4_t [_arm_]vorrq_m_n[u32](uint32x4_t a, const uint32_t imm, mve_pred16_t p)	a -> Qda imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VORRT.I32 Qda,#imm	Qda -> result	MVE
int8x16_t [_arm_]vqmovnbq[_s16](int8x16_t a, int16x8_t b)	a -> Qd b -> Qm	VQMOVNB.S16 Qd,Qm	Qd -> result	MVE
int16x8_t [_arm_]vqmovnbq[_s32](int16x8_t a, int32x4_t b)	a -> Qd b -> Qm	VQMOVNB.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vqmovnbq[_u16](uint8x16_t a, uint16x8_t b) uint16x8_t [_arm_]vqmovnbq[_u32](uint16x8_t a,	a -> Qd b -> Qm a -> Od	VQMOVNB.U16 Qd,Qm VOMOVNB.U32 Qd,Qm	Qd -> result  Od -> result	MVE MVE
int8x16_t [arm_]vqmovnbq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t [_arm_]vqmovnbq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p)	p -> Qm p -> Rp a -> Qd b -> Qm p -> Rp	VPS1 VQMOVNBT.S16 Qd,Qm VMSR P0,Rp VPST VQMOVNBT.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vqmovnbq_m[_u16](uint8x16_t a, uint16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVNBT.U16 Qd,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vqmovnbq_m[_u32](uint16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
int8x16_t [arm_]vqmovntq[_s16](int8x16_t a, int16x8_t	p -> Rp	VQMOVNBT.U32 Qd,Qm VQMOVNT.S16 Qd,Qm	Od -> result	MVE
b)	a -> Qd b -> Qm	VQMOVN1.S16 Qd,Qiii	Qu -> resuit	MIVE
int16x8_t [arm_]vqmovntq[_s32](int16x8_t a, int32x4_t	a -> Qd	VQMOVNT.S32 Qd,Qm	Qd -> result	MVE
b)	b->Qm			
uint8x16_t [arm_]vqmovntq[_u16](uint8x16_t a,	a -> Qd	VQMOVNT.U16 Qd,Qm	Qd -> result	MVE
uint16x8_t b) uint16x8 t [ arm ]vqmovntq[ u32](uint16x8 t a,	b -> Qm a -> Qd	VQMOVNT.U32 Qd,Qm	Od -> result	MVE
uint32x4_t b)	b -> Qm		Ç	
int8x16_t [arm_]vqmovntq_m[_s16](int8x16_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VQMOVNTT.S16 Qd,Qm		
int16x8_t [arm_]vqmovntq_m[_s32](int16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
uint8x16_t [arm_]vqmovntq_m[_u16](uint8x16_t a,	p -> Rp	VQMOVNTT.S32 Qd,Qm VMSR P0,Rp	Od -> result	MVE
uint16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSK PO,KP VPST	Qu -> resuit	IVI V E
umitono_t e, mve_preuto_t p)	p -> Rp	VQMOVNTT.U16 Qd,Qm		
uint16x8_t [arm_]vqmovntq_m[_u32](uint16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VQMOVNTT.U32 Qd,Qm		
uint8x16 t [ arm ]vqmovunbq[ s16](uint8x16 t a,	a -> Qd	VQMOVIVI1.032 Qd,Qm	Od -> result	MVE
int16x8_t b)	b -> Qm	. (2.55 : 55 .2.55 (2.5)	Q0	
uint16x8_t [arm_]vqmovunbq[_s32](uint16x8_t a,	a -> Qd	VQMOVUNB.S32 Qd,Qm	Qd -> result	MVE
int32x4_t b) uint8x16_t [arm_]vqmovunbq_m[_s16](uint8x16_t a,	b -> Qm	VMCD DO Do	Od -> result	MVE
int16x8 t b, mve pred16 t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qu -> resuit	IVI V E
	p -> Rp	VQMOVUNBT.S16 Qd,Qm		
uint16x8_t [arm_]vqmovunbq_m[_s32](uint16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VQMOVUNBT.S32 Qd,Qm		
uint8x16_t [arm_]vqmovuntq[_s16](uint8x16_t a,	a -> Qd	VQMOVUNT.S16 Qd,Qm	Qd -> result	MVE
int16x8_t b)	b -> Qm	. (2.50 . 51.5.55 (2.)(2.5	Q0	
uint16x8_t [_arm_]vqmovuntq[_s32](uint16x8_t a,	a -> Qd	VQMOVUNT.S32 Qd,Qm	Qd -> result	MVE
int32x4_t b) uint8x16_t [arm_]vqmovuntq_m[_s16](uint8x16_t a,	b -> Qm a -> Od	VMSR P0,Rp	Od -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qu b -> Qm	VPST	Qu -> resuit	WIVE
	p -> Rp	VQMOVUNTT.S16 Qd,Qm		
uint16x8_t [_arm_]vqmovuntq_m[_s32](uint16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VQMOVUNTT.S32 Qd,Qm		
int8x16_t [arm_]vqrshlq[_n_s8](int8x16_t a, int32_t b)	a -> Qda	VQRSHL.S8 Qda,Rm	Qda -> result	MVE
	b -> Rm			
int16x8_t [arm_]vqrshlq[_n_s16](int16x8_t a, int32_t b)	a -> Qda	VQRSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vqrshlq[_n_s32](int32x4_t a, int32_t b)	b -> Rm a -> Qda	VQRSHL.S32 Qda,Rm	Oda -> result	MVE
mt52x4_t [atm_]vqtsmq[_n_352](mt52x4_t a, mt52_t b)	b -> Rm	VQRSTE.552 Qua,Riii	Qua -> resuit	WIVE
uint8x16_t [arm_]vqrshlq[_n_u8](uint8x16_t a, int32_t	a -> Qda	VQRSHL.U8 Qda,Rm	Qda -> result	MVE
b) uint16x8_t [arm_]vqrshlq[_n_u16](uint16x8_t a, int32_t	b -> Rm	VORSHL.U16 Oda,Rm	0.1	MVE
b)	a -> Qda b -> Rm	VQRSHL.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [arm_]vqrshlq[_n_u32](uint32x4_t a, int32_t	a -> Qda	VQRSHL.U32 Qda,Rm	Qda -> result	MVE
b)	b -> Rm			
int8x16_t [_arm_]vqrshlq_m_n[_s8](int8x16_t a, int32_t	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
b, mve_pred16_t p)	p -> Rn	VQRSHLT.S8 Qda,Rm		
int16x8_t [arm_]vqrshlq_m_n[_s16](int16x8_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
int32x4_t [arm_]vqrshlq_m_n[_s32](int32x4_t a,	p -> Rp a -> Qda	VQRSHLT.S16 Qda,Rm VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST	Qua -> resuit	WVE
_ · · _• _ •	p -> Rp	VQRSHLT.S32 Qda,Rm		1
uint8x16_t [arm_]vqrshlq_m_n[_u8](uint8x16_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VQRSHLT.U8 Qda,Rm		
uint16x8_t [arm_]vqrshlq_m_n[_u16](uint16x8_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
nint20v4 t [ orm ]varshla m n[ v20](vint20v4 t -	p -> Rp	VQRSHLT.U16 Qda,Rm	Odo > monute	MVE
uint32x4_t [arm_]vqrshlq_m_n[_u32](uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
			I	
moz_to, mve_predio_tp)	p -> Rp	VQRSHLT.U32 Qda,Rm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [_arm_]vqrshlq[_s16](int16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VQRSHL.S16 Qd,Qm,Qn	Qd -> result	MVE/NEON
int32x4_t [arm_]vqrshlq[_s32](int32x4_t a, int32x4_t b)	a -> Qm b -> Qn	VQRSHL.S32 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vqrshlq[_u8](uint8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VQRSHL.U8 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vqrshlq[_u16](uint16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VQRSHL.U16 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint32x4_t [arm_]vqrshlq[_u32](uint32x4_t a, int32x4_t b)	a -> Qm b -> On	VQRSHL.U32 Qd,Qm,Qn	Qd -> result	MVE/NEON
int8x16_t [arm_]vqrshlq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.S8 Qd,Qm,Qn	Qd -> result	MVE
int16x8_t [arm_]vqrshlq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.S16 Qd,Qm,Qn	Qd -> result	MVE
int32x4_t [_arm_]vqrshlq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.S32 Qd,Qm,Qn	Qd -> result	MVE
uint8x16_t [_arm_]vqrshlq_m[_u8](uint8x16_t inactive, uint8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.U8 Qd,Qm,Qn	Qd -> result	MVE
uint16x8_t [_arm_]vqrshlq_m[_u16](uint16x8_t inactive, uint16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.U16 Qd,Qm,Qn	Qd -> result	MVE
uint32x4_t [_arm_]vqrshlq_m[_u32](uint32x4_t inactive, uint32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.U32 Qd,Qm,Qn	Qd -> result	MVE
int8x16_t [_arm_]vqrshrnbq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQRSHRNB.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vqrshrnbq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQRSHRNB.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqrshrnbq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQRSHRNB.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqrshrnbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQRSHRNB.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vqrshrnbq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQRSHRNBT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vqrshrnbq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQRSHRNBT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqrshrnbq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQRSHRNBT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqrshrnbq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQRSHRNBT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vqrshrntq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQRSHRNT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vqrshrntq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQRSHRNT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqrshrntq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQRSHRNT.U16 Qd,Qm,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [_arm_]vqrshrntq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQRSHRNT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vqrshrntq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQRSHRNTT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vqrshrntq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQRSHRNTT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqrshrntq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQRSHRNTT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vqrshrntq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQRSHRNTT.U32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqrshrunbq[_n_s16](uint8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQRSHRUNB.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqrshrunbq[_n_s32](uint16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQRSHRUNB.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqrshrunbq_m[_n_s16](uint8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQRSHRUNBT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqrshrunbq_m[_n_s32](uint16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQRSHRUNBT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqrshruntq[_n_s16](uint8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQRSHRUNT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vqrshruntq[_n_s32](uint16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQRSHRUNT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [arm_]vqrshruntq_m[_n_s16](uint8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQRSHRUNTT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqrshruntq_m[_n_s32](uint16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQRSHRUNTT.S32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vqshlq[_s8](int8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VQSHL.S8 Qd,Qm,Qn	Qd -> result	MVE/NEON
int16x8_t [_arm_]vqshlq[_s16](int16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VQSHL.S16 Qd,Qm,Qn	Qd -> result	MVE/NEON
int32x4_t [_arm_]vqshlq[_s32](int32x4_t a, int32x4_t b)	a -> Qm b -> Qn	VQSHL.S32 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vqshlq[_u8](uint8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VQSHL.U8 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vqshlq[_u16](uint16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VQSHL.U16 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vqshlq[_u32](uint32x4_t a, int32x4_t b)	a -> Qm b -> Qn	VQSHL.U32 Qd,Qm,Qn	Qd -> result	MVE/NEON
int8x16_t [_arm_]vqshlq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQSHLT.S8 Qd,Qm,Qn	Qd -> result	MVE
int16x8_t [arm]vqshlq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQSHLT.S16 Qd,Qm,Qn	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [_arm_]vqshlq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQSHLT.S32 Qd,Qm,Qn	Qd -> result	MVE
uint8x16_t [arm_]vqshlq_m[_u8](uint8x16_t inactive, uint8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQSHLT.U8 Qd,Qm,Qn	Qd -> result	MVE
uint16x8_t [_arm_]vqshlq_m[_u16](uint16x8_t inactive, uint16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQSHLT.U16 Qd,Qm,Qn	Qd -> result	MVE
uint32x4_t [_arm_]vqshlq_m[_u32](uint32x4_t inactive, uint32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQSHLT.U32 Qd,Qm,Qn	Qd -> result	MVE
int8x16_t [_arm_]vqshlq_n[_s8](int8x16_t a, const int imm)	a -> Qm 0 <= imm <= 7	VQSHL.S8 Qd,Qm,#imm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vqshlq_n[_s16](int16x8_t a, const int imm)	a -> Qm 0 <= imm <= 15	VQSHL.S16 Qd,Qm,#imm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vqshlq_n[_s32](int32x4_t a, const int imm)	a -> Qm 0 <= imm <= 31	VQSHL.S32 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vqshlq_n[_u8](uint8x16_t a, const int imm)	a -> Qm 0 <= imm <= 7	VQSHL.U8 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vqshlq_n[_u16](uint16x8_t a, const int imm)	a -> Qm 0 <= imm <= 15	VQSHL.U16 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vqshlq_n[_u32](uint32x4_t a, const int imm)	a -> Qm 0 <= imm <= 31	VQSHL.U32 Qd,Qm,#imm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vqshlq_m_n[_s8](int8x16_t inactive, int8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VQSHLT.S8 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [_arm_]vqshlq_m_n[_s16](int16x8_t inactive, int16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 15	VMSR P0,Rp VPST VQSHLT.S16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vqshlq_m_n[_s32](int32x4_t inactive, int32x4_t a, const int imm, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm 0 <= imm <= 31	VMSR P0,Rp VPST VQSHLT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqshlq_m_n[_u8](uint8x16_t inactive, uint8x16_t a, const int imm, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VQSHLT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqshlq_m_n[_u16](uint16x8_t inactive, uint16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VQSHLT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vqshlq_m_n[_u32](uint32x4_t inactive, uint32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VQSHLT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vqshlq_r[_s8](int8x16_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [arm_]vqshlq_r[_s16](int16x8_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vqshlq_r[_s32](int32x4_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vqshlq_r[_u8](uint8x16_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [_arm_]vqshlq_r[_u16](uint16x8_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vqshlq_r[_u32](uint32x4_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [arm_]vqshlq_m_r[_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQSHLT.S8 Qda,Rm	Qda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vqshlq_m_r[_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
·	p -> Rp	VQSHLT.S16 Qda,Rm		
int32x4_t [arm_]vqshlq_m_r[_s32](int32x4_t a, int32_t	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VQSHLT.S32 Qda,Rm		
uint8x16_t [arm_]vqshlq_m_r[_u8](uint8x16_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
uint16x8_t [arm_]vqshlq_m_r[_u16](uint16x8_t a,	p -> Rp a -> Qda	VQSHLT.U8 Qda,Rm VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VMSK FO,KP VPST	Qua -> resuit	MIVE
	p -> Rp	VQSHLT.U16 Qda,Rm		
uint32x4_t [arm_]vqshlq_m_r[_u32](uint32x4_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VQSHLT.U32 Qda,Rm		
uint8x16_t [arm_]vqshluq[_n_s8](int8x16_t a, const int	a -> Qm	VQSHLU.S8 Qd,Qm,#imm	Qd -> result	MVE
imm)	0 <= imm <= 7			1.00
uint16x8_t [arm_]vqshluq[_n_s16](int16x8_t a, const int imm)	a -> Qm 0 <= imm <= 15	VQSHLU.S16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [arm_]vqshluq[_n_s32](int32x4_t a, const	a -> Qm	VQSHLU.S32 Qd,Qm,#imm	Qd -> result	MVE
int imm)	0 <= imm <=			
uint8x16 t [ arm ]vqshluq m[ n s8](uint8x16 t	31 inactive -> Od	VMSR P0,Rp	Od -> result	MVE
inactive, int8x16_t a, const int imm, mve_pred16_t p)	a -> Qm	VPST	Qu > result	III V E
	0 <= imm <= 7	VQSHLUT.S8 Qd,Qm,#imm		
uint16x8 t[ arm ]vqshluq m[ n s16](uint16x8 t	p -> Rp inactive -> Od	VMSR P0,Rp	Od -> result	MVE
inactive, int16x8_t a, const int imm, mve_pred16_t p)	a -> Om	VPST	Qu -> result	WIVE
,,,	0 <= imm <=	VQSHLUT.S16 Qd,Qm,#imm		
	15			
uint32x4_t [arm_]vqshluq_m[_n_s32](uint32x4_t	p -> Rp inactive -> Od	VMSR P0,Rp	Od -> result	MVE
inactive, int32x4_t a, const int imm, mve_pred16_t p)	a -> Qm	VPST	Q	1
	0 <= imm <=	VQSHLUT.S32 Qd,Qm,#imm		
	31 p -> Rp			
int8x16_t [arm_]vqshrnbq[_n_s16](int8x16_t a,	a -> Qd	VQSHRNB.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t b, const int imm)	b -> Qm			
int16x8_t [arm_]vqshrnbq[_n_s32](int16x8_t a,	1 <= imm <= 8 a -> Od	VQSHRNB.S32 Qd,Qm,#imm	Od -> result	MVE
int32x4_t b, const int imm)	b -> Qm	V QSTICLVB.552 Qu,QIII,#IIIIII	Qu > result	III V E
	1 <= imm <=			
uint8x16_t [arm_]vqshrnbq[_n_u16](uint8x16_t a,	16 a -> Qd	VQSHRNB.U16 Qd,Qm,#imm	Od -> result	MVE
uint16x8_t b, const int imm)	b -> Qm	VQSTIKIVB.010 Qu,Qiii,#iiiiii	Qu -> result	IVI V E
	1 <= imm <= 8			
uint16x8_t [arm_]vqshrnbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Om	VQSHRNB.U32 Qd,Qm,#imm	Qd -> result	MVE
umt32x4_t b, const mt mmn)	1 <= imm <=			
	16			
int8x16_t [arm_]vqshrnbq_m[_n_s16](int8x16_t a, int16x8 t b, const int imm, mve_pred16_t p)	a -> Qd b -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
introxe_t b, const int inini, inve_pred1o_t p)	1 <= imm <= 8	VQSHRNBT.S16 Qd,Qm,#imm		
	p -> Rp			
int16x8_t [_arm_]vqshrnbq_m[_n_s32](int16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, const int imm, mve_pred16_t p)	b -> Qm 1 <= imm <=	VPST VQSHRNBT.S32 Qd,Qm,#imm		
	16			
mintOr16 4 f ages breshouts out a 1627 to 16	p -> Rp	VMCD DO D	04 1	MVE
uint8x16_t [arm_]vqshrnbq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
,,	1 <= imm <= 8	VQSHRNBT.U16 Qd,Qm,#imm		
	p -> Rp	An ago po p		) Auto
uint16x8_t [arm_]vqshrnbq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
and 274-1, to, const lift lillin, inve_prod10_t p)	1 <= imm <=	VQSHRNBT.U32 Qd,Qm,#imm		
	16			
int8x16_t [arm_]vqshrntq[_n_s16](int8x16_t a,	p -> Rp a -> Qd	VQSHRNT.S16 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vqsnrntq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qa b -> Qm	v Q311/11/1.310 Qu,Qm,#IMM	Qu -> resuit	IVI V E
	1 <= imm <= 8			
int16x8_t [_arm_]vqshrntq[_n_s32](int16x8_t a,	a -> Qd	VQSHRNT.S32 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t b, const int imm)	b -> Qm 1 <= imm <=			
	16		1	I

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [_arm_]vqshrntq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQSHRNT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqshrntq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQSHRNT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vqshrntq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQSHRNTT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vqshrntq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQSHRNTT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [arm_]vqshrntq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQSHRNTT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqshrntq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQSHRNTT.U32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [arm_]vqshrunbq[_n_s16](uint8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQSHRUNB.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqshrunbq[_n_s32](uint16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQSHRUNB.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqshrunbq_m[_n_s16](uint8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQSHRUNBT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vqshrunbq_m[_n_s32](uint16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQSHRUNBT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [arm_]vqshruntq[_n_s16](uint8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQSHRUNT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqshruntq[_n_s32](uint16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQSHRUNT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqshruntq_m[_n_s16](uint8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQSHRUNTT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vqshruntq_m[_n_s32](uint16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQSHRUNTT.S32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vrev16q[_s8](int8x16_t a)	a -> Qm	VREV16.8 Qd,Qm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vrev16q[_u8](uint8x16_t a) int8x16_t [_arm_]vrev16q_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	a -> Qm inactive -> Qd a -> Qm p -> Rp	VREV16.8 Qd,Qm  VMSR P0,Rp  VPST  VREV16T.8 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE
uint8x16_t [_arm_]vrev16q_m[_u8](uint8x16_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV16T.8 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vrev16q_x[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VREV16T.8 Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vrev16q_x[_u8](uint8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VREV16T.8 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vrev32q[_s8](int8x16_t a)	a -> Qm	VREV32.8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vrev32q[_s16](int16x8_t a)	a -> Qm	VREV32.16 Qd,Qm VREV32.8 Qd,Qm	Qd -> result	MVE/NEON MVE/NEON
uint8x16_t [arm_]vrev32q[_u8](uint8x16_t a) uint16x8_t [arm_]vrev32q[_u16](uint16x8_t a)	a -> Qm a -> Qm	VREV32.8 Qd,Qm VREV32.16 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
float16x8_t [arm_]vrev32q[_f16](float16x8_t a)	a -> Qm	VREV32.16 Qd,Qm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vrev32q_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
1.15.0.15	p -> Rp	VREV32T.8 Qd,Qm	0.1	) am
int16x8_t [arm_]vrev32q_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
introxo_t u, invo_prouro_t p)	p -> Rp	VREV32T.16 Qd,Qm		
uint8x16_t [arm_]vrev32q_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VREV32T.8 Qd,Qm	0.1	MVE
uint16x8_t [arm_]vrev32q_m[_u16](uint16x8_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, unitroxo_t a, mve_pred10_t p)	p -> Rp	VREV32T.16 Qd,Qm		
float16x8_t [arm_]vrev32q_m[_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VREV32T.16 Qd,Qm		
int8x16_t [arm_]vrev32q_x[_s8](int8x16_t a,	a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VREV32T.8 Qd,Qm		
int16x8_t [arm_]vrev32q_x[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VREV32T.16 Qd,Qm		
uint8x16_t [arm_]vrev32q_x[_u8](uint8x16_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VREV32T.8 Qd,Qm		
uint16x8_t [arm_]vrev32q_x[_u16](uint16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu > result	III V E
	1	VREV32T.16 Qd,Qm		
float16x8_t [arm_]vrev32q_x[_f16](float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
int8x16_t [arm_]vrev64q[_s8](int8x16_t a)	0 > Om	VREV32T.16 Qd,Qm VREV64.8 Qd,Qm	Od -> result	MVE/NEON
int8x16_t [arm_]vrev64q[_s8](int8x16_t a) int16x8_t [arm_]vrev64q[_s16](int16x8_t a)	a -> Qm a -> Qm	VREV64.8 Qd,Qfff VREV64.16 Qd,Qm	Qd -> result	MVE/NEON MVE/NEON
int32x4_t [arm_]vrev64q[_s32](int32x4_t a)	a -> Qm	VREV64.32 Qd,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vrev64q[_u8](uint8x16_t a)	a -> Qm	VREV64.8 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vrev64q[_u16](uint16x8_t a)	a -> Qm	VREV64.16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vrev64q[_u32](uint32x4_t a)	a -> Qm	VREV64.32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vrev64q[_f16](float16x8_t a)	a -> Qm	VREV64.16 Qd,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]vrev64q[_f32](float32x4_t a)	a -> Qm	VREV64.32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vrev64q_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mtox10_t a, mve_pred10_t p)	p -> Rp	VREV64T.8 Qd,Qm		
int16x8_t [arm_]vrev64q_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VREV64T.16 Qd,Qm		
int32x4_t [_arm_]vrev64q_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VREV64T.32 Qd,Qm		
uint8x16_t [arm_]vrev64q_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
uint8x16_t a, mve_pred16_t p)	a -> Qm	VPST	Q	
	p -> Rp	VREV64T.8 Qd,Qm		
uint16x8_t [arm_]vrev64q_m[_u16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VREV64T.16 Qd,Qm		
uint32x4_t [arm_]vrev64q_m[_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, uint32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Qu > result	III V E
	p -> Rp	VREV64T.32 Qd,Qm		
float16x8_t [arm_]vrev64q_m[_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
float32x4 t [ arm ]vrev64q m[ f32](float32x4 t	p -> Rp inactive -> Qd	VREV64T.16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Qu -> result	MVL
	p -> Rp	VREV64T.32 Qd,Qm		
int8x16_t [arm_]vrev64q_x[_s8](int8x16_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
int16v9 t1	0.5000	VREV64T.8 Qd,Qm	04	MVE
int16x8_t [arm_]vrev64q_x[_s16](int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
mvc_pred10_t p)	h -> wh	VREV64T.16 Qd,Qm		
int32x4_t [arm_]vrev64q_x[_s32](int32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VREV64T.32 Qd,Qm		
uint8x16_t [arm_]vrev64q_x[_u8](uint8x16_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
	1	VREV64T.8 Qd,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vrev64q_x[_u16](uint16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VREV64T.16 Qd,Qm		
uint32x4_t [_arm_]vrev64q_x[_u32](uint32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
float16x8_t [_arm_]vrev64q_x[_f16](float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VREV64T.32 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
float32x4_t [arm_]vrev64q_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VREV64T.16 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
int8x16_t [_arm_]vrshlq[_n_s8](int8x16_t a, int32_t b)	a -> Qda	VREV64T.32 Qd,Qm VRSHL.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [arm_]vrshlq[_n_s16](int16x8_t a, int32_t b)	b -> Rm a -> Qda	VRSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vrshlq[_n_s32](int32x4_t a, int32_t b)	b -> Rm a -> Qda	VRSHL.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [arm_]vrshlq[_n_u8](uint8x16_t a, int32_t b)	b -> Rm a -> Qda	VRSHL.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [_arm_]vrshlq[_n_u16](uint16x8_t a, int32_t	b -> Rm a -> Qda	VRSHL.U16 Qda,Rm	Qda -> result	MVE
b) uint32x4_t [arm_]vrshlq[_n_u32](uint32x4_t a, int32_t b)	b -> Rm a -> Qda b -> Rm	VRSHL.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [arm_]vrshlq_m_n[_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
int16x8 t [ arm ]vrshlq m n[ s16](int16x8 t a, int32 t	p -> Rp a -> Oda	VRSHLT.S8 Qda,Rm VMSR P0,Rp	Oda -> result	MVE
b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VRSHLT.S16 Qda,Rm	Qua > result	WY E
int32x4_t [_arm_]vrshlq_m_n[_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VRSHLT.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vrshlq_m_n[_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VRSHLT.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [arm_]vrshlq_m_n[_u16](uint16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VRSHLT.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [arm_]vrshlq_m_n[_u32](uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VRSHLT.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [arm_]vrshlq[_s8](int8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VRSHL:S8 Qd,Qm,Qn	Qd -> result	MVE/NEON
int16x8_t [arm_]vrshlq[_s16](int16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VRSHL.S16 Qd,Qm,Qn	Qd -> result	MVE/NEON
int32x4_t [arm_]vrshlq[_s32](int32x4_t a, int32x4_t b)	a -> Qm b -> Qn	VRSHL.S32 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint8x16_t [arm_]vrshlq[_u8](uint8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VRSHL.U8 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint16x8_t [arm_]vrshlq[_u16](uint16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VRSHL.U16 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vrshlq[_u32](uint32x4_t a, int32x4_t b)	a -> Qm b -> Qn	VRSHL.U32 Qd,Qm,Qn	Qd -> result	MVE/NEON
int8x16_t [_arm_]vrshlq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.S8 Qd,Qm,Qn	Qd -> result	MVE
int16x8_t [arm_]vrshlq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.S16 Qd,Qm,Qn	Qd -> result	MVE
int32x4_t [arm_]vrshlq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.S32 Qd,Qm,Qn	Qd -> result	MVE
uint8x16_t [_arm_]vrshlq_m[_u8](uint8x16_t inactive, uint8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.U8 Qd,Qm,Qn	Qd -> result	MVE
uint16x8_t [_arm_]vrshlq_m[_u16](uint16x8_t inactive, uint16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.U16 Qd,Qm,Qn	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vrshlq_m[_u32](uint32x4_t inactive, uint32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn	VMSR P0,Rp VPST VRSHLT.U32 Qd,Qm,Qn	Qd -> result	MVE
int8x16_t [_arm_]vrshlq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	p -> Rp a -> Qm b -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t [arm_]vrshlq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	p -> Rp a -> Qm b -> Qn p -> Rp	VRSHLT.S8 Qd,Qm,Qn VMSR P0,Rp VPST VRSHLT.S16 Qd,Qm,Qn	Qd -> result	MVE
int32x4_t [_arm_]vrshlq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.S32 Qd,Qm,Qn	Qd -> result	MVE
uint8x16_t [_arm_]vrshlq_x[_u8](uint8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.U8 Qd,Qm,Qn	Qd -> result	MVE
uint16x8_t [_arm_]vrshlq_x[_u16](uint16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.U16 Qd,Qm,Qn	Qd -> result	MVE
uint32x4_t [_arm_]vrshlq_x[_u32](uint32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.U32 Qd,Qm,Qn	Qd -> result	MVE
int8x16_t [_arm_]vshlcq[_s8](int8x16_t a, uint32_t * b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
int16x8_t [arm_]vshlcq[_s16](int16x8_t a, uint32_t * b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
int32x4_t [_arm_]vshlcq[_s32](int32x4_t a, uint32_t * b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint8x16_t [_arm_]vshlcq[_u8](uint8x16_t a, uint32_t * b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint16x8_t [arm_]vshlcq[_u16](uint16x8_t a, uint32_t * b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint32x4_t [_arm_]vshlcq[_u32](uint32x4_t a, uint32_t * b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
int8x16_t [arm_]vshlcq_m[_s8](int8x16_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
int16x8_t [arm_]vshlcq_m[_s16](int16x8_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
int32x4_t [arm_]vshlcq_m[_s32](int32x4_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint8x16_t [_arm_]vshlcq_m[_u8](uint8x16_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint16x8_t [_arm_]vshlcq_m[_u16](uint16x8_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint32x4_t [_arm_]vshlcq_m[_u32](uint32x4_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vshllbq[_n_s8](int8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VSHLLB.S8 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshllbq[_n_s16](int16x8_t a, const int imm)	a -> Qm 1 <= imm <= 16	VSHLLB.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshllbq[_n_u8](uint8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VSHLLB.U8 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vshllbq[_n_u16](uint16x8_t a, const int imm)	a -> Qm 1 <= imm <= 16	VSHLLB.U16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vshllbq_m[_n_s8](int16x8_t inactive, int8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHLLBT.S8 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshllbq_m[_n_s16](int32x4_t inactive, int16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHLLBT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vshllbq_m[_n_u8](uint16x8_t inactive, uint8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHLLBT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vshllbq_m[_n_u16](uint32x4_t inactive, uint16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHLLBT.U16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [_arm_]vshllbq_x[_n_s8](int8x16_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHLLBT.S8 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshllbq_x[_n_s16](int16x8_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHLLBT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshllbq_x[_n_u8](uint8x16_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHLLBT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [arm_]vshllbq_x[_n_u16](uint16x8_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHLLBT.U16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [_arm_]vshlltq[_n_s8](int8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VSHLLT.S8 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshlltq[_n_s16](int16x8_t a, const int imm)	a -> Qm 1 <= imm <= 16	VSHLLT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshlltq[_n_u8](uint8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VSHLLT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vshlltq[_n_u16](uint16x8_t a, const int imm)	a -> Qm 1 <= imm <=	VSHLLT.U16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vshlltq_m[_n_s8](int16x8_t inactive, int8x16_t a, const int imm, mve_pred16_t p)	16 inactive -> Qd a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHLLTT.S8 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshlltq_m[_n_s16](int32x4_t inactive, int16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 16	VMSR P0,Rp VPST VSHLLTT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vshlltq_m[_n_u8](uint16x8_t inactive, uint8x16_t a, const int imm, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHLLTT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vshlltq_m[_n_u16](uint32x4_t inactive, uint16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHLLTT.U16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [_arm_]vshlltq_x[_n_s8](int8x16_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHLLTT.S8 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshlltq_x[_n_s16](int16x8_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHLLTT.S16 Qd,Qm,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vshlltq_x[_n_u8](uint8x16_t a, const	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
int imm, mve_pred16_t p)	1 <= imm <= 8	VPST		
uint32x4_t [arm_]vshlltq_x[_n_u16](uint16x8_t a, const	p -> Rp a -> Qm	VSHLLTT.U8 Qd,Qm,#imm VMSR P0,Rp	Qd -> result	MVE
int imm, mve_pred16_t p)	1 <= imm <=	VPST	Qu => resuit	WIVE
	16	VSHLLTT.U16 Qd,Qm,#imm		
int8x16_t [arm_]vshlq[_s8](int8x16_t a, int8x16_t b)	p -> Rp a -> Qm	VSHL.S8 Qd,Qm,Qn	Od -> result	MVE/NEON
	b -> Qn	2 / 2 / 2		
int16x8_t [arm_]vshlq[_s16](int16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VSHL.S16 Qd,Qm,Qn	Qd -> result	MVE/NEON
int32x4_t [arm_]vshlq[_s32](int32x4_t a, int32x4_t b)	a -> Qm	VSHL.S32 Qd,Qm,Qn	Qd -> result	MVE/NEON
	b -> Qn	VCIII II0 O4 O O	0.1 >1t	MVEALEON
uint8x16_t [arm_]vshlq[_u8](uint8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VSHL.U8 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint16x8_t [arm_]vshlq[_u16](uint16x8_t a, int16x8_t	a -> Qm	VSHL.U16 Qd,Qm,Qn	Qd -> result	MVE/NEON
b) uint32x4_t [arm_]vshlq[_u32](uint32x4_t a, int32x4_t	b -> Qn a -> Qm	VSHL.U32 Qd,Qm,Qn	Od -> result	MVE/NEON
b)	b -> Qn	VSIIE.032 Qu,Qiii,Qii	Qu -> resuit	WIVE/IVEOIV
int8x16_t [_arm_]vshlq_m[_s8](int8x16_t inactive,	inactive -> Qd a -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	b -> Qn	VSHLT.S8 Od,Om,On		
	p -> Rp			
int16x8_t [arm_]vshlq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
merono_t u, merono_t o, mve_prouro_t p/	b -> Qn	VSHLT.S16 Qd,Qm,Qn		
int22v4 t.f. com lychla mf c22l/int22v4 timestive	p -> Rp inactive -> Qd	VMCD DO Do	Od > magnit	MVE
int32x4_t [arm_]vshlq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Om	VMSR P0,Rp VPST	Qd -> result	IVI V E
- / - <b>-</b> / - <b>-</b> /	b -> Qn	VSHLT.S32 Qd,Qm,Qn		
uint8x16_t [arm_]vshlq_m[_u8](uint8x16_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
uint8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qm	VPST	Qu > resuit	W. C.
	b -> Qn	VSHLT.U8 Qd,Qm,Qn		
uint16x8_t [arm_]vshlq_m[_u16](uint16x8_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qm	VPST		
	b -> Qn p -> Rp	VSHLT.U16 Qd,Qm,Qn		
uint32x4_t [arm_]vshlq_m[_u32](uint32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qm b -> Qn	VPST VSHLT.U32 Qd,Qm,Qn		
	p -> Rp	V311E1:032 Qu,QIII,QII		
int8x16_t [_arm_]vshlq_x[_s8](int8x16_t a, int8x16_t b,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Qn p -> Rp	VPST VSHLT.S8 Qd,Qm,Qn		
int16x8_t [arm_]vshlq_x[_s16](int16x8_t a, int16x8_t b,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Qn p -> Rp	VPST VSHLT.S16 Qd,Qm,Qn		
int32x4_t [arm_]vshlq_x[_s32](int32x4_t a, int32x4_t b,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Qn	VPST		
uint8x16_t [arm_]vshlq_x[_u8](uint8x16_t a, int8x16_t	p -> Rp a -> Qm	VSHLT.S32 Qd,Qm,Qn VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qn	VPST		
uint16x8_t [arm_]vshlq_x[_u16](uint16x8_t a,	p -> Rp a -> Qm	VSHLT.U8 Qd,Qm,Qn VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qn	VPST	Qu'y resuit	
uint32x4 t [ arm ]vshlq x[ u32](uint32x4 t a,	p -> Rp a -> Qm	VSHLT.U16 Qd,Qm,Qn VMSR P0,Rp	Qd -> result	MVE
int32x4_t [armjvsinq_x[_u32](unit32x4_t a, int32x4_t b, mve_pred16_t p)	b -> Qn	VPST	Qu -> result	WIVE
1.016.15	p -> Rp	VSHLT.U32 Qd,Qm,Qn	01	NOTE
int8x16_t [arm_]vshlq_n[_s8](int8x16_t a, const int imm)	a -> Qm 0 <= imm <= 7	VSHL.S8 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vshlq_n[_s16](int16x8_t a, const int	a -> Qm	VSHL.S16 Qd,Qm,#imm	Qd -> result	MVE
imm)	0 <= imm <= 15			
int32x4_t [arm_]vshlq_n[_s32](int32x4_t a, const int	a -> Qm	VSHL.S32 Qd,Qm,#imm	Qd -> result	MVE
imm)	0 <= imm <= 31			
uint8x16_t [arm_]vshlq_n[_u8](uint8x16_t a, const int	a -> Qm	VSHL.U8 Qd,Qm,#imm	Qd -> result	MVE
imm)	0 <= imm <= 7			
uint16x8_t [arm_]vshlq_n[_u16](uint16x8_t a, const int imm)	a -> Qm 0 <= imm <=	VSHL.U16 Qd,Qm,#imm	Qd -> result	MVE
	15			
uint32x4_t [arm_]vshlq_n[_u32](uint32x4_t a, const int	a -> Qm 0 <= imm <=	VSHL.U32 Qd,Qm,#imm	Qd -> result	MVE
imm)				

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vshlq_m_n[_s8](int8x16_t inactive, int8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VSHLT.S8 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vshlq_m_n[_s16](int16x8_t inactive, int16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VSHLT.S16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshlq_m_n[_s32](int32x4_t inactive, int32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VSHLT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vshlq_m_n[_u8](uint8x16_t inactive, uint8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VSHLT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshlq_m_n[_u16](uint16x8_t inactive, uint16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VSHLT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vshlq_m_n[_u32](uint32x4_t inactive, uint32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VSHLT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vshlq_x_n[_s8](int8x16_t a, const int imm, mve_pred16_t p)	a -> Qm 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VSHLT.S8 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vshlq_x_n[_s16](int16x8_t a, const int imm, mve_pred16_t p)	a -> Qm 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VSHLT.S16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshlq_x_n[_s32](int32x4_t a, const int imm, mve_pred16_t p)	a -> Qm 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VSHLT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vshlq_x_n[_u8](uint8x16_t a, const int imm, mve_pred16_t p)	a -> Qm 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VSHLT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vshlq_x_n[_u16](uint16x8_t a, const int imm, mve_pred16_t p)	a -> Qm 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VSHLT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vshlq_x_n[_u32](uint32x4_t a, const int imm, mve_pred16_t p)	a -> Qm 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VSHLT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vshlq_r[_s8](int8x16_t a, int32_t b)	a -> Qda b -> Rm	VSHL.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [arm_]vshlq_r[_s16](int16x8_t a, int32_t b)	a -> Qda b -> Rm	VSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vshlq_r[_s32](int32x4_t a, int32_t b)	a -> Qda b -> Rm	VSHL.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vshlq_r[_u8](uint8x16_t a, int32_t b)	a -> Qda b -> Rm	VSHL.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [_arm_]vshlq_r[_u16](uint16x8_t a, int32_t b)	a -> Qda b -> Rm	VSHL.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [arm_]vshlq_r[_u32](uint32x4_t a, int32_t b)	a -> Qda b -> Rm	VSHL.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [arm_]vshlq_m_r[_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [arm_]vshlq_m_r[_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vshlq_m_r[_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vshlq_m_r[_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [arm_]vshlq_m_r[_u16](uint16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.U16 Qda,Rm	Qda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [arm_]vshlq_m_r[_u32](uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [arm_]vrshrnbq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VRSHRNB.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vrshrnbq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VRSHRNB.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vrshrnbq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VRSHRNB.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vrshrnbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VRSHRNB.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vrshrnbq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNBT.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vrshrnbq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNBT.132 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vrshrnbq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNBT.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vrshrnbq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNBT.132 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vrshrntq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VRSHRNT.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vrshrntq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VRSHRNT.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vrshrntq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VRSHRNT.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vrshrntq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VRSHRNT.132 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vrshrntq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNTT.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vrshrntq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNTT.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vrshrntq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNTT.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vrshrntq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNTT.132 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vrshrq[_n_s8](int8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VRSHR.S8 Qd,Qm,#imm	Qd -> result	MVE/NEON
int16x8_t [arm_]vrshrq[_n_s16](int16x8_t a, const int imm)	a -> Qm 1 <= imm <= 16	VRSHR.S16 Qd,Qm,#imm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vrshrq[_n_s32](int32x4_t a, const int imm)	a -> Qm 1 <= imm <= 32	VRSHR.S32 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vrshrq[_n_u8](uint8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VRSHR.U8 Qd,Qm,#imm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vrshrq[_n_u16](uint16x8_t a, const int imm)	a -> Qm 1 <= imm <= 16	VRSHR.U16 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vrshrq[_n_u32](uint32x4_t a, const int imm)	a -> Qm 1 <= imm <= 32	VRSHR.U32 Qd,Qm,#imm	Qd -> result	MVE/NEON
int8x16_t [arm_]vrshrq_m[_n_s8](int8x16_t inactive, int8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 8	VMSR P0,Rp VPST VRSHRT.S8 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [_arm_]vrshrq_m[_n_s16](int16x8_t inactive, int16x8_t a, const int imm, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm 1 <= imm <= 16	VMSR P0,Rp VPST VRSHRT.S16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vrshrq_m[_n_s32](int32x4_t inactive, int32x4_t a, const int imm, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm 1 <= imm <= 32	VMSR P0,Rp VPST VRSHRT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vrshrq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, const int imm, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vrshrq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 16	VMSR P0,Rp VPST VRSHRT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vrshrq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, const int imm, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm 1 <= imm <= 32	VMSR P0,Rp VPST VRSHRT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vrshrq_x[_n_s8](int8x16_t a, const int imm, mve_pred16_t p)	p -> Rp a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRT.S8 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [_arm_]vrshrq_x[_n_s16](int16x8_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 16	VMSR P0,Rp VPST VRSHRT.S16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vrshrq_x[_n_s32](int32x4_t a, const int imm, mve_pred16_t p)	p -> Rp a -> Qm 1 <= imm <= 32	VMSR P0,Rp VPST VRSHRT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vrshrq_x[_n_u8](uint8x16_t a, const int imm, mve_pred16_t p)	p -> Rp a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vrshrq_x[_n_u16](uint16x8_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 16	VMSR P0,Rp VPST VRSHRT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vrshrq_x[_n_u32](uint32x4_t a, const int imm, mve_pred16_t p)	p -> Rp a -> Qm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VRSHRT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vshmbq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VSHRNB.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vshrnbq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VSHRNB.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vshrnbq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VSHRNB.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshrnbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VSHRNB.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vshmbq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHRNBT.I16 Qd,Qm,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vshrnbq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHRNBT.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vshrnbq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHRNBT.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshrnbq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHRNBT.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vshrntq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VSHRNT.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vshrntq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VSHRNT.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vshrntq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VSHRNT.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshrntq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VSHRNT.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vshrntq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHRNTT.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vshrntq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHRNTT.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vshrntq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHRNTT.II6 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshrntq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHRNTT.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vshrq[_n_s8](int8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VSHR.S8 Qd,Qm,#imm	Qd -> result	MVE/NEON
int16x8_t [arm_]vshrq[_n_s16](int16x8_t a, const int imm)	a -> Qm 1 <= imm <= 16	VSHR.S16 Qd,Qm,#imm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vshrq[_n_s32](int32x4_t a, const int imm)	a -> Qm 1 <= imm <= 32	VSHR.S32 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vshrq[_n_u8](uint8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VSHR.U8 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vshrq[_n_u16](uint16x8_t a, const int imm)	a -> Qm 1 <= imm <= 16	VSHR.U16 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vshrq[_n_u32](uint32x4_t a, const int imm)	a -> Qm 1 <= imm <= 32	VSHR.U32 Qd,Qm,#imm	Qd -> result	MVE/NEON
int8x16_t [arm_]vshrq_m[_n_s8](int8x16_t inactive, int8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHRT.S8 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vshrq_m[_n_s16](int16x8_t inactive, int16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHRT.S16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [arm_]vshrq_m[_n_s32](int32x4_t inactive, int32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHRT.S32 Qd,Qm,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [_arm_]vshrq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHRT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshrq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHRT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vshrq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHRT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vshrq_x[_n_s8](int8x16_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHRT.S8 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [_arm_]vshrq_x[_n_s16](int16x8_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHRT.S16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshrq_x[_n_s32](int32x4_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHRT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vshrq_x[_n_u8](uint8x16_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHRT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshrq_x[_n_u16](uint16x8_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHRT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vshrq_x[_n_u32](uint32x4_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHRT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vsliq[_n_s8](int8x16_t a, int8x16_t b, const int imm)	a -> Qd b -> Qm 0 <= imm <= 7	VSLI.8 Qd,Qm,#imm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vsliq[_n_s16](int16x8_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 0 <= imm <= 15	VSLI.16 Qd,Qm,#imm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vsliq[_n_s32](int32x4_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 0 <= imm <= 31	VSLI.32 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vsliq[_n_u8](uint8x16_t a, uint8x16_t b, const int imm)	a -> Qd b -> Qm 0 <= imm <= 7	VSLI.8 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vsliq[_n_u16](uint16x8_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 0 <= imm <= 15	VSLI.16 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vsliq[_n_u32](uint32x4_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 0 <= imm <= 31	VSLI.32 Qd,Qm,#imm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vsliq_m[_n_s8](int8x16_t a, int8x16_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VSLIT.8 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [_arm_]vsliq_m[_n_s16](int16x8_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VSLIT.16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vsliq_m[_n_s32](int32x4_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VSLIT.32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vsliq_m[_n_u8](uint8x16_t a, uint8x16_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VSLIT.8 Qd,Qm,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [_arm_]vsliq_m[_n_u16](uint16x8_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VSLIT.16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vsliq_m[_n_u32](uint32x4_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VSLIT.32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vsriq[_n_s8](int8x16_t a, int8x16_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VSRI.8 Qd,Qm,#imm	Qd -> result	MVE/NEON
int16x8_t [arm_]vsriq[_n_s16](int16x8_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VSRI.16 Qd,Qm,#imm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vsriq[_n_s32](int32x4_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 32	VSRI.32 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vsriq[_n_u8](uint8x16_t a, uint8x16_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VSRI.8 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vsriq[_n_u16](uint16x8_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VSRI.16 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vsriq[_n_u32](uint32x4_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 32	VSRI.32 Qd,Qm,#imm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vsriq_m[_n_s8](int8x16_t a, int8x16_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSRIT.8 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vsriq_m[_n_s16](int16x8_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSRIT.16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [arm_]vsriq_m[_n_s32](int32x4_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSRIT.32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [arm_]vsriq_m[_n_u8](uint8x16_t a, uint8x16_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSRIT.8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vsriq_m[_n_u16](uint16x8_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSRIT.16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vsriq_m[_n_u32](uint32x4_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSRIT.32 Qd,Qm,#imm	Qd -> result	MVE
float16_t [_arm_]vgetq_lane[_f16](float16x8_t a, const int idx)	$a \rightarrow Qn$ 0 <= idx <= 7	VMOV.U16 Rt,Qn[idx]	Rt -> result	MVE/NEON
float32_t [_arm_]vgetq_lane[_f32](float32x4_t a, const int idx)	$a \rightarrow Qn$ $0 \le idx \le 3$	VMOV.32 Rt,Qn[idx]	Rt -> result	MVE/NEON
int8_t [_arm_]vgetq_lane[_s8](int8x16_t a, const int idx)	a -> Qn 0 <= idx <= 15	VMOV.S8 Rt,Qn[idx]	Rt -> result	MVE/NEON
int16_t [_arm_]vgetq_lane[_s16](int16x8_t a, const int idx)	a -> Qn 0 <= idx <= 7	VMOV.S16 Rt,Qn[idx]	Rt -> result	MVE/NEON
int32_t [_arm_]vgetq_lane[_s32](int32x4_t a, const int idx)	a -> Qn 0 <= idx <= 3	VMOV.32 Rt,Qn[idx]	Rt -> result	MVE/NEON
int64_t [_arm_]vgetq_lane[_s64](int64x2_t a, const int idx)	a -> Qn 0 <= idx <= 1	VMOV Rt1,Rt2,D(2*n+idx)	[Rt1,Rt2] -> result	MVE/NEON
uint8_t [_arm_]vgetq_lane[_u8](uint8x16_t a, const int idx)	a -> Qn 0 <= idx <= 15	VMOV.U8 Rt,Qn[idx]	Rt -> result	MVE/NEON
uint16_t [arm_]vgetq_lane[_u16](uint16x8_t a, const int idx)	a -> Qn 0 <= idx <= 7	VMOV.U16 Rt,Qn[idx]	Rt -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32_t [_arm_]vgetq_lane[_u32](uint32x4_t a, const int idx)	a -> Qn 0 <= idx <= 3	VMOV.32 Rt,Qn[idx]	Rt -> result	MVE/NEON
uint64_t [_arm_]vgetq_lane[_u64](uint64x2_t a, const int idx)	a -> Qn 0 <= idx <= 1	VMOV Rt1,Rt2,D(2*n+idx)	[Rt1,Rt2] -> result	MVE/NEON
float16x8_t [_arm_]vsetq_lane[_f16](float16_t a, float16x8_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 7	VMOV.16 Qd[idx],Rt	Qd -> result	MVE/NEON
float32x4_t [_arm_]vsetq_lane[_f32](float32_t a, float32x4_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 3	VMOV.32 Qd[idx],Rt	Qd -> result	MVE/NEON
int8x16_t [_arm_]vsetq_lane[_s8](int8_t a, int8x16_t b, const int idx)	$a \rightarrow Rt$ $b \rightarrow Qd$ 0 <= idx <= 15	VMOV.8 Qd[idx],Rt	Qd -> result	MVE/NEON
int16x8_t [_arm_]vsetq_lane[_s16](int16_t a, int16x8_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 7	VMOV.16 Qd[idx],Rt	Qd -> result	MVE/NEON
int32x4_t [arm_]vsetq_lane[_s32](int32_t a, int32x4_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 3	VMOV.32 Qd[idx],Rt	Qd -> result	MVE/NEON
int64x2_t [arm_]vsetq_lane[_s64](int64_t a, int64x2_t b, const int idx)	a -> [Rt1,Rt2] b -> Qd 0 <= idx <= 1	VMOV D(2*d+idx),Rt1,Rt2	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vsetq_lane[_u8](uint8_t a, uint8x16_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 15	VMOV.8 Qd[idx],Rt	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vsetq_lane[_u16](uint16_t a, uint16x8_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 7	VMOV.16 Qd[idx],Rt	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vsetq_lane[_u32](uint32_t a, uint32x4_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 3	VMOV.32 Qd[idx],Rt	Qd -> result	MVE/NEON
uint64x2_t [_arm_]vsetq_lane[_u64](uint64_t a, uint64x2_t b, const int idx)	a -> [Rt1,Rt2] b -> Qd 0 <= idx <= 1	VMOV D(2*d+idx),Rt1,Rt2	Qd -> result	MVE/NEON
mve_pred16_t [arm_]vctp8q(uint32_t a)	a -> Rn	VCTP.8 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vctp16q(uint32_t a)	a -> Rn	VCTP.16 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vctp32q(uint32_t a)	a -> Rn	VCTP.32 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vctp64q(uint32_t a)	a -> Rn	VCTP.64 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vctp8q_m(uint32_t a, mve_pred16_t p)	a -> Rn p -> Rp	VMSR P0,Rp VPST VCTPT.8 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vctp16q_m(uint32_t a, mve_pred16_t p)	a -> Rn p -> Rp	VMSR P0,Rp VPST VCTPT.16 Rn VMRS Rd.P0	Rd -> result	MVE
mve_pred16_t [_arm_]vctp32q_m(uint32_t a, mve_pred16_t p)	a -> Rn p -> Rp	VMSR P0,Rp VPST VCTPT.32 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vctp64q_m(uint32_t a, mve_pred16_t p)	a -> Rn p -> Rp	VMSR P0,Rp VPST VCTPT.64 Rn VMRS Rd,P0	Rd -> result	MVE
int8x16_t [arm_]vuninitializedq_s8(void)			Qd -> result	MVE
int16x8_t [_arm_]vuninitializedq_s16(void)			Qd -> result	MVE
int32x4_t [_arm_]vuninitializedq_s32(void)	-		Qd -> result	MVE
int64x2_t [arm_]vuninitializedq_s64(void) uint8x16_t [arm_]vuninitializedq_u8(void)			Qd -> result Qd -> result	MVE MVE
uint16x8_t [arm_]vuninitializedq_u16(void)			Od -> result	MVE
uint32x4_t [arm_]vuninitializedq_u32(void)			Qd -> result	MVE
uint64x2_t [arm_]vuninitializedq_u64(void)			Qd -> result	MVE
float16x8_t [arm_]vuninitializedq_f16(void)			Qd -> result	MVE
float32x4_t [_arm_]vuninitializedq_f32(void) int8x16_t [_arm_]vuninitializedq(int8x16_t t)	t -> Do Not		Qd -> result Qd -> result	MVE MVE
	Evaluate			
int16x8_t [arm_]vuninitializedq(int16x8_t t)	t -> Do Not Evaluate		Qd -> result	MVE
int32x4_t [arm_]vuninitializedq(int32x4_t t)	t -> Do Not Evaluate		Qd -> result	MVE
int64x2_t [arm_]vuninitializedq(int64x2_t t)	t -> Do Not Evaluate		Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [arm_]vuninitializedq(uint8x16_t t)	t -> Do Not Evaluate		Qd -> result	MVE
uint16x8_t [arm_]vuninitializedq(uint16x8_t t)	t -> Do Not Evaluate		Qd -> result	MVE
uint32x4_t [arm_]vuninitializedq(uint32x4_t t)	t -> Do Not Evaluate		Qd -> result	MVE
uint64x2_t [arm_]vuninitializedq(uint64x2_t t)	t -> Do Not Evaluate		Qd -> result	MVE
float16x8_t [arm_]vuninitializedq(float16x8_t t)	t -> Do Not Evaluate		Qd -> result	MVE
float32x4_t [arm_]vuninitializedq(float32x4_t t)	t -> Do Not Evaluate		Qd -> result	MVE
int16x8_t [arm_]vreinterpretq_s16[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [_arm_]vreinterpretq_s32[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [_arm_]vreinterpretq_f32[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [arm_]vreinterpretq_s64[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [arm_]vreinterpretq_s8[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [arm_]vreinterpretq_s32[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [arm_]vreinterpretq_s64[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [arm_]vreinterpretq_s8[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [arm_]vreinterpretq_s16[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [arm_]vreinterpretq_s64[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [arm_]vreinterpretq_s8[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [arm_]vreinterpretq_s16[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [arm_]vreinterpretq_s32[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [arm_]vreinterpretq_s64[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [arm_]vreinterpretq_s8[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [arm_]vreinterpretq_s16[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [arm_]vreinterpretq_s32[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [arm_]vreinterpretq_s64[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [arm_]vreinterpretq_s8[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [_arm_]vreinterpretq_s16[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [arm_]vreinterpretq_s32[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vreinterpretq_u32[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_u16](uint16x8_t	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [_arm_]vreinterpretq_u64[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [_arm_]vreinterpretq_s8[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [_arm_]vreinterpretq_s16[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [_arm_]vreinterpretq_s32[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [_arm_]vreinterpretq_f32[_u32](uint32x4_t	a -> Od	NOP	Od -> result	MVE/NEON
a)				
uint8x16_t [arm_]vreinterpretq_u8[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_u32](uint32x4_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)				
uint64x2_t [arm_]vreinterpretq_u64[_u32](uint32x4_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)	0.1	WOR	0.1	MENTON
int64x2_t [arm_]vreinterpretq_s64[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [_arm_]vreinterpretq_f16[_u32](uint32x4_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a) int8x16_t [arm_]vreinterpretq_s8[_u64](uint64x2_t a)	a -> Od	NOP	Qd -> result	MVE/NEON
int16x8_t [arm_]vreinterpretq_s16[_u64](uint64x2_t a)	a -> Qd a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [arm_]vreinterpretq_s32[_u64](uint64x2_t a)	a -> Qd a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_u64](uint64x2_t	a -> Qd a -> Qd	NOP	Qd -> result	MVE/NEON
a)	a -> Qu	NOI	Qu => resuit	WIVE/INEON
uint8x16_t [arm_]vreinterpretq_u8[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_u64](uint64x2_t	a -> Qd	NOP	Od -> result	MVE/NEON
a)	2 . 4.		Qu , Iosun	
uint32x4_t [arm_]vreinterpretq_u32[_u64](uint64x2_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)	`			
int64x2_t [arm_]vreinterpretq_s64[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_u64](uint64x2_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)				
int8x16_t [arm_]vreinterpretq_s8[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [arm_]vreinterpretq_s16[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [arm_]vreinterpretq_s32[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [arm_]vreinterpretq_s8[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [_arm_]vreinterpretq_s16[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [_arm_]vreinterpretq_s32[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_f16](float16x8_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)	. 01	NOR	0.1 1:	MUENICON
uint8x16_t [_arm_]vreinterpretq_u8[_f16](float16x8_t a)	a -> Qd	NOP NOP	Qd -> result	MVE/NEON MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_f16](float16x8_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a) uint32x4_t [_arm_]vreinterpretq_u32[_f16](float16x8_t	a -> Qd	NOP	Od -> result	MVE/NEON
a)	a -> Qu	NOI	Qu => resuit	WIVE/INDOIN
uint64x2_t [arm_]vreinterpretq_u64[_f16](float16x8_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)			Ì	
int64x2_t [arm_]vreinterpretq_s64[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64_t [arm_]lsll(uint64_t value, int32_t shift)	value ->	LSLL RdaLo,RdaHi,Rm	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]		-> result	
1.64	shift -> Rm	AGDY DAY TO THE	m 1 *** * · · ·	) (T) (T)
int64_t [arm_]asrl(int64_t value, int32_t shift)	value ->	ASRL RdaLo,RdaHi,Rm	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo] shift -> Rm		-> result	
uint64_t [arm_]uqrshll(uint64_t value, int32_t shift)	value ->	UQRSHLL	[RdaHi,RdaLo]	MVE
umto4_t [arm_juqrsim(umto4_t value, mt32_t simt)	[RdaHi,RdaLo]	RdaLo,RdaHi,#64,Rm	-> result	IVI V IL
	shift -> Rm	RuaLo,Ruarii,#04,Riii	-> icsuit	
uint64 t [ arm ]uqrshll sat48(uint64 t value, int32 t	value ->	UQRSHLL	[RdaHi,RdaLo]	MVE
shift)	[RdaHi,RdaLo]	RdaLo,RdaHi,#48,Rm	-> result	
, ,	shift -> Rm	., ,,		
int64_t [arm_]sqrshrl(int64_t value, int32_t shift)	value ->	SQRSHRL	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]	RdaLo,RdaHi,#64,Rm	-> result	
	shift -> Rm			
int64_t [arm_]sqrshrl_sat48(int64_t value, int32_t shift)	value ->	SQRSHRL	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]	RdaLo,RdaHi,#48,Rm	-> result	
1.04.15	shift -> Rm	***************************************	m 1 *** * · · ·	) (T) (T)
uint64_t [arm_]uqshll(uint64_t value, const int shift)	value ->	UQSHLL RdaLo,RdaHi,#shift	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]		-> result	
uint64_t [_arm_]urshrl(uint64_t value, const int shift)	1 <= shift <= 32	IIDCUDI Daal a Daali #akita	[DdoH; DdoI c]	MVE
unito4_t [ariii_jursiiri(unito4_t value, const int shift)	value -> [RdaHi,RdaLo]	URSHRL RdaLo,RdaHi,#shift	[RdaHi,RdaLo] -> result	MVE
	1 <= shift <= 32		-/ ICSUIT	
int64_t [arm_]srshrl(int64_t value, const int shift)	value ->	SRSHRL RdaLo,RdaHi,#shift	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]		-> result	

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64_t [arm_]sqshll(int64_t value, const int shift)	value -> [RdaHi,RdaLo] 1 <= shift <= 32	SQSHLL RdaLo,RdaHi,#shift	[RdaHi,RdaLo] -> result	MVE
uint32_t [arm_]uqrshl(uint32_t value, int32_t shift)	value -> Rda shift -> Rm	UQRSHL Rda,Rm	Rda -> result	MVE
int32_t [arm_]sqrshr(int32_t value, int32_t shift)	value -> Rda shift -> Rm	SQRSHR Rda,Rm	Rda -> result	MVE
uint32_t [arm_]uqshl(uint32_t value, const int shift)	value -> Rda 1 <= shift <= 32	UQSHL Rda,#shift	Rda -> result	MVE
uint32_t [arm_]urshr(uint32_t value, const int shift)	value -> Rda 1 <= shift <= 32	URSHR Rda,#shift	Rda -> result	MVE
int32_t [arm_]sqshl(int32_t value, const int shift)	value -> Rda 1 <= shift <= 32	SQSHL Rda,#shift	Rda -> result	MVE
int32_t [arm_]srshr(int32_t value, const int shift)	value -> Rda 1 <= shift <= 32	SRSHR Rda,#shift	Rda -> result	MVE