

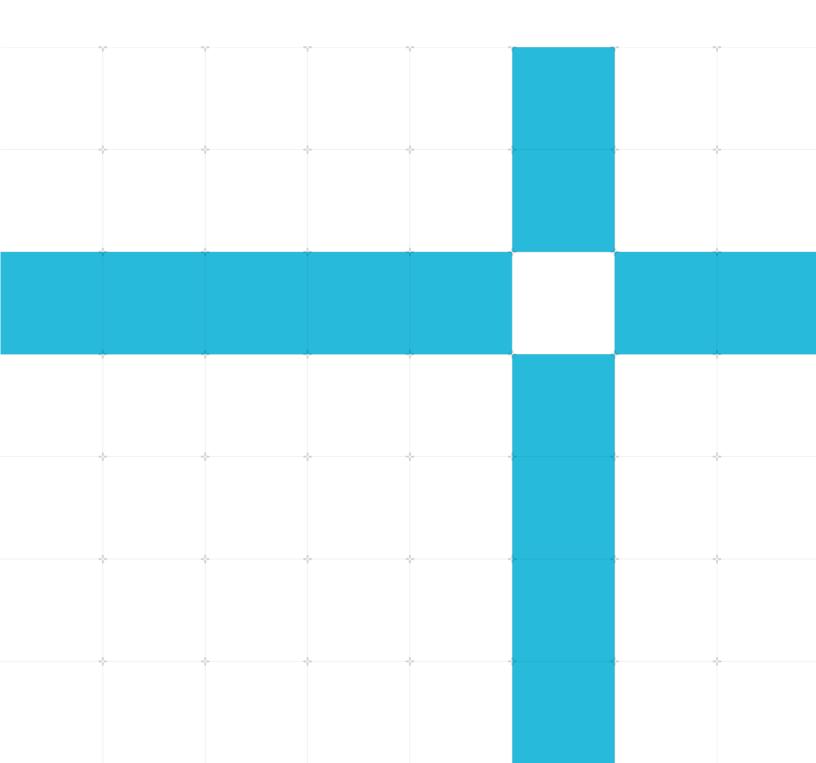
Arm MVE Intrinsics Reference for ACLE Q3 2019

Non-Confidential

Copyright $\ \ \, \ \ \,$ 2019 Arm Limited (or its affiliates). All rights reserved.

Issue Q319-00

101809



Arm MVE Intrinsics Reference 101809

Arm MVE Intrinsics

Reference

Copyright © 2019 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

Non-Confidential Proprietary Notice

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

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

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

This document may include technical inaccuracies or typographical errors.

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

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

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written

Arm MVE Intrinsics Reference 101809

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

Argument Preparation	Instruction	Result	Supported Architectures
a -> [Rt, Rt2] b -> [Rt3 Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3] Qd[1] Rt4 Rt2	Qd -> result	MVE
a -> [Rt, Rt2]	VMOV Qd[2],Qd[0],Rt3,Rt	Qd -> result	MVE
a -> [Rt, Rt2]	VMOV Qd[2],Qd[0],Rt3,Rt	Qd -> result	MVE
a -> [Rt, Rt2]	VMOV Qd[2],Qd[0],Rt3,Rt	Qd -> result	MVE
a -> [Rt, Rt2]	VMOV Qd[2],Qd[0],Rt3,Rt	Qd -> result	MVE
a -> [Rt, Rt2]	VMOV Qd[2],Qd[0],Rt3,Rt	Qd -> result	MVE
a -> [Rt, Rt2]	VMOV Qd[2],Qd[0],Rt3,Rt	Qd -> result	MVE
a -> [Rt, Rt2]	VMOV Qd[2],Qd[0],Rt3,Rt	Qd -> result	MVE
a -> [Rt, Rt2]	VMOV Qd[2],Qd[0],Rt3,Rt	Qd -> result	MVE
a -> [Rt, Rt2]	VMOV Qd[2],Qd[0],Rt3,Rt	Qd -> result	MVE
a -> Rn	VMOV Qd[3],Qd[1],Rt4,Rt2 VDDUP.U8 Qd,Rn,imm	Qd -> result	MVE
a -> Rn	VDDUP.U16 Qd,Rn,imm	Qd -> result	MVE
a -> Rn	VDDUP.U32 Qd,Rn,imm	Qd -> result	MVE
imm in [1,2,4,8] *a -> Rn	VDDUP.U8 Qd,Rn,imm	Qd -> result	MVE
imm in [1,2,4,8] *a -> Rn	VDDUP.U16 Qd,Rn,imm	Rn -> *a Qd -> result	MVE
imm in [1,2,4,8] *a -> Rn	VDDUP.U32 Qd,Rn,imm	Rn -> *a Qd -> result	MVE
imm in [1,2,4,8] inactive -> Qd	VMSR P0,Rp	Rn -> *a Qd -> result	MVE
a -> Rn imm in [1,2,4,8]	VPST VDDUPT.U8 Qd,Rn,imm		
p -> Rp inactive -> Od	VMSR P0,Rp	Od -> result	MVE
a -> Rn imm in [1,2,4,8]	VPST VDDUPT.U16 Qd,Rn,imm		
p -> Rp inactive -> Od	VMSR P0,Rp	Od -> result	MVE
a -> Rn imm in [1,2,4,8]	VPST VDDUPT.U32 Qd,Rn,imm		
p -> Rp	VMSR P0.Rp	Od -> result	MVE
*a -> Rn	VPST	Rn -> *a	
p -> Rp		Od -> result	MVE
*a -> Rn	VPST	Rn -> *a	
p -> Rp	2	Od -> result	MVE
*a -> Rn	VPST	Rn -> *a	11112
p -> Rp		Od -> result	MVE
imm in [1,2,4,8]	VPST	Qu' y Tesan	11112
a -> Rn	VMSR P0,Rp	Qd -> result	MVE
p -> Rp	VDDUPT.U16 Qd,Rn,imm	Od -> result	MVE
imm in [1,2,4,8]	VPST	Qu > resun	11112
*a -> Rn	VMSR P0,Rp	Qd -> result	MVE
p -> Rp	VDDUPT.U8 Qd,Rn,imm		MVE
*a -> Kn imm in [1,2,4,8]	VMSR PO,Rp VPST	Rn -> *a	IVI V E
	Preparation a -> [Rt, Rt2] b -> [Rt3, Rt4] a -> Rn	Preparation a -> [Rt, Rt2] b -> [Rt3, Rt4] VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2 a -> [Rt, Rt2] VMOV Qd[3],Qd[1],Rt4,Rt2 b -> [Rt3, Rt4] VMOV Qd[3],Qd[1],Rt4,Rt2 b -> [Rt3, Rt4] VMOV Qd[3],Qd[1],Rt4,Rt2 b -> [Rt3, Rt4] VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[2]	Preparation a > [Rt, Rt2] b > [Rt3, Rt4] vMOV Qd[3],Qd[1],Rt4,Rt2 b > [Rt3, Rt4] vMOV Qd[3],Qd[1],Rt4,Rt2 b > [Rt3, Rt4] vMOV Qd[3],Qd[1],Rt4,Rt2 a > [Rt, Rt2] vMOV Qd[3],Qd[1],Rt4,Rt2 vMOV Qd[3],Qd[3],Rt4,Rt2 vMOV

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
uint8x16_t [_arm_]vdwdupq[_n]_u8(uint32_t a, uint32_t b, const int imm)	p -> Rp a -> Rn b -> Rm imm in [1,2,4,8]	VDDUPT.U32 Qd,Rn,imm VDWDUP.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
uint16x8_t [_arm_]vdwdupq[_n]_u16(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U16 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32x4_t [_arm_]vdwdupq[_n]_u32(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint8x16_t [_arm_]vdwdupq[_wb]_u8(uint32_t * a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U8 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [_arm_]vdwdupq[_wb]_u16(uint32_t * a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U16 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]vdwdupq[_wb]_u32(uint32_t * a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U32 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [_arm_]vdwdupq_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]	VMSR P0,Rp VPST VDWDUPT.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
uint16x8_t [arm_]vdwdupq_m[_n_u16](uint16x8_t inactive, uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32x4_t [_arm_]vdwdupq_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 VDWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint8x16_t [_arm_]vdwdupq_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 VDWDUPT.U8 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	inactive -> Qd *a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]vdwdupq_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 VDWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [_arm_]vdwdupq_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 VDWDUPT.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
uint16x8_t [_arm_]vdwdupq_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 VDWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32x4_t [_arm_]vdwdupq_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 VDWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint8x16_t [_arm_]vdwdupq_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 VDWDUPT.U8 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	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 imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]vdwdupq_x[_wb]_u32(uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	*a -> Rp *a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	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 imm)	a -> Rn imm in [1,2,4,8]	VIDUP.U8 Qd,Rn,imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [_arm_]vidupq[_n]_u16(uint32_t a, const int imm)	a -> Rn imm in [1,2,4,8]	VIDUP.U16 Qd,Rn,imm	Qd -> result	MVE
uint32x4_t [_arm_]vidupq[_n]_u32(uint32_t a, const int imm)	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] p -> Rp	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)	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
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
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
uint32x4_t [_arm_]vdupq_m[_n_u32](uint32x4_t inactive, uint32_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.32 Qd,Rt	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [arm_]vdupq_m[_n_f16](float16x8_t inactive, float16_t a, mve_pred16_t p)	inactive -> Qd a -> Rt	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, noutro_t a, mve_predro_t p)	p -> Rp	VDUPT.16 Qd,Rt		
float32x4_t [_arm_]vdupq_m[_n_f32](float32x4_t inactive, float32_t a, mve_pred16_t p)	inactive -> Qd a -> Rt	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VDUPT.32 Qd,Rt		
int8x16_t [arm_]vdupq_x_n_s8(int8_t a, mve_pred16_t	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		
int32x4_t [arm_]vdupq_x_n_s32(int32_t a,	a -> Rt	VDUPT.16 Qd,Rt VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu y resum	111,2
		VDUPT.32 Qd,Rt		
uint8x16_t [arm_]vdupq_x_n_u8(uint8_t a, mve_pred16_t p)	a -> Rt p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
nive_picaro_t p)	p -> Kp	VDUPT.8 Qd,Rt		
uint16x8_t [arm_]vdupq_x_n_u16(uint16_t a,	a -> Rt	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
uint32x4 t[arm]vdupq x n u32(uint32 t a,	a -> Rt	VDUPT.16 Qd,Rt VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu -> resuit	MVE
		VDUPT.32 Qd,Rt		
float16x8_t [arm_]vdupq_x_n_f16(float16_t a,	a -> Rt	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VDUPT.16 Qd,Rt		
float32x4_t [arm_]vdupq_x_n_f32(float32_t a,	a -> Rt	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
	a -> On	VDUPT.32 Qd,Rt	D.114	MVE
mve_pred16_t [arm_]vcmpeqq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMP.F16 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq[_f32](float32x4_t a,	a -> Qn	VCMP.F32 eq,Qn,Qm	Rd -> result	MVE
float32x4_t b)	b -> Qm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VCMP.I8 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq[_s16](int16x8_t a,	a -> Qn	VCMP.I16 eq,Qn,Qm	Rd -> result	MVE
int16x8_t b)	b -> Qm	VMRS Rd,P0		
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		
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	Ru -> resuit	WIVE
mve_pred16_t [arm_]vcmpeqq[_n_f16](float16x8_t a,	a -> Qn	VCMP.F16 eq,Qn,Rm	Rd -> result	MVE
float16_t b)	b -> Rm	VMRS Rd,P0 VCMP.F32 eq,Qn,Rm	D1 . 1	MVE
mve_pred16_t [arm_]vcmpeqq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VCMP.F32 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq[_n_s8](int8x16_t a,	a -> Qn	VCMP.I8 eq,Qn,Rm	Rd -> result	MVE
int8_t b)	b -> Rm	VMRS Rd,P0		Nam.
mve_pred16_t [arm_]vcmpeqq[_n_s16](int16x8_t a, int16 t b)	a -> Qn b -> Rm	VCMP.I16 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq[_n_s32](int32x4_t a,	a -> Qn	VCMP.I32 eq,Qn,Rm	Rd -> result	MVE
int32_t b)	b -> Rm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq[_n_u8](uint8x16_t a,	a -> Qn b -> Rm	VCMP.I8 eq,Qn,Rm	Rd -> result	MVE
uint8_t b) mve_pred16_t [arm_]vcmpeqq[_n_u16](uint16x8_t a,	a -> Qn	VMRS Rd,P0 VCMP.I16 eq,Qn,Rm	Rd -> result	MVE
uint16_t b)	b->Rm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq[_n_u32](uint32x4_t a,	a -> Qn	VCMP.I32 eq,Qn,Rm	Rd -> result	MVE
uint32_t b) mve_pred16_t [arm_]vcmpeqq_m[_f16](float16x8_t a,	b -> Rm a -> On	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST	ita > icauit	
	p -> Rp	VCMPT.F16 eq,Qn,Qm		
mve_pred16_t [arm_]vcmpeqq_m[_f32](float32x4_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR PU,RP VPST	Ku -> resuit	1V1 V IL
_ , _ı _ ı ,	p -> Rp	VCMPT.F32 eq,Qn,Qm		
mayo madd6 4 [0.50	VMRS Rd,P0	D.4	MYE
mve_pred16_t [arm_]vcmpeqq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rd -> result	MVE
	p -> Rp	VCMPT.I8 eq,Qn,Qm		
		VMRS Rd,P0		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [_arm_]vcmpeqq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I16 eq,Qn,Qm	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMRS Rd,P0 VMSR P0,Rp VPST VCMPT.I32 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpeqq_m[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I8 eq.Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq_m[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR Rd,10 VMSR PO,Rp VPST VCMPT.I16 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I32 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpeqq_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 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq_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 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq_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 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq_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 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpeqq_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 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpeqq_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 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq_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 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq_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 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMP.F16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMP.F32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VCMP.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VCMP.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VCMP.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VCMP.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VCMP.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VCMP.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMRS RO,Rp VPST VCMPT.F32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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.18 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)	a -> Qn b -> Rm	VCMP.F16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VCMP.F32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VCMP.I8 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VCMP.I16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	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	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	VMRS Rd,P0 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
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 -> Qm	VCMP.F32 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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 -> Om	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 -> Om	VMSR P0,Rp VPST	Rd -> result	MVE
moatroxo_t b, mve_predio_t p)	p -> Rp	VCMPT.F16 ge,Qn,Qm VMRS Rd,P0		
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)	a -> Qn b -> Rm p -> Rp	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
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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	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	VMRS Rd,P0 VMSR P0,Rp VPST VCMPT.S16 gt,Qn,Qm	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	VMRS Rd,P0 VMSR P0,Rp VPST VCMPT.S32 gt,Qn,Qm	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VMRS Rd,P0 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 VCMP.S32 gt,Qn,Rm	Rd -> result	MVE MVE
mve_pred16_t [arm_]vcmpgtq[_n_s32](int32x4_t a, int32_t b) mve_pred16_t [arm_]vcmpgtq_m[_n_f16](float16x8_t	a -> Qn b -> Rm a -> Qn	VCMP.S32 gt,Qn,Rm VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
a, float16_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VCMPT.F16 gt,Qn,Rm VMRS Rd,P0	Ku -> result	WIVE
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	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 Rd, F0 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
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
$\label{lem:mve_pred16_t [_arm_]vcmpleq[_n_f32](float32x4_t a, float32_t b)} mve_pred16_t [_arm_]vcmpleq[_n_f32](float32x4_t a, float32_t b)$	a -> Qn b -> Rm	VCMP.F32 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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 PO,Rp VPST VCMPT.F16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
mve_pred16_t [arm_]vcmphiq_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 hi,Qn,Rm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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)	$\begin{array}{l} b -> Rm \\ p -> Rp \end{array}$	VPST VCMPT.U16 hi,Qn,Rm		
myyo mod 16 4 f omy lyomahia mf n y221/yint22y/4 4	0.500	VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq_m[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Rd -> result	MVE
a, ambt.o, mre_predio_t.p/	p -> Rp	VCMPT.U32 hi,Qn,Rm VMRS Rd,P0		
int8x16_t [arm_]vminq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMIN.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vminq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMIN.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vminq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMIN.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vminq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMIN.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vminq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMIN.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vminq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMIN.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vminq_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 VMINT.S8 Qd,Qn,Qm		
int16x8_t [arm_]vminq_m[_s16](int16x8_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST	Q	
	b -> Qm p -> Rp	VMINT.S16 Qd,Qn,Qm		
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 -> Qm p -> Rp	VPST VMINT.S32 Qd,Qn,Qm		
uint8x16_t [arm_]vminq_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> 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 b -> Qm	VPST VMINT.U16 Qd,Qn,Qm		
	p -> QIII	VIVIIIV1.010 Qu,Qii,Qiii		
uint32x4_t [arm_]vminq_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	VPST VMINT.U32 Qd,Qn,Qm		
	p -> Qm	VIVIIN1.032 Qu,Qii,Qiii		
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		
int16x8_t [arm_]vminq_x[_s16](int16x8_t a, int16x8_t	p -> Rp a -> Qn	VMINT.S8 Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST	Ç	
	p -> Rp	VMINT.S16 Qd,Qn,Qm		1.00
int32x4_t [arm_]vminq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
b, mvc_prearo_t p)	p -> Rp	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 p -> Rp	VPST VMINT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vminq_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_]vminq_x[_u32](uint32x4_t a,	p -> Rp a -> Qn	VMINT.U16 Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
uint32x4_t [arm_]vininq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	b -> Qm	VMSK PO,KP VPST	Qu -> resuit	IVI V IS
	p -> Rp	VMINT.U32 Qd,Qn,Qm	_	
uint8x16_t [arm_]vminaq[_s8](uint8x16_t a, int8x16_t b)	a -> Qda b -> Qm	VMINA.S8 Qda,Qm	Qda -> result	MVE
uint16x8_t [arm_]vminaq[_s16](uint16x8_t a, int16x8_t b)	a -> Qda b -> Qm	VMINA.S16 Qda,Qm	Qda -> result	MVE
uint32x4_t [_arm_]vminaq[_s32](uint32x4_t a, int32x4_t b)	a -> Qda b -> Qm	VMINA.S32 Qda,Qm	Qda -> result	MVE
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 Se Ode Om		
uint16x8_t [arm_]vminaq_m[_s16](uint16x8_t a,	p -> Rp a -> Qda	VMINAT.S8 Qda,Qm VMSR P0,Rp	Qda -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST	100000	\
	p -> Rp	VMINAT.S16 Qda,Qm	01	MVE
1 22 4 4 5 1 1 5 227 1 22 4 5				
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8_t [arm_]vminvq[_s8](int8_t a, int8x16_t b)	a -> Rda b -> Qm	VMINV.S8 Rda,Qm	Rda -> result	MVE
int16_t [arm_]vminvq[_s16](int16_t a, int16x8_t b)	a -> Rda b -> Qm	VMINV.S16 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vminvq[_s32](int32_t a, int32x4_t b)	a -> Rda b -> Qm	VMINV.S32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vminvq[_u8](uint8_t a, uint8x16_t b)	a -> Rda	VMINV.U8 Rda,Qm	Rda -> result	MVE
uint16_t [arm_]vminvq[_u16](uint16_t a, uint16x8_t b)	b -> Qm 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
int8_t [arm_]vminvq_p[_s8](int8_t a, int8x16_t b, mve_pred16_t p)	b -> Qm a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
	p -> Rp	VMINVT.S8 Rda,Qm VMSR P0,Rp	D.116	MVE
int16_t [_arm_]vminvq_p[_s16](int16_t a, int16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSK PO,Rp VPST VMINVT.S16 Rda,Qm	Rda -> result	MVE
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 p -> Rp	VPST VMINVT.S32 Rda,Qm		
uint8_t [_arm_]vminvq_p[_u8](uint8_t a, uint8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
uint16_t [arm_]vminvq_p[_u16](uint16_t a, uint16x8_t	p -> Rp a -> Rda	VMINVT.U8 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINVT.U16 Rda,Qm		
uint32_t [_arm_]vminvq_p[_u32](uint32_t a, uint32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
uint8_t [_arm_]vminavq[_s8](uint8_t a, int8x16_t b)	p -> Rp a -> Rda	VMINVT.U32 Rda,Qm VMINAV.S8 Rda,Qm	Rda -> result	MVE
uint16_t [_arm_]vminavq[_s16](uint16_t a, int16x8_t b)	b -> Qm a -> Rda	VMINAV.S16 Rda,Qm	Rda -> result	MVE
uint32 t [_arm_]vminavq[_s32](uint32 t a, int32x4_t b)	b -> Qm a -> Rda	VMINAV.S32 Rda,Om	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 p -> Rp	VMSK FO,RP VPST VMINAVT.S8 Rda,Qm	Kua -> resuit	MIVE
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,Qm		
uint32_t [arm_]vminavq_p[_s32](uint32_t a, int32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
float16x8_t [arm_]vminnmq[_f16](float16x8_t a,	p -> Rp a -> Qn	VMINAVT.S32 Rda,Qm 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	Qd -> result	MVE/NEON
float32x4_t b) float16x8_t [_arm_]vminnmq_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 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 b -> Om	VPST VMINNMT.F32 Qd,Qn,Qm		
float16x8_t [_arm_]vminnmq_x[_f16](float16x8_t a,	p -> Rp a -> Qn	VMSR P0,Rp	Od -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINNMT.F16 Qd,Qn,Qm	Zu > Icoun	
float32x4_t [_arm_]vminnmq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
float16x8_t [_arm_]vminnmaq[_f16](float16x8_t a,	p -> Rp a -> Oda	VMINNMT.F32 Qd,Qn,Qm VMINNMA.F16 Qda,Qm	Oda -> result	MVE
float16x8_t b)	b->Qm			
float32x4_t [_arm_]vminnmaq[_f32](float32x4_t a, float32x4_t b)	a -> Qda b -> Qm	VMINNMA.F32 Qda,Qm	Qda -> result	MVE
float16x8_t [_arm_]vminnmaq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qda b -> Qm	VMSR P0,Rp VPST	Qda -> result	MVE
float32x4_t [_arm_]vminnmaq_m[_f32](float32x4_t a,	p -> Rp a -> Qda	VMINNMAT.F16 Qda,Qm 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, float16x8_t b)	a -> Rda b -> Qm	VMINNMV.F16 Rda,Qm	Rda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32_t [_arm_]vminnmvq[_f32](float32_t a, float32x4_t b)	a -> Rda b -> Om	VMINNMV.F32 Rda,Qm	Rda -> result	MVE
float16_t [_arm_]vminnmvq_p[_f16](float16_t a, float16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
float32_t [_arm_]vminnmvq_p[_f32](float32_t a, float32x4_t b, mve_pred16_t p)	p -> Rp a -> Rda b -> Qm	VMINNMVT.F16 Rda,Qm VMSR P0,Rp VPST	Rda -> result	MVE
float16_t [arm_]vminnmavq[_f16](float16_t a, float16x8_t b)	p -> Rp a -> Rda b -> Qm	VMINNMVT.F32 Rda,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, float16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINNMAVT.F16 Rda,Qm	Rda -> result	MVE
float32_t [arm_]vminnmavq_p[_f32](float32_t a, float32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINNMAVT.F32 Rda,Qm	Rda -> result	MVE
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 -> Qm	VMAX.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vmaxq_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 VMAXT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vmaxq_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 VMAXT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vmaxq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VMAXT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [arm_]vmaxq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmaxq_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 VMAXT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmaxq_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 VMAXT.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vmaxq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vmaxq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vmaxq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vmaxq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmaxq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmaxq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXT.U32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vmaxaq[_s8](uint8x16_t a, int8x16_t b)	a -> Qda b -> Qm	VMAX1.032 Qd,Qli,Qlii 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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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, int8x16_t b, mve_pred16_t p)	a -> Qda b -> Qm	VMSR P0,Rp VPST VMAXAT.S8 Qda,Qm	Qda -> result	MVE
uint16x8_t [arm_]vmaxaq_m[_s16](uint16x8_t a, int16x8_t b, mve_pred16_t p)	p -> Rp a -> Qda b -> Qm	VMSR P0,Rp VPST	Qda -> result	MVE
uint32x4_t [_arm_]vmaxaq_m[_s32](uint32x4_t a, int32x4_t b, mve_pred16_t p)	p -> Rp a -> Qda b -> Qm	VMAXAT.S16 Qda,Qm VMSR P0,Rp VPST	Qda -> result	MVE
int8_t [arm_]vmaxvq[_s8](int8_t a, int8x16_t b)	p -> Rp a -> Rda b -> Qm	VMAXAT.S32 Qda,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 b -> Qm	VMAXV.S32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vmaxvq[_u8](uint8_t a, uint8x16_t b)	a -> Rda b -> Qm	VMAXV.U8 Rda,Qm	Rda -> result	MVE
uint16_t [arm_]vmaxvq[_u16](uint16_t a, uint16x8_t b)	a -> Rda b -> Qm	VMAXV.U16 Rda,Qm	Rda -> result	MVE
uint32_t [_arm_]vmaxvq[_u32](uint32_t a, uint32x4_t b)	a -> Rda b -> Om	VMAXV.U32 Rda,Qm	Rda -> result	MVE
int8_t [_arm_]vmaxvq_p[_s8](int8_t a, int8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.S8 Rda,Qm	Rda -> result	MVE
int16_t [_arm_]vmaxvq_p[_s16](int16_t a, int16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.S16 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vmaxvq_p[_s32](int32_t a, int32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.S32 Rda,Qm	Rda -> result	MVE
uint8_t [_arm_]vmaxvq_p[_u8](uint8_t a, uint8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.U8 Rda,Qm	Rda -> result	MVE
uint16_t [_arm_]vmaxvq_p[_u16](uint16_t a, uint16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.U16 Rda,Qm	Rda -> result	MVE
uint32_t [_arm_]vmaxvq_p[_u32](uint32_t a, uint32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.U32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vmaxavq[_s8](uint8_t a, int8x16_t b)	a -> Rda b -> Qm	VMAXAV.S8 Rda,Qm	Rda -> result	MVE
uint16_t [arm_]vmaxavq[_s16](uint16_t a, int16x8_t b)	a -> Rda b -> Qm	VMAXAV.S16 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vmaxavq[_s32](uint32_t a, int32x4_t b)	a -> Rda b -> Qm	VMAXAV.S32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vmaxavq_p[_s8](uint8_t a, int8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXAVT.S8 Rda,Qm	Rda -> result	MVE
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 b, mve_pred16_t p)	p -> Rp a -> Rda b -> Qm p -> Rp	VMAXAVT.S16 Rda,Qm VMSR P0,Rp VPST VMAXAVT.S32 Rda,Qm	Rda -> result	MVE
float16x8_t [_arm_]vmaxnmq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VMAXNM.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]vmaxnmq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VMAXNM.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [_arm_]vmaxnmq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VMAXNMT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vmaxnmq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vmaxnmq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vmaxnmq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vmaxnmaq[_f16](float16x8_t a, float16x8_t b)	a -> Qda b -> Qm	VMAXNMA.F16 Qda,Qm	Qda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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, float16x8_t b, mve_pred16_t p)	a -> Qda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMAT.F16 Qda,Qm	Qda -> result	MVE
float32x4_t [_arm_]vmaxnmaq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMAT.F32 Qda,Qm	Qda -> result	MVE
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, float32x4_t b)	a -> Rda b -> Qm	VMAXNMV.F32 Rda,Qm	Rda -> result	MVE
float16_t [arm_]vmaxnmvq_p[_f16](float16_t a, float16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMVT.F16 Rda,Qm	Rda -> result	MVE
float32_t [_arm_]vmaxnmvq_p[_f32](float32_t a, float32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMVT.F32 Rda,Qm	Rda -> result	MVE
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, float16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMAVT.F16 Rda,Qm	Rda -> result	MVE
float32_t [arm_]vmaxnmavq_p[_f32](float32_t a, float32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMAVT.F32 Rda,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq[_s8](uint32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn c -> Qm	VABAV.S8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq[_s16](uint32_t a, int16x8_t b, int16x8_t c)	a -> Rda b -> Qn c -> Qm	VABAV.S16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq[_s32](uint32_t a, int32x4_t b, int32x4_t c)	a -> Rda b -> Qn c -> Qm	VABAV.S32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq[_u8](uint32_t a, uint8x16_t b, uint8x16_t c)	a -> Rda b -> Qn c -> Qm	VABAV.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq[_u16](uint32_t a, uint16x8_t b, uint16x8_t c)	a -> Rda b -> Qn c -> Qm	VABAV.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq[_u32](uint32_t a, uint32x4_t b, uint32x4_t c)	a -> Rda b -> Qn c -> Qm	VABAV.U32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq_p[_s8](uint32_t a, int8x16_t b, int8x16_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VABAVT.S8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq_p[_s16](uint32_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VABAVT.S16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq_p[_s32](uint32_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VABAVT.S32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq_p[_u8](uint32_t a, uint8x16_t b, uint8x16_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VABAVT.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq_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 VABAVT.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vabavq_p[_u32](uint32_t a, uint32x4_t b, uint32x4_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VABAVT.U32 Rda,Qn,Qm	Rda -> result	MVE
int8x16_t [arm_]vabdq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VABD.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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 -> Om	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)	a -> Qm	VABS.F16 Qd,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]vabsq[_f32](float32x4_t a)	a -> Qm	VABS.F32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vabsq[_s8](int8x16_t a) int16x8_t [arm_]vabsq[_s16](int16x8_t a)	a -> Qm a -> Qm	VABS.S8 Qd,Qm VABS.S16 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
int32x4_t [_arm_]vabsq[_s10](int32x4_t a)	a -> Qm a -> Qm	VABS.S36 Qd,Qm VABS.S32 Qd,Qm	Qd -> result	MVE/NEON MVE/NEON
float16x8_t [arm_]vabsq_m[_f16](float16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VABST.F16 Qd,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [_arm_]vabsq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int8x16_t [arm_]vabsq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm p -> Rp	VABST.F32 Qd,Qm 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 Od,Om	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 Od,Om	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_nzevqc 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.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29	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	AND Rt,#1 VMSR P0,Rp VPST VADCIT.I32 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_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
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.132 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_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
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.18 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,Qfl,Qfff VMSR P0,Rp VPST VADDT.116 Qd,Qfl,Qff	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		
1 100 4 17 1 1 1 1 1001/1 100 4 11 11	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	Qu -> result	WIVE
mve_predio_t p)	P - AP	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	,	
		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, int8x16_t a, mve_pred16_t p)	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
mtox16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VCLZT.I8 Qd,Qm		
int16x8_t [arm_]vclzq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int16x8_t a, mve_pred16_t p)	a -> Om	VPST	Qu -> result	WYL
mirono_t a, mve_prearo_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 a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCLZT.I32 Qd,Qm		
uint8x16_t [arm_]vclzq_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	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		
uint32x4_t [arm_]vclzq_m[_u32](uint32x4_t inactive,	p -> Rp inactive -> Qd	VCLZT.I16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
uint32x4_t [arm]vci2q_m[_u32](uint32x4_t mactive, uint32x4_t a, mve_pred16_t p)	a -> Qm	VMSK PU,KP VPST	Qu -> resuit	NIVE
umi32x4_t a, mvc_pred10_t p)	p -> Rp	VCLZT.I32 Qd,Qm		
int8x16_t [arm_]vclzq_x[_s8](int8x16_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu' > Tesun	
_1 _ 1/	1 1	VCLZT.I8 Qd,Qm		
int16x8_t [arm_]vclzq_x[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		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		
uint8x16 t [arm]vclzq x[u8](uint8x16 t a,	0	VCLZT.I32 Qd,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	a -> Qm p -> Rp	VMSK PO,KP VPST	Qu -> result	NIVE
mve_pred10_t p)	p -> Kp	VCLZT.I8 Qd,Qm		
uint16x8_t [arm_]vclzq_x[_u16](uint16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Q	1
		VCLZT.I16 Qd,Qm		
uint32x4_t [arm_]vclzq_x[_u32](uint32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	-	1
		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		1
float32x4_t [arm_]vnegq_m[_f32](float32x4_t inactive,	p -> Rp inactive -> Qd	VNEGT.F16 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
float32x4_t [arm_]vnegq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VMSR P0,Rp VPST	Qu -> resuit	IVI V E
noato2x=_t a, nive_picuro_t p)	p -> Rp	VNEGT.F32 Qd,Qm		1
int8x16_t [arm_]vnegq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
	-		20 - 105uit	1
int8x16_t a, mve_pred16_t p)	a -> Qm	VPST		

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	VPST		
	p -> Rp	VNEGT.S16 Qd,Qm	0.1 1:	MATE
int32x4_t [_arm_]vnegq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mi32x4_t a, mve_pred10_t p)	p -> Rp	VNEGT.S32 Qd,Qm		
float16x8_t [arm_]vnegq_x[_f16](float16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VNEGT.F16 Qd,Qm		
float32x4_t [arm_]vnegq_x[_f32](float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VNEGT.F32 Qd,Qm		
int8x16_t [arm_]vnegq_x[_s8](int8x16_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu's resun	1.1 1 2
-1 - 1		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		
int22v4 t [a > Om	VNEGT.S16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
int32x4_t [_arm_]vnegq_x[_s32](int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSK PO,KP VPST	Qu -> resuit	NIVE
mve_pred10_t p)	p > Kp	VNEGT.S32 Qd,Qm		
int8x16_t [arm_]vmulhq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
	b->Qm			
int16x8_t [arm_]vmulhq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VMULH.S16 Qd,Qn,Qm	Qd -> result	MVE
· (22 4 · f	b -> Qm	VD 411 11 C22 O 1 O O	0.1 1:	MATE
int32x4_t [_arm_]vmulhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMULH.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16 t [arm]vmulhq[u8](uint8x16 t a, uint8x16 t	a -> On	VMULH.U8 Qd,Qn,Qm	Od -> result	MVE
b)	b -> Qm	, mezmes qu,qu,qu	Qu'y resuit	
uint16x8_t [arm_]vmulhq[_u16](uint16x8_t a,	a -> Qn	VMULH.U16 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t b)	b->Qm			
uint32x4_t [arm_]vmulhq[_u32](uint32x4_t a,	a -> Qn	VMULH.U32 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t b) int8x16_t [_arm_]vmulhq_m[_s8](int8x16_t inactive,	b -> Qm inactive -> Qd	VMCD DO Do	Od -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qu	VMSR P0,Rp VPST	Qu -> resuit	MVE
mtox10_t a, mtox10_t o, mvc_pred10_t p)	b -> Om	VMULHT.S8 Qd,Qn,Qm		
	p -> Rp			
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	VPST		
	b -> Qm p -> Rp	VMULHT.S16 Qd,Qn,Qm		
int32x4_t [arm_]vmulhq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn	VPST	Qu > resurt	W. C.
_ · · _ · _ · _ •	b -> Qm	VMULHT.S32 Qd,Qn,Qm		
	p -> Rp			
uint8x16_t [arm_]vmulhq_m[_u8](uint8x16_t inactive,	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
uint8x16_t a, uint8x16_t b, mve_pred16_t p)	b -> Qm	VMULHT.U8 Qd,Qn,Qm		
	p -> Rp	VIVICEITI.CO Qu,Qii,Qiii		
uint16x8_t [arm_]vmulhq_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	VMULHT.U16 Qd,Qn,Qm		
vint22v4 t.f. ama lymytha mf v22l/vint22v4 t inactiva	p -> Rp inactive -> Qd	VMCD DO Do	Od > mooult	MVE
uint32x4_t [arm_]vmulhq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> On	VMSR P0,Rp VPST	Qd -> result	NIVE
umt52x4_t a, umt52x4_t b, mvc_preuto_t p)	b -> Qm	VMULHT.U32 Qd,Qn,Qm		
	p -> Rp	. , , , ,		
int8x16_t [arm_]vmulhq_x[_s8](int8x16_t a, int8x16_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
int16x8 t [arm]vmulhq x[s16](int16x8 t a, int16x8 t	p -> Rp a -> Qn	VMULHT.S8 Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VMSK FO,KP VPST	Qu -> resuit	IVI V IL
c, mre_predic_t p)	p -> Rp	VMULHT.S16 Qd,Qn,Qm		
int32x4_t [arm_]vmulhq_x[_s32](int32x4_t a, int32x4_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
nint0v16 4 f	p -> Rp	VMULHT.S32 Qd,Qn,Qm	04 5 19	MATE
uint8x16_t [arm_]vmulhq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
amonto_c o, mito_prodio_t p)	p -> QIII	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	VPST		
	p -> Rp	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 VMULHT U32 Od On Om		
	p -> Rp	VMULHT.U32 Qd,Qn,Qm	1	
uint16x8_t [arm_]vmullbq_poly[_p8](uint8x16_t a,	a -> Qn	VMULLB.P8 Qd,Qn,Qm	Qd -> result	MVE

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,	a -> Qn	VMULLB.U32 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t b) uint16x8_t [arm_]vmullbq_poly_m[_p8](uint16x8_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	b -> Qm inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qm p -> Rp	VMULLBT.P8 Qd,Qn,Qm		
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	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)	p -> Rp inactive -> Qd a -> Qn b -> Qm	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)	p -> Rp inactive -> Qd a -> Qn b -> Qm	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)	p -> Rp 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	VPST		
	b -> Rm p -> Rp	VMULT.I8 Qd,Qn,Rm		
uint16x8_t [arm_]vmulq_m[_n_u16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn	VPST	(a : 222	
	b -> Rm	VMULT.I16 Qd,Qn,Rm		
	p -> Rp	VA (CD, DO D	0.1	NOTE
uint32x4_t [_arm_]vmulq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
macuve, umiozxvi_t a, umioz_t o, mve_preuro_t p)	b -> Rm	VMULT.I32 Qd,Qn,Rm		
	p -> Rp			
float16x8_t [arm_]vmulq_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_]vmulq_x[_f32](float32x4_t a,	p -> Rp a -> On	VMULT.F16 Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
float32x4_t tarm_jviidiq_xt_152j(float52x4_t a, float32x4_t b, mve_pred16_t p)	b -> Om	VPST	Qu -> resurt	WYL
	p -> Rp	VMULT.F32 Qd,Qn,Qm		
$float16x8_t \ [__arm_]vmulq_x[_n_f16](float16x8_t \ a,$	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float16_t b, mve_pred16_t p)	b -> Rm	VPST		
float32x4_t [arm_]vmulq_x[_n_f32](float32x4_t a,	p -> Rp a -> Qn	VMULT.F16 Qd,Qn,Rm VMSR P0,Rp	Qd -> result	MVE
float32_t b, mve_pred16_t p)	b -> Rm	VPST	Qu > resuit	
_ · · - • · · · · · · · · · · · · · · · ·	p -> Rp	VMULT.F32 Qd,Qn,Rm		
int8x16_t [arm_]vmulq_x[_s8](int8x16_t a, int8x16_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Qm	VPST		
int16x8_t [arm_]vmulq_x[_s16](int16x8_t a, int16x8_t	p -> Rp a -> On	VMULT.I8 Qd,Qn,Qm VMSR P0.Rp	Od -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST	Qu -> resurt	WYL
	p -> Rp	VMULT.I16 Qd,Qn,Qm		
int32x4_t [arm_]vmulq_x[_s32](int32x4_t a, int32x4_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
int8x16_t [arm_]vmulq_x[_n_s8](int8x16_t a, int8_t b,	p -> Rp a -> On	VMULT.I32 Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	b -> Rm	VPST	Qu -> resurt	WYL
	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	VPST		
int32x4_t [arm_]vmulq_x[_n_s32](int32x4_t a, int32_t	p -> Rp a -> On	VMULT.I16 Qd,Qn,Rm VMSR P0,Rp	Od -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST	Qu -> resurt	WYL
· -• -•	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 -> Qn	VMSR P0,Rp	Od -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Q	
	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 p -> Rp	VPST VMULT.I32 Qd,Qn,Qm		
uint8x16_t [arm_]vmulq_x[_n_u8](uint8x16_t a, uint8_t	a -> Qn	VMSR P0,Rp	Od -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST		
The state of the s	p -> Rp	VMULT.I8 Qd,Qn,Rm		100
uint16x8_t [_arm_]vmulq_x[_n_u16](uint16x8_t a, uint16 t b, mve_pred16 t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Qd -> result	MVE
umito_t o, mve_preuto_t p)	b -> Rm p -> Rp	VMULT.I16 Qd,Qn,Rm		
uint32x4_t [arm_]vmulq_x[_n_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, mve_pred16_t p)	b -> Rm	VPST	-	
1.00 (1.1)	p -> Rp	VMULT.I32 Qd,Qn,Rm	01. *	NOTE
int32x4_t [_arm_]vsbciq[_s32](int32x4_t a, int32x4_t b, unsigned * carry_out)	a -> Qn b -> Qm	VSBCI.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc	Qd -> result Rt ->	MVE
ansigned carry_out/	2 -> QIII	LSR Rt,#29	*carry_out	
		AND Rt,#1	3 = 1 111	
uint32x4_t [arm_]vsbciq[_u32](uint32x4_t a,		VSBCI.I32 Qd,Qn,Qm	Qd -> result	MVE
	a -> Qn	In the party and	TO .	
uint32x4_t b, unsigned * carry_out)	a -> Qn b -> Qm	VMRS Rt,FPSCR_nzcvqc	Rt ->	
		LSR Rt,#29	Rt -> *carry_out	
		- 1		MVE
uint32x4_t b, unsigned * carry_out) int32x4_t [_arm_]vsbciq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, unsigned * carry_out,	b -> Qm inactive -> Qd a -> Qn	LSR Rt,#29 AND Rt,#1 VMSR P0,Rp VPST	*carry_out Qd -> result Rt ->	MVE
uint32x4_t b, unsigned * carry_out) int32x4_t [_arm_]vsbciq_m[_s32](int32x4_t inactive,	b -> Qm inactive -> Qd a -> Qn b -> Qm	LSR Rt,#29 AND Rt,#1 VMSR P0,Rp VPST VSBCIT.I32 Qd,Qn,Qm	*carry_out Qd -> result	MVE
uint32x4_t b, unsigned * carry_out) int32x4_t [_arm_]vsbciq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, unsigned * carry_out,	b -> Qm inactive -> Qd a -> Qn	LSR Rt,#29 AND Rt,#1 VMSR P0,Rp VPST	*carry_out Qd -> result Rt ->	MVE

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_nzevqe BFI Rs,Rt,#29,#1 VMSR FPSCR_nzevqe,Rs VMSR P0,Rp VPST VSBCT.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqe 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.II6 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.I8 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)	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	VMSR P0,Rp VPST VSUBT.I32 Qd,Qn,Qm VMSR P0,Rp	Qd -> result Qd -> result	MVE
int8x1o_t [arm]vsubq_x[_n_s8](int8x1o_t a, int8_t b, mve_pred16_t p) int16x8_t [arm]vsubq_x[_n_s16](int16x8_t a, int16_t	a -> Qn b -> Rm p -> Rp	VMSR PURP VPST VSUBT.I8 Qd,Qn,Rm VMSR P0,Rp	Qd -> result Qd -> result	MVE MVE
b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I16 Qd,Qn,Rm	Qu -> resuit	IVIVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [arm_]vsubq_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_]vsubq_x[_u8](uint8x16_t a,	p -> Rp a -> On	VSUBT.I32 Qd,Qn,Rm VMSR P0.Rp	Od -> result	MVE
uint8x16_t [arm_]vsuoq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR PO,RP VPST	Qa -> result	MVE
	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 b, mve_pred16_t p)	b -> Qm	VPST	Qu' y Tesun	111,12
	p -> Rp	VSUBT.I32 Qd,Qn,Qm		
uint8x16_t [arm_]vsubq_x[_n_u8](uint8x16_t a, uint8_t	a -> Qn b -> Rm	VMSR P0,Rp VPST	Qd -> result	MVE
b, mve_pred16_t p)	p -> Rn	VSUBT.I8 Qd,Qn,Rm		
uint16x8_t [arm_]vsubq_x[_n_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint16_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VSUBT.I16 Qd,Qn,Rm		
uint32x4_t [_arm_]vsubq_x[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Qd -> result	MVE
umi32_t b, mve_pred10_t p)	p -> Rm	VSUBT.I32 Qd,Qn,Rm		
float16x8_t [arm_]vsubq_x[_f16](float16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
G 20 to f 20 to f move 20 to	p -> Rp	VSUBT.F16 Qd,Qn,Qm	0.1	NO.
float32x4_t [arm_]vsubq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
noat32x4_t b, mvc_picu1o_t p)	p -> Rp	VSUBT.F32 Qd,Qn,Qm		
float16x8_t [arm_]vsubq_x[_n_f16](float16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float16_t b, mve_pred16_t p)	b -> Rm	VPST		
float32x4 t [arm]vsubq x[n f32](float32x4 t a,	p -> Rp	VSUBT.F16 Qd,Qn,Rm	Od -> result	MVE
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
nouts2_t b, mvc_prearo_t p)	p -> Rp	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, float32x4_t b)	a -> Qn b -> Qm	VCADD.F32 Qd,Qn,Qm,#90	Qd -> result	MVE/NEON
int8x16_t [_arm_]vcaddq_rot90[_s8](int8x16_t a,	a -> Qn	VCADD.I8 Qd,Qn,Qm,#90	Od -> result	MVE
int8x16_t b)	b -> Qm		Q 2 / 220201	
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	VCADD 122 04 0 0 #00	0.1 >	MVE
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 -> Qn	VCADD.I8 Qd,Qn,Qm,#90	Qd -> 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 [aim_]vcaddq_fot90[_u32](uint32x4_t a, uint32x4_t b)	b -> Qm	VCADD.132 Qu,Qii,Qiii,#90	Qu -> result	IVIVE
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	VC/102.10 Qu,Qii,Qiii,#270	Qu > resuit	III 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, int32x4_t b)	a -> Qn b -> Qm	VCADD.I32 Qd,Qn,Qm,#270	Qd -> result	MVE
uint8x16_t [arm_]vcaddq_rot270[_u8](uint8x16_t a,	a -> Qn	VCADD.I8 Qd,Qn,Qm,#270	Qd -> result	MVE
uint8x16_t b)	b -> Qm		Q 2 / 22020	
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	VCADD 122 04 0 0 1270	04 : 1:	MATE
uint32x4_t [_arm_]vcaddq_rot270[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VCADD.I32 Qd,Qn,Qm,#270	Qd -> result	MVE
float16x8_t [_arm_]vcaddq_rot90_m[_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VCADDT.F16 Qd,Qn,Qm,#90		
float32x4_t [arm_]vcaddq_rot90_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	IVI V I
2	b -> Qm	VCADDT.F32 Qd,Qn,Qm,#90		
	p -> Rp			

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vcaddq_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 b -> Qm p -> Rp	VPST VCADDT.I8 Qd,Qn,Qm,#90		
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 p -> Rp	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)	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
$ \begin{array}{lll} float 32x4_t \ [_arm_]vcaddq_rot 90_x [_f 32](float 32x4_t \ a, \\ float 32x4_t \ b, \ mve_pred 16_t \ p) \end{array} $	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.18 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, uint32x4_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
float16x8_t [arm_]vcaddq_rot270_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,#270	Qd -> result	MVE
float32x4_t [_arm_]vcaddq_rot270_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,#270	Qd -> result	MVE
int8x16_t [_arm_]vcaddq_rot270_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,#270	Qd -> result	MVE
int16x8_t [arm_]vcaddq_rot270_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,#270	Qd -> result	MVE
int32x4_t [arm_]vcaddq_rot270_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,#270	Qd -> result	MVE
uint8x16_t [_arm_]vcaddq_rot270_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
uint16x8_t [_arm_]vcaddq_rot270_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VCADDT.I8 Qd,Qn,Qm,#270 VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t [_arm_]vcaddq_rot270_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm p -> Rp	VCADDT.I16 Qd,Qn,Qm,#270 VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#270	Qd -> result	MVE
float16x8_t [_arm_]vcmlaq[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Om	VCMLA.F16 Qda,Qn,Qm,#0	Qda -> result	MVE/NEON
float32x4_t [_arm_]vcmlaq[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F32 Qda,Qn,Qm,#0	Qda -> result	MVE/NEON
float16x8_t [_arm_]vcmlaq_rot90[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F16 Qda,Qn,Qm,#90	Qda -> result	MVE/NEON
float32x4_t [_arm_]vcmlaq_rot90[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F32 Qda,Qn,Qm,#90	Qda -> result	MVE/NEON
float16x8_t [_arm_]vcmlaq_rot180[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F16 Qda,Qn,Qm,#180	Qda -> result	MVE/NEON
float32x4_t [_arm_]vcmlaq_rot180[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F32 Qda,Qn,Qm,#180	Qda -> result	MVE/NEON
float16x8_t [_arm_]vcmlaq_rot270[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F16 Qda,Qn,Qm,#270	Qda -> result	MVE/NEON
float32x4_t [_arm_]vcmlaq_rot270[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F32 Qda,Qn,Qm,#270	Qda -> result	MVE/NEON
float16x8_t [_arm_]vcmlaq_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 VCMLAT.F16 Qda,Qn,Qm,#0	Qda -> result	MVE
float32x4_t [_arm_]vcmlaq_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 VCMLAT.F32 Qda,Qn,Qm,#0	Qda -> result	MVE
float16x8_t [_arm_]vcmlaq_rot90_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 VCMLAT.F16 Qda,Qn,Qm,#90	Qda -> result	MVE
float32x4_t [_arm_]vcmlaq_rot90_m[_f32](float32x4_t a, float32x4_t b, float32x4_t c, mve_pred16_t p)	p -> Rp a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VCMLAT.F32 Qda,Qn,Qm,#90	Qda -> result	MVE
float16x8_t [_arm_]vcmlaq_rot180_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 VCMLAT.F16 Qda,Qn,Qm,#180	Qda -> result	MVE
float32x4_t [_arm_]vcmlaq_rot180_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 VCMLAT.F32 Qda,Qn,Qm,#180	Qda -> result	MVE

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 p -> Rp	VPST VCMLAT.F16 Qda,Qn,Qm,#270		
float32x4_t [arm_]vcmlaq_rot270_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	VPST		
	c -> Qm p -> Rp	VCMLAT.F32 Qda,Qn,Qm,#270		
float16x8_t [arm_]vcmulq[_f16](float16x8_t a,	a -> Qn	VCMUL.F16 Qd,Qn,Qm,#0	Qd -> result	MVE
float16x8_t b) float32x4 t [arm]vcmulg[f32](float32x4 t a,	b -> Qm a -> On	VCMUL.F32 Qd,Qn,Qm,#0	Od -> result	MVE
float32x4_t b)	b -> Qm	VCWOL.1-32 Qu,Qii,Qiii,#0	Qu -> resuit	WVE
float16x8_t [_arm_]vcmulq_rot90[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMUL.F16 Qd,Qn,Qm,#90	Qd -> result	MVE
float32x4_t [_arm_]vcmulq_rot90[_f32](float32x4_t a,	a -> Qn	VCMUL.F32 Qd,Qn,Qm,#90	Qd -> result	MVE
float32x4_t b) float16x8_t [arm_]vcmulq_rot180[_f16](float16x8_t a,	b -> Qm a -> Qn	VCMUL.F16 Qd,Qn,Qm,#180	Qd -> result	MVE
float16x8_t b) float32x4_t [arm_]vcmulq_rot180[_f32](float32x4_t a,	b -> Qm a -> Qn	VCMUL.F32 Qd,Qn,Qm,#180	Od -> result	MVE
float32x4_t b)	b -> Qm		`	
float16x8_t [_arm_]vcmulq_rot270[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMUL.F16 Qd,Qn,Qm,#270	Qd -> result	MVE
float32x4_t [arm_]vcmulq_rot270[_f32](float32x4_t a,	a -> Qn	VCMUL.F32 Qd,Qn,Qm,#270	Qd -> result	MVE
float32x4_t b) float16x8_t [arm_]vcmulq_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 VCMULT.F16 Qd,Qn,Qm,#0		
	p -> Rp	VCMULT.F16 Qd,Qfi,Qffi,#0		
float32x4_t [_arm_]vcmulq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, moat32x4_t a, moat32x4_t b, mve_pred1o_t p)	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	Qu -> resurt	MVL
	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, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, moarroxo_t a, noatroxo_t o, mve_preuro_t p)	b -> Qm	VCMULT.F16 Qd,Qn,Qm,#180		
float32x4_t [arm_]vcmulq_rot180_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	VPST	Qu > resuit	I III V E
	b -> Qm p -> Rp	VCMULT.F32 Qd,Qn,Qm,#180		
float16x8_t [arm_]vcmulq_rot270_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 b -> Qm	VPST VCMULT.F16 Qd,Qn,Qm,#270		
	p -> Rp			
float32x4_t [_arm_]vcmulq_rot270_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, noat32x4_t a, noat32x4_t b, mvc_pred16_t p)	b -> Qm	VCMULT.F32 Qd,Qn,Qm,#270		
float16x8_t [arm_]vcmulq_x[_f16](float16x8_t a,	p -> Rp a -> On	VMSR P0,Rp	Od -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Q 2	
float32x4 t [arm]vcmulq x[f32](float32x4 t a,	p -> Rp a -> Qn	VCMULT.F16 Qd,Qn,Qm,#0 VMSR P0,Rp	Qd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
float16x8_t [arm_]vcmulq_rot90_x[_f16](float16x8_t a,	p -> Rp a -> Qn	VCMULT.F32 Qd,Qn,Qm,#0 VMSR P0,Rp	Qd -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
float32x4_t [_arm_]vcmulq_rot90_x[_f32](float32x4_t a,	p -> Rp a -> Qn	VCMULT.F16 Qd,Qn,Qm,#90 VMSR P0,Rp	Qd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
float16x8_t [arm_]vcmulq_rot180_x[_f16](float16x8_t	p -> Rp a -> Qn	VCMULT.F32 Qd,Qn,Qm,#90 VMSR P0,Rp	Qd -> result	MVE
a, float16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
float32x4_t [arm_]vcmulq_rot180_x[_f32](float32x4_t	p -> Rp a -> Qn	VCMULT.F16 Qd,Qn,Qm,#180 VMSR P0,Rp	Qd -> result	MVE
a, float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
float16x8_t [arm_]vcmulq_rot270_x[_f16](float16x8_t	p -> Rp a -> Qn	VCMULT.F32 Qd,Qn,Qm,#180 VMSR P0,Rp	Qd -> result	MVE
a, float16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMULT.F16 Qd,Qn,Qm,#270		

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		
1.0.16.1	p -> Rp	VCMULT.F32 Qd,Qn,Qm,#270	0.1	NAME AND ONLY
int8x16_t [_arm_]vqabsq[_s8](int8x16_t a)	a -> Qm	VQABS.S8 Qd,Qm	Qd -> result	MVE/NEON 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 -> Qm	VPST		
	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 p -> Rp	VPST VQABST.S16 Qd,Qm		
int32x4_t [arm_]vqabsq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Qu' > Tesun	
	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	VOADD SICOLOG Dog	0.1 >16	MVE
int16x8_t [arm_]vqaddq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQADD.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [arm_]vqaddq[_n_s32](int32x4_t a, int32_t b)	a -> Qn	VQADD.S32 Qd,Qn,Rm	Od -> result	MVE
	b -> Rm			
uint8x16_t [arm_]vqaddq[_n_u8](uint8x16_t a, uint8_t	a -> Qn	VQADD.U8 Qd,Qn,Rm	Qd -> result	MVE
b)	b -> Rm			
uint16x8_t [arm_]vqaddq[_n_u16](uint16x8_t a,	a -> Qn	VQADD.U16 Qd,Qn,Rm	Qd -> result	MVE
uint16_t b) uint32x4_t [arm_]vqaddq[_n_u32](uint32x4_t a,	b -> Rm a -> On	VOADD.U32 Qd,Qn,Rm	Od -> result	MVE
uint32_t b)	b -> Rm	VQADD:032 Qu,Qii,Kiii	Qu => result	WIVE
int8x16_t [arm_]vqaddq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VQADD.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
	b -> Qm	2 . 2 . 2	Ì	
$int16x8_t\ [__arm_]vqaddq[_s16](int16x8_t\ a,\ int16x8_t\ b)$	a -> Qn	VQADD.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
	b -> Qm	VO.1777 010 0	0.1	NAME AND ONLY
int32x4_t [arm_]vqaddq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQADD.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vqaddq[_u8](uint8x16_t a, uint8x16_t	a -> On	VQADD.U8 Qd,Qn,Qm	Od -> result	MVE/NEON
b)	b -> Qm		Q	
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) int8x16_t [_arm_]vqaddq_m[_n_s8](int8x16_t inactive,	b -> Qm inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn	VPST	Qu -> resuit	WIVE
	b -> Rm	VQADDT.S8 Qd,Qn,Rm		
	p -> Rp			
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 b -> Rm	VPST VQADDT.S16 Qd,Qn,Rm		
	p -> Rp	VQADD1.310 Qu,Qii,Kiii		
int32x4_t [arm_]vqaddq_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	VPST		
	b -> Rm	VQADDT.S32 Qd,Qn,Rm		
' 10 16 15 1 11 1 1 01/ ' 10 16 1	p -> Rp	VALGE DO D	01	NOWE
uint8x16_t [arm_]vqaddq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
macave, unitox10_t a, unito_t b, mvc_picu10_t p)	b -> Rm	VQADDT.U8 Qd,Qn,Rm		
	p -> Rp	2	<u> </u>	
uint16x8_t [arm_]vqaddq_m[_n_u16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm	VQADDT.U16 Qd,Qn,Rm		
uint32x4_t [arm_]vqaddq_m[_n_u32](uint32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn	VPST	Qu > 1csuit	111,12
	b -> Rm	VQADDT.U32 Qd,Qn,Rm		
	p -> Rp			
int8x16_t [_arm_]vqaddq_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 VQADDT.S8 Qd,Qn,Qm		
	p -> QIII	, AVD 1.39 An'All'Alli		
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		
::-/20-4 4 f	p -> Rp	VACD DO D.	01 > 1:	MVE
int32x4_t [_arm_]vqaddq_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
moza-t_t a, moza-t_t o, mvc_predio_t p)	b -> Qm	VQADDT.S32 Qd,Qn,Qm		
	p -> Rp		1	l .

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 b -> Qm	VQRDMLADHX.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vqrdmladhxq[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b)	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 [_arm_]vqdmlahq_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 VQDMLAHT.S32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [arm_]vqrdmlahq[_n_s8](int8x16_t a, int8x16_t b, int8_t c)	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	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)	c -> Rm 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 c -> Rm	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)	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
$\label{eq:continuous} \begin{array}{ll} int8x16_t \ [_arm_]vqdmlsdhq_m[_s8](int8x16_t \ inactive, \\ int8x16_t \ a, \ int8x16_t \ b, \ mve_pred16_t \ p) \end{array}$	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, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMULHT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vqrdmulhq_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 VQRDMULHT.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vqrdmulhq_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 VQRDMULHT.S32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vqrdmulhq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VQRDMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqrdmulhq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQRDMULH.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
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, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMULHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqrdmulhq_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 VQRDMULHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vqrdmulhq_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 VQRDMULHT.S32 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vqdmullbq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQDMULLB.S16 Qd,Qn,Rm	Qd -> result	MVE
int64x2_t [_arm_]vqdmullbq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VQDMULLB.S32 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vqdmullbq_m[_n_s16](int32x4_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULLBT.S16 Qd,Qn,Rm	Qd -> result	MVE
int64x2_t [arm_]vqdmullbq_m[_n_s32](int64x2_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULLBT.S32 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vqdmullbq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQDMULLB.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [_arm_]vqdmullbq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQDMULLB.S32 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vqdmullbq_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 VQDMULLBT.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [arm_]vqdmullbq_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 VQDMULLBT.S32 Qd,Qn,Qm	Qd -> result	MVE
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 b)	a -> Qn b -> Rm	VQDMULLT.S32 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vqdmulltq_m[_n_s16](int32x4_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULLTT.S16 Qd,Qn,Rm	Qd -> result	MVE
int64x2_t [arm_]vqdmulltq_m[_n_s32](int64x2_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULLTT.S32 Qd,Qn,Rm	Qd -> result	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 b)	a -> Qn b -> Qm	VQDMULLT.S32 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vqdmulltq_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 VQDMULLTT.S16 Qd,Qn,Qm	Qd -> result	MVE

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	VPST		
	b -> Qm	VQDMULLTT.S32 Qd,Qn,Qm		
int8x16_t [arm_]vqnegq[_s8](int8x16_t a)	p -> Rp a -> Qm	VQNEG.S8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vqnegq[_s16](int16x8_t a)	a -> Qm	VQNEG.S16 Qd,Qm	Od -> result	MVE/NEON
int32x4_t [arm_]vqnegq[_s32](int32x4_t a)	a -> Qm	VQNEG.S32 Qd,Qm	Qd -> result	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		
. 16.0	p -> Rp	VQNEGT.S8 Qd,Qm	01 . 1	MATE
int16x8_t [arm_]vqnegq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
microxo_t a, mvc_picuro_t p)	p -> Rp	VQNEGT.S16 Qd,Qm		
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	VPST		
	p -> Rp	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
int16x8_t [arm_]vqsubq[_n_s16](int16x8_t a, int16_t b)	b -> Rm a -> On	VQSUB.S16 Qd,Qn,Rm	Od -> result	MVE
introxo_t [arm_]vqsubq[_n_sro](introxo_t a, intro_t b)	b -> Rm	VQSCB.S10 Qu,Qii,Kiii	Qu -> resuit	WIVE
int32x4_t [arm_]vqsubq[_n_s32](int32x4_t a, int32_t b)	a -> Qn	VQSUB.S32 Qd,Qn,Rm	Qd -> result	MVE
	b -> Rm			
uint8x16_t [arm_]vqsubq[_n_u8](uint8x16_t a, uint8_t	a -> Qn	VQSUB.U8 Qd,Qn,Rm	Qd -> result	MVE
b) uint16x8_t [arm_]vqsubq[_n_u16](uint16x8_t a,	b -> Rm a -> On	VQSUB.U16 Qd,Qn,Rm	Od -> result	MVE
uint16x6_t [arm_]vqsubq[_n_u10](uint16x6_t a, uint16_t b)	b -> Rm	VQSOB.O10 Qu,Qii,Kiii	Qu -> resuit	WIVE
uint32x4_t [arm_]vqsubq[_n_u32](uint32x4_t a,	a -> On	VQSUB.U32 Qd,Qn,Rm	Od -> result	MVE
uint32_t b)	b -> Rm	(1,000)		
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	VPST		
	b -> Rm p -> Rp	VQSUBT.S8 Qd,Qn,Rm		
int16x8_t [arm_]vqsubq_m[_n_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn	VPST	Q	1
	b -> Rm	VQSUBT.S16 Qd,Qn,Rm		
	p -> Rp			
int32x4_t [_arm_]vqsubq_m[_n_s32](int32x4_t inactive,	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPS1 VQSUBT.S32 Qd,Qn,Rm		
	p -> Rp	, 490B1.532 (d, 41, 1011		
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	VPST		
	b -> Rm	VQSUBT.U8 Qd,Qn,Rm		
vint16v0 t [ome lyggyleg m[n v16]/vint16v0 t	p -> Rp inactive -> Od	VMCD DO Do	Od -> result	MVE
uint16x8_t [arm_]vqsubq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, unitroxo_t a, unitro_t b, inve_prearo_t p)	b -> Rm	VQSUBT.U16 Qd,Qn,Rm		
	p -> Rp			
uint32x4_t [arm_]vqsubq_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	VQSUBT.U32 Qd,Qn,Rm		
int8x16_t [arm_]vqsubq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VQSUB.S8 Qd,Qn,Qm	Od -> result	MVE/NEON
mionio_t [ami_] (qoaoq[_bo](mionio_t a, mionio_t o)	b -> Qm	, 6002.00 60,61,611	Qu' > 100uit	1,1,12,1,201,
int16x8_t [arm_]vqsubq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VQSUB.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
	b -> Qm			
int32x4_t [arm_]vqsubq[_s32](int32x4_t a, int32x4_t b)	a -> Qn	VQSUB.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vqsubq[_u8](uint8x16_t a, uint8x16_t	b -> Qm a -> Qn	VQSUB.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
b)	b -> Qm	VQSCB.Co Qu,Qii,Qiii	Qu -> result	WIVE/NEON
uint16x8_t [arm_]vqsubq[_u16](uint16x8_t a,	a -> Qn	VQSUB.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t b)	b -> Qm			
uint32x4_t [arm_]vqsubq[_u32](uint32x4_t a,	a -> Qn	VQSUB.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t b) int8x16 t [arm]vqsubq m[s8](int8x16 t inactive,	b -> Qm inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qu	VPST VPST	Qu -> Icsuit	141 4 15
_ / r · - r · - r · - r · r ·	b -> Qm	VQSUBT.S8 Qd,Qn,Qm		
	p -> Rp			
		VMSR P0,Rp	Qd -> result	MVE
int16x8_t [_arm_]vqsubq_m[_s16](int16x8_t inactive,	inactive -> Qd			
int16x8_t [_arm_]vqsubq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	a -> Qn b -> Qm	VPST VQSUBT.S16 Qd,Qn,Qm		
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VQSUBT.S16 Qd,Qn,Qm	Od -> result	MVE
	a -> Qn b -> Qm		Qd -> result	MVE

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 Qd -> 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 [armvldrbq_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]		
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
	p > Kp	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)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE/NEON
int32x4_t [arm_]vld1q[_s32](int32_t const * base) uint8x16_t [arm_]vld1q[_u8](uint8_t const * base)	base -> Rn base -> Rn	VLDRW.32 Qd,[Rn] VLDRB.8 Qd,[Rn]	Qd -> result 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,	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VLDRBT.8 Qd,[Rn]		
int16x8_t [arm_]vld1q_z[_s16](int16_t const * base,	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VLDRHT.16 Qd,[Rn]		
int32x4_t [_arm_]vld1q_z[_s32](int32_t const * base,	base -> Rn	VLDRH1.10 Qu,[KII] VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Q	
		VLDRWT.32 Qd,[Rn]		
uint8x16_t [_arm_]vld1q_z[_u8](uint8_t const * base,	base -> Rn	VMSR P0,Rp VPST	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VLDRBT.8 Qd,[Rn]		
uint16x8_t [arm_]vld1q_z[_u16](uint16_t const * base,	base -> Rn	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VLDRHT.16 Qd,[Rn]		
uint32x4_t [_arm_]vld1q_z[_u32](uint32_t const * base,	base -> Rn	VMSR P0,Rp VPST	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPS1 VLDRWT.32 Qd,[Rn]		
float16x8_t [arm_]vld1q_z[_f16](float16_t const * base,	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VLDRHT.16 Qd,[Rn]		
float32x4_t [arm_]vld1q_z[_f32](float32_t const * base,	base -> Rn	VLDRH1.10 Qu,[KII] VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu' > Tesun	
		VLDRWT.32 Qd,[Rn]		1.07
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	VLDRH.U16 Qd,[Rn,Qm]	Qd -> result	MVE
const * base, uint16x8_t offset) uint32x4_t [arm_]vldrhq_gather_offset[_u32](uint16_t	offset -> Qm 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	VLDRH.F16 Qd,[Rn,Qm]	Qd -> result	MVE
const * base, uint16x8_t offset)	base -> Rn offset -> Qm	VLDKIT.F10 Qd,[Kn,Qm]	Qu -> resuit	MVE
int16x8_t [arm_]vldrhq_gather_offset_z[_s16](int16_t	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
const * base, uint16x8_t offset, mve_pred16_t p)	offset -> Qm	VPST		
int32x4_t [arm_]vldrhq_gather_offset_z[_s32](int16_t	p -> Rp base -> Rn	VLDRHT.U16 Qd,[Rn,Qm] VMSR P0,Rp	Od -> result	MVE
const * base, uint32x4_t offset, mve_pred16_t p)	offset -> Qm	VPST	Qu -> resuit	1V1 V ID
uint16x8_t	p -> Rp base -> Rn	VLDRHT.S32 Qd,[Rn,Qm] VMSR P0,Rp	Qd -> result	MVE
uint10x8_t [_arm_]vldrhq_gather_offset_z[_u16](uint16_t const *	offset -> Qm	VMSR PU,RP VPST	Qu -> resuit	IVI V E
base, uint16x8_t offset, mve_pred16_t p)	p -> Rp	VLDRHT.U16 Qd,[Rn,Qm]		
uint32x4_t	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
[_arm_]vldrhq_gather_offset_z[_u32](uint16_t const * base, uint32x4_t offset, mve_pred16_t p)	offset -> Qm p -> Rp	VPST VLDRHT.U32 Qd,[Rn,Qm]		
float16x8_t	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
[_arm_]vldrhq_gather_offset_z[_f16](float16_t const *	offset -> Qm	VPST		
base, uint16x8_t offset, mve_pred16_t p) int16x8_t	p -> Rp base -> Rn	VLDRHT.F16 Qd,[Rn,Qm] VLDRH.U16 Qd,[Rn,Qm,UXTW	Qd -> result	MVE
[_arm_]vldrhq_gather_shifted_offset[_s16](int16_t const	offset -> Qm	#1]	Qu -> resuit	101 0 15
* base, uint16x8_t offset)	1 P	Al DDH 633 O LD C AMAZZA	01.	Myr
int32x4_t [_arm_]vldrhq_gather_shifted_offset[_s32](int16_t const	base -> Rn offset -> Qm	VLDRH.S32 Qd,[Rn,Qm,UXTW #1]	Qd -> result	MVE
* base, uint32x4_t offset)	onset -/ Qili	"1]		

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 [_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 PO,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 [_arm_]vldrwq_gather_shifted_offset[_u32](uint32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRW.U32 Qd,[Rn,Qm,UXTW #2]	Qd -> result	MVE
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_lvldrwq_gather_shifted_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,UXTW #2]	Qd -> result	MVE
uint32x4_t [_arm_]vldrwq_gather_shifted_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,UXTW #2]	Qd -> result	MVE
float32x4_t [_arm_]vldrwq_gather_shifted_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,UXTW #2]	Qd -> result	MVE
int32x4_t [arm_]vldrwq_gather_base_s32(uint32x4_t addr, const int offset)	addr -> Qn offset in +/- 4*[0127]	VLDRW.U32 Qd,[Qn,#offset]	Qd -> result	MVE
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 +/- 4*[0127]	VLDRW.U32 Qd,[Qn,#offset]	Qd -> result	MVE
int32x4_t [_arm_]vldrwq_gather_base_z_s32(uint32x4_t addr, const int offset, mve_pred16_t p)	addr -> Qn offset in +/- 4*[0127] p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Qn,#offset]	Qd -> result	MVE
uint32x4_t [arm_]vldrwq_gather_base_z_u32(uint32x4_t addr, const int offset, mve_pred16_t p)	addr -> Qn offset in +/- 4*[0127] p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Qn,#offset]	Qd -> result	MVE
float32x4_t [_arm_]vldrwq_gather_base_z_f32(uint32x4_t addr, const int offset, mve_pred16_t p)	addr -> Qn offset in +/- 4*[0127] p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Qn,#offset]	Qd -> result	MVE
int32x4_t [_arm_]vldrwq_gather_base_wb_s32(uint32x4_t * addr, const int offset)	*addr -> Qn offset in +/- 4*[0127]	VLDRW.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
uint32x4_t [arm_]vldrwq_gather_base_wb_u32(uint32x4_t * addr, const int offset)	*addr -> Qn offset in +/- 4*[0127]	VLDRW.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
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 [arm_]vldrwq_gather_base_wb_z_s32(uint32x4_t * addr, const int offset, mve_pred16_t p)	*addr -> Qn offset in +/- 4*[0127] p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
uint32x4_t [_arm_]vldrwq_gather_base_wb_z_u32(uint32x4_t * addr, const int offset, mve_pred16_t p)	*addr -> Qn offset in +/- 4*[0127] p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
float32x4_t [arm_]vldrwq_gather_base_wb_z_f32(uint32x4_t * addr, const int offset, mve_pred16_t p)	*addr -> Qn offset in +/- 4*[0127] p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
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 p -> Rp	VMSR P0,Rp VPST VLDRDT.U64 Qd,[Rn,Qm]	Qd -> result	MVE
uint64x2_t [_arm_]vldrdq_gather_offset_z[_u64](uint64_t const * base, uint64x2_t offset, mve_pred16_t p) int64x2_t	base -> Rn offset -> Qm p -> Rp base -> Rn	VMSR P0,Rp VPST VLDRDT.U64 Qd,[Rn,Qm] VLDRD.U64 Qd,[Rn,Qm,UXTW	Qd -> result Qd -> result	MVE MVE
[_arm_]vldrdq_gather_shifted_offset[_s64](int64_t const * base, uint64x2_t offset)	offset -> Qm	#3]		
uint64x2_t [arm_]vldrdq_gather_shifted_offset[_u64](uint64_t const * base, uint64x2_t offset)	base -> Rn offset -> Qm	VLDRD.U64 Qd,[Rn,Qm,UXTW #3]	Qd -> result	MVE

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_]vldrdq_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] -> Qd2	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] -> Od4	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] 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	base -> Rn	VSTRB.16 Qd,[Rn]	void -> result	MVE
value) void [_arm_]vstrbq[_u32](uint8_t * base, uint32x4_t	value -> Qd base -> Rn	VSTRB.32 Qd,[Rn]	void -> result	MVE
value) void [_arm_]vstrbq_p[_s8](int8_t * base, int8x16_t value,	value -> Qd base -> Rn	VMSR P0,Rp	void -> result	MVE
mve_pred16_t p)	value -> Qd p -> Rp	VPST VSTRBT.8 Qd,[Rn]		
void [_arm_]vstrbq_p[_s16](int8_t * base, int16x8_t value, mve_pred16_t p)	base -> Rn value -> Qd	VMSR P0,Rp VPST	void -> result	MVE
void [_arm_]vstrbq_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]	void -> result	WVE
void [_arm_]vstrbq_p[_u8](uint8_t * base, uint8x16_t	base -> Rn	VMSR P0,Rp	void -> result	MVE
value, mve_pred16_t p)	value -> Qd p -> Rp	VPST VSTRBT.8 Qd,[Rn]		
void [_arm_]vstrbq_p[_u16](uint8_t * base, uint16x8_t value, mve_pred16_t p)	base -> Rn value -> Qd	VMSR P0,Rp VPST	void -> result	MVE
void [_arm_]vstrbq_p[_u32](uint8_t * base, uint32x4_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]		
void [_arm_]vstrhq[_s16](int16_t * base, int16x8_t value)	base -> Rn	VSTRH.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrhq[_s32](int16_t * base, int32x4_t	value -> Qd base -> Rn	VSTRH.32 Qd,[Rn]	void -> result	MVE
value) void [_arm_]vstrhq[_u16](uint16_t * base, uint16x8_t	value -> Qd base -> Rn	VSTRH.16 Qd,[Rn]	void -> result	MVE
value) void [_arm_]vstrhq[_u32](uint16_t * base, uint32x4_t	value -> Qd base -> Rn	VSTRH.32 Qd,[Rn]	void -> result	MVE
value) void [_arm_]vstrhq[_f16](float16_t * base, float16x8_t	value -> Qd base -> Rn	VSTRH.16 Qd,[Rn]	void -> result	MVE
value) void [_arm_]vstrhq_p[_s16](int16_t * base, int16x8_t	value -> Qd base -> Rn	VMSR P0,Rp	void -> result	MVE
value, mve_pred16_t p)	value -> Qd	VPST	void => resuit	MVE
void [_arm_]vstrhq_p[_s32](int16_t * base, int32x4_t	p -> Rp base -> Rn	VSTRHT.16 Qd,[Rn] VMSR P0,Rp	void -> result	MVE
value, mve_pred16_t p)	value -> Qd p -> Rp	VPST VSTRHT.32 Qd,[Rn]		
void [_arm_]vstrhq_p[_u16](uint16_t * base, uint16x8_t value, mve_pred16_t p)	base -> Rn value -> Qd	VMSR P0,Rp VPST	void -> result	MVE
void [arm_]vstrhq_p[_u32](uint16_t * base, uint32x4_t	p -> Rp base -> Rn	VSTRHT.16 Qd,[Rn] VMSR P0,Rp	void -> result	MVE
value, mve_pred16_t p)	value -> Qd p -> Rp	VPST VSTRHT.32 Qd,[Rn]		
void [_arm_]vstrhq_p[_f16](float16_t * base, float16x8_t	base -> Rn value -> Od	VMSR P0,Rp VPST	void -> result	MVE
value, mve_pred16_t p)	p -> Rp	VSTRHT.16 Qd,[Rn]		
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	VMSR P0,Rp VPST	void -> result	MVE
	p -> Rp base -> Rn	VSTRWT.32 Qd,[Rn]	void -> result	MVE
<pre>void [_arm_]vstrwq_p[_u32](uint32_t * base, uint32x4_t value, mve_pred16_t p)</pre>	value -> Qd	VMSR P0,Rp VPST	void -> resuit	MVE
void [_arm_]vstrwq_p[_f32](float32_t * base,	p -> Rp base -> Rn	VSTRWT.32 Qd,[Rn] VMSR P0,Rp	void -> result	MVE
float32x4_t value, mve_pred16_t p)	value -> Qd p -> Rp	VPST VSTRWT.32 Qd,[Rn]		
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	VSTRW.32 Qd,[Rn]	void -> result	MVE/NEON
void [_arm_]vst1q[_u8](uint8_t * base, uint8x16_t value)	value -> Qd base -> Rn	VSTRB.8 Qd,[Rn]	void -> result	MVE/NEON
void [_arm_]vst1q[_u16](uint16_t * base, uint16x8_t	value -> Qd base -> Rn	VSTRH.16 Qd,[Rn]	void -> result	MVE/NEON
value) void [_arm_]vst1q[_u32](uint32_t * base, uint32x4_t	value -> Qd base -> Rn	VSTRW.32 Qd,[Rn]	void -> result	MVE/NEON
value)	value -> Qd	Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z Z		

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	VSTRITT-10 Qd,[Rii] 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 #1]	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	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	#2] 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]	VPST	-> result	
	b -> Qm p -> Rp	VADDLVAT.S32 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	,2
	b -> Qm	VADDLVAT.U32		
	p -> Rp	RdaLo,RdaHi,Qm		
int64_t [arm_]vaddlvq[_s32](int32x4_t a)	a -> Qm	VADDLV.S32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE
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	-> result	
		VADDLVT.S32		
1.54 - 5 - 2 - 1 - 1 - 5 - 2 - 2 - 2 - 2 - 2 - 2 - 2 - 2 - 2		RdaLo,RdaHi,Qm	(D 1 YY: D 1 Y 3) a m
uint64_t [arm_]vaddlvq_p[_u32](uint32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	[RdaHi,RdaLo] -> result	MVE
nive_pred10_t p)	p -> Kp	VADDLVT.U32	-> resurt	
		RdaLo,RdaHi,Qm		
int32_t [arm_]vaddvaq[_s8](int32_t a, int8x16_t b)	a -> Rda b -> Om	VADDVA.S8 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvaq[_s16](int32_t a, int16x8_t b)	a -> Rda	VADDVA.S16 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvaq[_s32](int32_t a, int32x4_t b)	b -> Qm a -> Rda	VADDVA.S32 Rda,Qm	Rda -> result	MVE
uint32_t [_arm_]vaddvaq[_u8](uint32_t a, uint8x16_t b)	b -> Qm a -> Rda	VADDVA.U8 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vaddvaq[_u16](uint32_t a, uint16x8_t b)	b -> Qm a -> Rda	VADDVA.U16 Rda,Qm	Rda -> result	MVE
	b -> Qm			
uint32_t [arm_]vaddvaq[_u32](uint32_t a, uint32x4_t b)	a -> Rda b -> Qm	VADDVA.U32 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvaq_p[_s8](int32_t a, int8x16_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm p -> Rp	VPST VADDVAT.S8 Rda,Qm		
int32_t [arm_]vaddvaq_p[_s16](int32_t a, int16x8_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mvs_red16_t p)	b -> Qm	VPST	Rua -> resuit	WIVE
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
mve_pred16_t p)	b -> Qm	VPST	Rua => resuit	WIVE
mve_pleare_t p)	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		
	p -> Rp	VADDVAT.U8 Rda,Qm		
uint32_t [arm_]vaddvaq_p[_u16](uint32_t a, uint16x8_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
uint32_t [arm_]vaddvaq_p[_u32](uint32_t a, uint32x4_t	p -> Rp a -> Rda	VADDVAT.U16 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Om	VMSK PO,RP VPST	Aua -/ ICSUII	IVI V IS
-r	p -> Rp	VADDVAT.U32 Rda,Qm		
int32_t [arm_]vaddvq[_s8](int8x16_t a)	a -> Qm	VADDV.S8 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvq[_s16](int16x8_t a)	a -> Qm	VADDV.S16 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,	a -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
mve_pred16_t p)	p -> Rp	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 -> Rp	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		
uint32_t [arm_]vaddvq_p[_u8](uint8x16_t a,	a -> Qm	VADDVT.S32 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	a -> Qm p -> Rp	VMSR PO,RP VPST	Aua -> resuit	IVI V IL
	l r · · · · · ·	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	1	

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) uint32_t [_arm_]vmladavq[_u32](uint32x4_t a,	a -> Qn b -> Qm a -> Qn	VMLADAV.U16 Rda,Qn,Qm VMLADAV.U32 Rda,Qn,Qm	Rda -> result Rda -> result	MVE MVE
uint32_t [_arm_]viiiadavq[_u32](uint32x4_t a, uint32x4_t b) int32_t [_arm_]vmladavq_p[_s8](int8x16_t a, int8x16_t	a -> Qn b -> Qm a -> Qn	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VMSK F0,Kp VPST VMLADAVT.S8 Rda,Qn,Qm	Kua -> resuit	WIVE
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] -> result	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

mid22_t_l_mm_lymidawaq_st_st_end to st_end t	Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
C > Qm	int32_t [arm_]vmlsdavaq_p[_s32](int32_t a, int32x4_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
int22_t_mm_lymbidavq_strain_	b, int32x4_t c, mve_pred16_t p)	c -> Qm			
	int32_t [arm_]vmlsdavq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VMLSDAV.S8 Rda,Qn,Qm	Rda -> result	MVE
mi32_1 [_mm_ vmisdavaq_p]_s8[(mi82_d_1 a, mi81_e_1 b)	int32_t [arm_]vmlsdavq[_s16](int16x8_t a, int16x8_t b)		VMLSDAV.S16 Rda,Qn,Qm	Rda -> result	MVE
mid2_1 [_arm_ wmbdavq_p]_s8[(mid3_1 a, int8x16_1 b)	int32_t [arm_]vmlsdavq[_s32](int32x4_t a, int32x4_t b)		VMLSDAV.S32 Rda,Qn,Qm	Rda -> result	MVE
D		-		Rda -> result	MVE
b. me.pedld.pp b. > Qm	o, invo_predio_t p/		1 1		
ini32_t [_arm_ymlsdavaq_[_s3](ini32_t a, ini8x16_t b, b, color b > 0		b -> Qm	VPST	Rda -> result	MVE
b, mve, pred16_ip b	int32 t [arm lymlsdayg n[s32](int32x4 t a int32x4 t			Rda -> result	MVF.
im32_t [_arm_ vmlsdavaxq[_s16](im52_t a, im16x8_t b, b > 0n		b -> Qm	VPST	Rua -> result	MVL
C > Om	int32_t [arm_]vmlsdavaxq[_s8](int32_t a, int8x16_t b,			Rda -> result	MVE
b > 0m	int8x16_t c)				
ini324_t_c arm ymlsdavaxq_p[s32](ini32_t a, ini32xd_t b, b on c o om lini32xd_t c)		b -> Qn	VMLSDAVAX.S16 Rda,Qn,Qm	Rda -> result	MVE
c > Qm		a -> Rda	VMLSDAVAX.S32 Rda,Qn,Qm	Rda -> result	MVE
b. int8x16_t_c, mve_pred16_tp) c	me2.r(o)				
c > Qm				Rda -> result	MVE
mi32_t L_arm_ vmlsdavaxq_p[_s32](imt32x4_t a, imt16x8_t b, imt16x8_t c, mve_pred16_t p)	b, int8x16_t c, mve_pred16_t p)	c -> Qm			
b. int16x8_t c, mve_pred16_t p) b > Qn	int32 t [arm]vmlsdavayq n[s16](int32 t a int16x8 t		VMSR P0 Rn	Rda -> result	MVE
P > Rp		b -> Qn	VPST	Rua -> result	MVL
b. Qn					
C > Qm P > Rp Rda,Qm,Qm Rda > result MVE				Rda -> result	MVE
P	b, int32x4_t c, mve_pred16_t p)				
int32_t [_arm_ vmlsdavxq[_s16](int16x8_t a, int16x8_t b)		-			
b > Qm	int32_t [arm_]vmlsdavxq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VMLSDAVX.S8 Rda,Qn,Qm	Rda -> result	MVE
Int32_t [_arm_]vmlsdavxq[_s32](int32x4_t a, int32x4_t b)			VMLSDAVX.S16 Rda,Qn,Qm	Rda -> result	MVE
b, mve_pred16_t p) b > Qm p > Rp VVST VVMLSDAVXT.S8 Rda,Qn,Qm int32_t [_arm_]vmlsdavxq_p[_s16](int16x8_t a, a > Qn int32_t [_arm_]vmlsdavxq_p[_s32](int32x4_t a, a > Qn int32_t [_arm_]vmlsdavxq_p[_s32](int32x4_t a, a > Qn b > Qm p > Rp VVST VVMLSDAVXT.S16 Rda,Qn,Qm int32_t [_arm_]vmlsdavxq_p[_s32](int32x4_t a, a > Qn b > Qm p > Rp VMLSDAVXT.S16 Rda,Qn,Qm int64_t [_arm_]vmlsldavaq[_s16](int64_t a, int16x8_t b, int16x8_t c) int64_t [_arm_]vmlsldavaq[_s32](int64_t a, int32x4_t b, int16x8_t c) int64_t [_arm_]vmlsldavaq[_s32](int64_t a, int16x8_t b, int16x8_t c, mve_pred16_t p) int64_t [_arm_]vmlsldavaq_p[_s16](int64_t a, int16x8_t b, int16x8_t c, mve_pred16_t p) int64_t [_arm_]vmlsldavaq_p[_s32](int64_t a, int22x4_t b, int16x8_t c, mve_pred16_t p) int64_t [_arm_]vmlsldavaq_p[_s32](int64_t a, int32x4_t b, int32x4_t c, mve_pred16_t p) int64_t [_arm_]vmlsldavaq_p[_s32](int64_t a, int32x4_t b, int32x4_t c, mve_pred16_t p) int64_t [_arm_]vmlsldavaq_p[_s32](int64_t a, int32x4_t b, int64_t [_arm_]vmlsldavq[_s32](int64_t a, int32x4_t b, int64_t [_arm_]vmlsldavq[_s32](int6x_t a, int16x8_t b, ob > Qn			VMLSDAVX.S32 Rda,Qn,Qm	Rda -> result	MVE
p -> Rp		-		Rda -> result	MVE
int32_t	b, mve_pred16_t p)				
p -> Rp	int32_t [arm_]vmlsdavxq_p[_s16](int16x8_t a,		VMSR P0,Rp	Rda -> result	MVE
int32_t [_arm_]vmlsdavxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	int16x8_t b, mve_pred16_t p)				
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	int32 t [arm lymlsdayxq p[s32](int32x4 t a.			Rda -> result	MVE
int64_t [_arm_]vmlsldavaq[_s16](int64_t a, int16x8_t b, int16x8_t c)		b -> Qm	VPST		
$ \begin{array}{c} int16x8_t\ c) & \begin{bmatrix} RdaHi,RdaLo \\ b > Qn \\ c > Qm \\ \\ int32x4_t\ c) \\ \\ \hline \\ int64_t\ [_arm_] vmlsldavaq[_s32](int64_t\ a, int32x4_t\ b, \\ int32x4_t\ c) \\ \\ \hline \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ $	int64 tf arm lymleldayaaf a161/int64 to int1640 th			[D4aH; D4aI al	MVE
c -> Qm	· · · - · · - ·	[RdaHi,RdaLo]			MVE
int64_t [_arm_]vmlsldavaq[_s32](int64_t a, int32x4_t b, int32x4_t c)					
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	int64_t [arm_]vmlsldavaq[_s32](int64_t a, int32x4_t b,		VMLSLDAVA.S32	[RdaHi,RdaLo]	MVE
C -> Qm		[RdaHi,RdaLo]			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$		c -> Qm			
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$					MVE
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	o, merozo_t e, mve_preuro_t p)			-/ icsuit	
$ \begin{array}{c} int64_t \ [_arm_]vmlsldavaq_p[_s32](int64_t \ a, int32x4_t \ b, int32x4_t \ c, mve_pred16_t \ p) \\ \\ b = 0 \\ c = 0 \\ c$		c -> Qm			
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		a ->			MVE
$ \begin{array}{c ccccc} & c \rightarrow Qm & RdaLo,RdaHi,Qn,Qm \\ & p \rightarrow Rp & & & & & \\ \hline int64_t \left[_arm_]vmlsldavq\left[_s16\right](int16x8_t \ a, int16x8_t \\ b) & a \rightarrow Qn & VMLSLDAV.S16 & \left[RdaHi,RdaLo\right] & MVE \\ b \rightarrow Qm & RdaLo,RdaHi,Qn,Qm & -> result & & \\ \hline int64_t \left[_arm_]vmlsldavq\left[_s32\right](int32x4_t \ a, int32x4_t & a \rightarrow Qn & VMLSLDAV.S32 & \left[RdaHi,RdaLo\right] & MVE \\ \hline \end{array} $	b, int32x4_t c, mve_pred16_t p)			-> result	
p -> Rp VMLSLDAV.S16 [RdaHi,RdaLo] MVE int64_t [_arm_]vmlsldavq[_s16](int16x8_t a, int16x8_t b) a -> Qn VMLSLDAV.S16 [RdaHi,RdaLo] MVE b -> Qm RdaLo,RdaHi,Qn,Qm -> result -> result WLSLDAV.S32 [RdaHi,RdaLo] MVE					
b)		p -> Rp			
	b)	b -> Qm	RdaLo,RdaHi,Qn,Qm	-> result	
b) b -> Qm RdaLo,RdaHi,Qn,Qm -> result	·- ·	-			MVE

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	-> result	
int64_t [arm_]vmlsldavq_p[_s32](int32x4_t a, int32x4_t	0.2.00	RdaLo,RdaHi,Qn,Qm	[DdoII; DdoI o]	MVE
b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	[RdaHi,RdaLo] -> result	MVE
	p -> Rp	VMLSLDAVT.S32		
int64_t [arm_]vmlsldavaxq[_s16](int64_t a, int16x8_t b,	a ->	RdaLo,RdaHi,Qn,Qm VMLSLDAVAX.S16	[RdaHi,RdaLo]	MVE
int16x8_t c)	[RdaHi,RdaLo] b -> On	RdaLo,RdaHi,Qn,Qm	-> result	
	c -> Qn			
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
III(32A4_(C)	b -> Qn	Kualo,Kuarii,Qii,Qiii	-> icsuit	
int64_t [arm_]vmlsldavaxq_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	MVE
	b -> Qn c -> Qm	VMLSLDAVAXT.S16		
	p -> Rp	RdaLo,RdaHi,Qn,Qm		
int64_t [_arm_]vmlsldavaxq_p[_s32](int64_t a, int32x4_t	a -> [RdaHi,RdaLo]	VMSR P0,Rp VPST	[RdaHi,RdaLo] -> result	MVE
b, int32x4_t c, mve_pred16_t p)	b -> Qn	VMLSLDAVAXT.S32	-> resuit	
	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) int64 t [arm]vmlsldavxq[s32](int32x4 t a, int32x4 t	b -> Qm	RdaLo,RdaHi,Qn,Qm VMLSLDAVX.S32	-> result	MVE
into4_t [arm_jvmisidavxq[_s52](int52x4_t a, int52x4_t b)	a -> Qn b -> Qm	RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlsldavxq_p[_s16](int16x8_t a,	a -> Qn	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMLSLDAVXT.S16	-> result	
		RdaLo,RdaHi,Qn,Qm	ID I III D I I I	Marc
int64_t [arm_]vmlsldavxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	[RdaHi,RdaLo] -> result	MVE
_ / /	p -> Rp	VMLSLDAVXT.S32		
int8x16_t [arm_]vhaddq[_n_s8](int8x16_t a, int8_t b)	a -> Qn	RdaLo,RdaHi,Qn,Qm VHADD.S8 Qd,Qn,Rm	Qd -> result	MVE
	b -> Rm	AMARD GICOLO B	011	Marc
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,	a -> Qn	VHADD.U16 Qd,Qn,Rm	Qd -> result	MVE
uint16_t b) uint32x4_t [_arm_]vhaddq[_n_u32](uint32x4_t a,	b -> Rm a -> Qn	VHADD.U32 Qd,Qn,Rm	Qd -> result	MVE
uint32_t b)	b->Rm	7411 DD 00 010 0	011	MURATRON
int8x16_t [arm_]vhaddq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VHADD.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vhaddq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VHADD.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vhaddq[_s32](int32x4_t a, int32x4_t b)	b -> Qm a -> Qn	VHADD.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vhaddq[_u8](uint8x16_t a, uint8x16_t	b -> Qm a -> Qn	VHADD.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
b) uint16x8_t [arm_]vhaddq[_u16](uint16x8_t a,	b -> Qm a -> Qn	VHADD.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t b) uint32x4_t [arm_]vhaddq[_u32](uint32x4_t a,	b -> Qm a -> Qn	VHADD.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t b)	b->Qm			
int8x16_t [arm_]vhaddq_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
	b -> Rm	VHADDT.S8 Qd,Qn,Rm		
int16x8_t [arm_]vhaddq_m[_n_s16](int16x8_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm p -> Rp	VHADDT.S16 Qd,Qn,Rm		
int32x4_t [_arm_]vhaddq_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	VPST VHADDT.S32 Qd,Qn,Rm		
	p -> Rp			

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [arm_]vhaddq_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 VHADDT.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [arm_]vhaddq_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 VHADDT.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vhaddq_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 VHADDT.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [_arm_]vhaddq_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 VHADDT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vhaddq_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 VHADDT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vhaddq_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 VHADDT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [arm_]vhaddq_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 VHADDT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vhaddq_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 VHADDT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vhaddq_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 VHADDT.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vhaddq_x[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHADDT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vhaddq_x[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHADDT.S16 Qd,Qn,Rm	Qd -> result	MVE
$int32x4_t \ [_arm_] vhaddq_x [_n_s32] (int32x4_t \ a, int32_t \ b, mve_pred16_t \ p)$	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHADDT.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vhaddq_x[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHADDT.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vhaddq_x[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHADDT.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vhaddq_x[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHADDT.U32 Qd,Qn,Rm	Qd -> result	MVE
$ int8x16_t \ [_arm_] vhaddq_x \ [_s8] (int8x16_t \ a, int8x16_t \ b, mve_pred16_t \ p) $	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vhaddq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vhaddq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vhaddq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vhaddq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vhaddq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vhcaddq_rot90[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VHCADD.S8 Qd,Qn,Qm,#90	Qd -> result	MVE
int16x8_t [arm_]vhcaddq_rot90[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VHCADD.S16 Qd,Qn,Qm,#90	Qd -> result	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 Qd,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, mis2x4_t a, mis2x4_t b, mive_pred16_t p)	b -> Qm	VHCADDT.S32 Qd,Qn,Qm,#90		
:	p -> Rp	VMCD DO D.	0.1 >	MVE
int8x16_t [arm_]vhcaddq_rot90_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VHCADDT.S8 Qd,Qn,Qm,#90		
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
mi32x4_t b, mve_pred1o_t p)	p -> Rp	VHCADDT.S32 Qd,Qn,Qm,#90		
int8x16_t [arm_]vhcaddq_rot270[_s8](int8x16_t a,	a -> Qn	VHCADD.S8 Qd,Qn,Qm,#270	Qd -> result	MVE
int8x16_t b) int16x8_t [_arm_]vhcaddq_rot270[_s16](int16x8_t a,	b -> Qm a -> On	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	VPST		
	b -> Qm p -> Rp	VHCADDT.S8 Qd,Qn,Qm,#270		
int16x8_t [arm_]vhcaddq_rot270_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,#270		
	p -> Rp	VIICIBB 1.510 Qu,Qii,Qiii,ii270		
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
mactive, mi32x4_t a, mi32x4_t b, mive_pred10_t p)	b -> Qm	VHCADDT.S32 Qd,Qn,Qm,#270		
	p -> Rp		01 . 1	Mare
int8x16_t [arm_]vhcaddq_rot270_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VHCADDT.S8 Qd,Qn,Qm,#270		
int16x8_t [arm_]vhcaddq_rot270_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,#270		
int32x4_t [arm_]vhcaddq_rot270_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t b, inve_pred10_t p)	p -> Rp	VHCADDT.S32 Qd,Qn,Qm,#270		
int8x16_t [arm_]vhsubq[_n_s8](int8x16_t a, int8_t b)	a -> Qn	VHSUB.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vhsubq[_n_s16](int16x8_t a, int16_t b)	b -> Rm a -> Qn	VHSUB.S16 Qd,Qn,Rm	Qd -> result	MVE
	b->Rm	VIVIOUS 022 0 22 7		
int32x4_t [arm_]vhsubq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VHSUB.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vhsubq[_n_u8](uint8x16_t a, uint8_t	a -> Qn	VHSUB.U8 Qd,Qn,Rm	Qd -> result	MVE
b) uint16x8_t [arm_]vhsubq[_n_u16](uint16x8_t a,	b -> Rm a -> Qn	VHSUB.U16 Qd,Qn,Rm	Od -> result	MVE
uint16_t b)	b -> Rm		Ì	141 4 17
uint32x4_t [_arm_]vhsubq[_n_u32](uint32x4_t a,	a -> Qn	VHSUB.U32 Qd,Qn,Rm	Qd -> result	MVE
uint32_t b) int8x16_t [arm_]vhsubq[_s8](int8x16_t a, int8x16_t b)	b -> Rm a -> Qn	VHSUB.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
	b -> Qm			
int16x8_t [arm_]vhsubq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VHSUB.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vhsubq[_s32](int32x4_t a, int32x4_t b)	a -> Qn	VHSUB.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vhsubq[_u8](uint8x16_t a, uint8x16_t	b -> Qm a -> Qn	VHSUB.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
b)	b -> Qm	11130B.00 Qu,QII,QIII	Qu -> resuit	IVI V E/INEOIN
uint16x8_t [_arm_]vhsubq[_u16](uint16x8_t a,	a -> Qn	VHSUB.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t b) uint32x4_t [arm_]vhsubq[_u32](uint32x4_t a,	b -> Qm a -> Qn	VHSUB.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t b)	b -> Qm			

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 b)	b -> Rm p -> Rp a -> Qn	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	VPST VHSUBT.S8 Qd,Qn,Qm VMSR P0,Rp VPST	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, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHSUBT.U8 Od,On,Om	Qd -> result	MVE
uint16x8_t [arm_]vhsubq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHSUBT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [arm_]vhsubq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int8x16_t [arm_]vrhaddq[_s8](int8x16_t a, int8x16_t b)	p -> Rp a -> Qn b -> Om	VHSUBT.U32 Qd,Qn,Qm VRHADD.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
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 VRHADD.U8 Qd,Qn,Qm	Qd -> result Od -> result	MVE/NEON MVE/NEON
uint8x16_t [_arm_]vrhaddq[_u8](uint8x16_t a, uint8x16_t b) uint16x8_t [_arm_]vrhaddq[_u16](uint16x8_t a,	a -> Qn b -> Qm a -> Qn	VRHADD.U8 Qd,Qn,Qm VRHADD.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON MVE/NEON
uint16x8_t b) uint32x4_t [_arm_]vrhaddq[_u32](uint32x4_t a,	b -> Qm a -> Qn	VRHADD.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t b) int8x16_t [_arm_]vrhaddq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	b -> Qm inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRHADDT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vrhaddq_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 VRHADDT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vrhaddq_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 VRHADDT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vrhaddq_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 VRHADDT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vrhaddq_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 VRHADDT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vrhaddq_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 VRHADDT.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vrhaddq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRHADDT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vrhaddq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRHADDT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vrhaddq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRHADDT.S32 Od,On,Om	Qd -> result	MVE
uint8x16_t [_arm_]vrhaddq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRHADDT.U8 Od,On,Om	Qd -> result	MVE
uint16x8_t [_arm_]vrhaddq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRHADDT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vrhaddq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRHADDT.U32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vfmaq[_n_f16](float16x8_t a, float16x8_t b, float16_t c)	a -> Qda b -> Qn c -> Rm	VFMA.F16 Qda,Qn,Rm	Qda -> result	MVE/NEON
float32x4_t [arm_]vfmaq[_n_f32](float32x4_t a, float32x4_t b, float32_t c)	a -> Qda b -> Qn c -> Rm	VFMA.F32 Qda,Qn,Rm	Qda -> result	MVE/NEON
float16x8_t [arm_]vfmaq_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 VFMAT.F16 Qda,Qn,Rm	Qda -> result	MVE
float32x4_t [arm_]vfmaq_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 VFMAT.F32 Qda,Qn,Rm	Qda -> result	MVE

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 -> Om	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 p -> Rp	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)	a -> Qda b -> Qn c -> Qm p -> Rp	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)	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,	p -> Rp a -> Qn	VRMULHT.S16 Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p) uint8x16_t [arm_]vrmulhq_x[_u8](uint8x16_t a,	b -> Qm p -> Rp a -> Qn	VPST VRMULHT.S32 Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VRMULHT.U8 Qd,Qn,Qm	Qu -> resuit	IVIVE
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	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VRMULHT.U32 Qd,Qn,Qm		
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	VMSR P0,Rp VPST	Qd -> result	MVE
uint16x8_t [_arm_]vcvtaq_m[_u16_f16](uint16x8_t inactive, float16x8_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm	VCVTAT.S32.F32 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t [_arm_]vcvtaq_m[_u32_f32](uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm	VCVTAT.U16.F16 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t [arm_]vcvtaq_x_s16_f16(float16x8_t a,	a -> Qm p -> Rp a -> Qm	VCVTAT.U32.F32 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p) int32x4_t [arm_]vcvtaq_x_s32_f32(float32x4_t a,	p -> Rp a -> Qm	VPST VCVTAT.S16.F16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VCVTAT.S32.F32 Qd,Qm	Ì	·
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
int16v0 t [own lyoutes old f16(floot16v0 to)	a > 0m	VCVTN.S16.F16 Od,Om	Od > monule	MATERIEON
int16x8_t [arm_]vcvtnq_s16_f16(float16x8_t a) int32x4_t [arm_]vcvtnq_s32_f32(float32x4_t a)	a -> Qm	VCVTN.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
	a -> Qm	` ' \	Qd -> result Qd -> result	MVE/NEON MVE/NEON
uint16x8_t [_arm_]vcvtnq_u16_f16(float16x8_t a)	a -> Qm	VCVTN.U16.F16 Qd,Qm VCVTN.U32.F32 Qd,Qm		MVE/NEON MVE/NEON
uint32x4_t [_arm_]vcvtnq_u32_f32(float32x4_t a) int16x8_t [_arm_]vcvtnq_m[_s16_f16](int16x8_t inactive, float16x8_t a, mve_pred16_t p)	a -> Qm inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result Qd -> result	MVE/NEON MVE
	p -> Rp	VCVTNT.S16.F16 Qd,Qm		
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	VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t [_arm_]vcvtnq_m[_u32_f32](uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm	VCVTNT.U16.F16 Qd,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	Qd -> result	MVE
int32x4_t [_arm_]vcvtnq_x_s32_f32(float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VCVTNT.S16.F16 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
uint16x8_t [_arm_]vcvtnq_x_u16_f16(float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VCVTNT.S32.F32 Qd,Qm 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.U16.F16 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

matrixs. m_perque_as_25_200m325_41 a = 0 m	Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
inclose_1_mm_pevpg_ml_s3Politicitions_1	uint16x8_t [arm_]vcvtpq_u16_f16(float16x8_t a)	a -> Qm	VCVTP.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
		a -> Qm	VCVTP.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
Dock				Qd -> result	MVE
	inactive, float16x8_t a, mve_pred16_t p)	-			
mactive, floations_1_a, mve_predfo_1_p)	: .22 4 . F . 22 . MOV: .22 4 .			0.1	MATE
Dec Po Po Po Po Po Po Po P				Qd -> result	MVE
mint2s4_fl_arm_lvevtpq_mf_u32_f32[cloid32s4_t] mintve > Qd MSR PQRp Qd > result MVE mintve > Qd mintve, flootif5s4_fl_arm_lvevtpq_mf_u32_f32[cloid32s4_t] mintve > Qd MSR PQRp Qd > result MVE mintve > Qd MSR PQRp Qd > result MVE mintve > Qd MSR PQRp Qd > result MVE MVE mintve > Qd MSR PQRp Qd > result MVE MVE mintve > Qd MSR PQRp Qd > result MVE M	mactive, moats2x4_t a, mve_pred16_t p)	•			
mactive float fock a, mee_pred 6, p p p N P N P N P N P N P N P N P N P N P N P N P N P N N	uint16x8 t [arm]vcvtpq m[u16 f16](uint16x8 t			Od -> result	MVE
p > Rp		,		Qu' > Tesun	111.12
inactive_float224_1_a_mm_vevtpq_x_s16_flofthoat16x8_1 a,			VCVTPT.U16.F16 Qd,Qm		
P				Qd -> result	MVE
IntloSe_t1_arm_pvvvpq_x_s16_f16(float16s8_t a, p > Qm	inactive, float32x4_t a, mve_pred16_t p)				
more	16 0 15 1 16 0 16 0 1			0.1 1:	MATE
CVCYPTS13EF16 QLQm				Qu -> result	MVE
int254_tamlevernq_x_s12_f32(lont254_t)	mive_predio_t p)	p -> Kp	1 12		
ave_pred16_t_p p>-Rp	int32x4 t [arm]vcvtpq x s32 f32(float32x4 t a.	a -> Om		Od -> result	MVE
a > Qm					
Description			VCVTPT.S32.F32 Qd,Qm		
		a -> Qm		Qd -> result	MVE
a > Qn	mve_pred16_t p)	p -> Rp			
mve_pred16_t.p p > Rp	20 00 1			0.1) a m
Note				Qd -> result	MVE
Initiask_t_arm_lvevtung_sif_file(filoatifask_ta) a > Qm	mve_pred16_t p)	p -> Kp			
	int16x8 t [arm]vcvtmq s16 f16(float16x8 t a)	a -> Om		Od -> result	MVE/NEON
				_ `	
mactive, float16x8_t a, mve_pred16_t p		_	VCVTM.U32.F32 Qd,Qm		
	int16x8_t [arm_]vcvtmq_m[_s16_f16](int16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
Int32x4_1 _ arm_ vevtmq_m[_s32_[32](int32x4_1 anactive > Od nactive, float32x4_1 a, mve_pred16_1 p)	inactive, float16x8_t a, mve_pred16_t p)				
nactive, float32x4_t a, mve_pred16_t p)					
p > Rp	int32x4_t [_arm_]vcvtmq_m[_s32_f32](int32x4_t			Qd -> result	MVE
uint16x8_t _arm_ vevtmq_m[_u16_t16 (uint16x8_t a	inactive, float32x4_t a, mve_pred16_t p)				
inactive, float16x8_t a, mve_pred16_t p)	uint16v8 t [arm]vevtma m[u16 f16](uint16v8 t			Od -> result	MVF
D				Qu -> resurt	WYL
inactive, float32x4_t a, mve_pred16_tp)	annual style commons _r m, nave_process_r p/	•			
p -> Rp	uint32x4_t [arm_]vcvtmq_m[_u32_f32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
Intl6x8_t	inactive, float32x4_t a, mve_pred16_t p)	•			
mve_pred16_t p)					
Int32x4_t [_arm_]vcvtmq_x_s32_f32(float32x4_t a, mve_pred16_t p)		•		Qd -> result	MVE
int32x4_t	mve_pred16_t p)	p -> Kp	1 12		
mve_pred16_t p	int32x4 t [arm]vevtmq x s32 f32(float32x4 t a	a -> Om		Od -> result	MVF
WCVTMT.S32.F32 Qd,Qm				Qu > resurt	III V E
mve_pred16_t p)			VCVTMT.S32.F32 Qd,Qm		
VCVTMT.U16.F16 Qd,Qm	uint16x8_t [arm_]vcvtmq_x_u16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t [_arm_]vcvtmq_x_u32_f32(float32x4_t a, mve_pred16_t p)	mve_pred16_t p)	p -> Rp			
mve_pred16_t p)	20 00 00 00 00 00 00 00 00 00 00 00 00 0	^		0.1	Nam.
Section		-		Qd -> result	MVE
Float16x8_t [_arm_]vcvtbq_f16_f32(float16x8_t a, float32x4_t b)	mvc_preuro_t p)	h -> Kh		1	
float32x4_t b b -> Qm	float16x8 t [arm]vcvtbq f16 f32(float16x8 t a.	a -> Od		Od -> result	MVE
Float32x4_t [_arm_]vcvtbq_f32_f16(float16x8_t a) a -> Qm			2	Z= 1.00m	=
float32x4_t b, mve_pred16_t p)	float32x4_t [arm_]vcvtbq_f32_f16(float16x8_t a)	_	VCVTB.F32.F16 Qd,Qm	Qd -> result	MVE
p -> Rp				Qd -> result	MVE
float32x4_t [_arm_]vcvtbq_m_f32_f16(float32x4_t inactive -> Qd a -> Qm VPST VCVTBT.F32.F16 Qd,Qm Qd -> result MVE	float32x4_t b, mve_pred16_t p)	-		1	
inactive, float16x8_t a, mve_pred16_t p)	G 22 4 4 5 1 4 22 21 22 2		2	01	NOTE
p -> Rp				Qd -> result	MVE
Float32x4_t [_arm_]vcvtbq_x_f32_f16(float16x8_t a, mve_pred16_t p)	mactive, noatroxo_t a, mve_pred16_t p)	•		1	
mve_pred16_t p) p -> Rp VPST VCVTBT.F32.F16 Qd,Qm float16x8_t [_arm_]vcvttq_f16_f32(float16x8_t a, float32x4_t b) a -> Qd b -> Qm VCVTT.F16.F32 Qd,Qm Qd -> result MVE float32x4_t b a -> Qm VCVTT.F32.F16 Qd,Qm Qd -> result MVE float16x8_t [_arm_]vcvttq_f32_f16(float16x8_t a, float32x4_t b, mve_pred16_t p) a -> Qd b -> Qm VMSR PO,Rp VPST Qd -> result MVE float32x4_t [_arm_]vcvttq_m_f32_f16(float32x4_t inactive -> Qd VMSR PO,Rp VPST Qd -> result MVE float32x4_t [_arm_]vcvttq_m_f32_f16(float32x4_t inactive -> Qd VMSR PO,Rp VPST Qd -> result MVE	float32x4 t [arm]vcvtbg x f32 f16(float16x8 ta			Od -> result	MVE
VCVTBT.F32.F16 Qd,Qm	· - · - ·	-			. =
float32x4_t b) b -> Qm	·		VCVTBT.F32.F16 Qd,Qm		
Float32x4_t			VCVTT.F16.F32 Qd,Qm	Qd -> result	MVE
float16x8_t [_arm_]vcvttq_m_f16_f32(float16x8_t a, float32x4_t b, mve_pred16_t p)					
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$					
p -> Rp VCVTTT.F16.F32 Qd,Qm		-		Qd -> result	MVE
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) VPST Qd -> result MVE	Hoat32x4_t b, mve_pred16_t p)			1	
inactive, float16x8_t a, mve_pred16_t p) a -> Qm VPST	float32x4 t [arm lyevtta m f32 f16(float32x4 +			Od -> result	MVF
		-		Qu -> Icsuit	141 4 1
$p \rightarrow \kappa p = VCV111.F32.F16 Vd,Vm = I$		p -> Rp	VCVTTT.F32.F16 Qd,Qm	1	

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [arm_]vcvttq_x_f32_f16(float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
float16x8_t [arm_]vcvtq[_f16_s16](int16x8_t a)	0.5000	VCVTTT.F32.F16 Qd,Qm	Od > monule	MVENEON
float16x8_t [_arm_]vcvtq[_f16_s16](int16x8_t a) float16x8_t [_arm_]vcvtq[_f16_u16](uint16x8_t a)	a -> Qm a -> Qm	VCVT.F16.S16 Qd,Qm VCVT.F16.U16 Qd,Qm	Qd -> result Od -> result	MVE/NEON MVE/NEON
float32x4_t [arm_]vcvtq[_f32_s32](int32x4_t a)	a -> Qm	VCVT.F32.S32 Qd,Qm	Od -> result	MVE/NEON
float32x4_t [arm_]vcvtq[_f32_u32](uint32x4_t a)	a -> Qm	VCVT.F32.U32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [_arm_]vcvtq_m[_f16_s16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VCVTT.F16.S16 Qd,Qm		
float16x8_t [_arm_]vcvtq_m[_f16_u16](float16x8_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.F16.U16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vcvtq_m[_f32_s32](float32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.F32.S32 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vcvtq_m[_f32_u32](float32x4_t inactive, uint32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.F32.U32 Qd,Qm	Qd -> result	MVE
float16x8_t [_arm_]vcvtq_x[_f16_u16](uint16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VCVTT.F16.U16 Qd,Qm	Qu > resuit	III V E
float16x8_t [_arm_]vcvtq_x[_f16_s16](int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.F16.S16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vcvtq_x[_f32_s32](int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.F32.S32 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vcvtq_x[_f32_u32](uint32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.F32.U32 Qd,Qm	Qd -> result	MVE
float16x8_t [_arm_]vcvtq_n[_f16_s16](int16x8_t a, const int imm6)	a -> Qm 1 <= imm6 <= 16	VCVT.F16.S16 Qd,Qm,imm6	Qd -> result	MVE/NEON
float16x8_t [_arm_]vcvtq_n[_f16_u16](uint16x8_t a, const int imm6)	a -> Qm 1 <= imm6 <= 16	VCVT.F16.U16 Qd,Qm,imm6	Qd -> result	MVE/NEON
float32x4_t [_arm_]vcvtq_n[_f32_s32](int32x4_t a, const int imm6)	a -> Qm 1 <= imm6 <= 32	VCVT.F32.S32 Qd,Qm,imm6	Qd -> result	MVE/NEON
float32x4_t [_arm_]vcvtq_n[_f32_u32](uint32x4_t a, const int imm6)	a -> Qm 1 <= imm6 <= 32	VCVT.F32.U32 Qd,Qm,imm6	Qd -> result	MVE/NEON
float16x8_t [_arm_]vcvtq_m_n[_f16_s16](float16x8_t inactive, int16x8_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm6 <= 16 p -> Rp	VMSR P0,Rp VPST VCVTT.F16.S16 Qd,Qm,imm6	Qd -> result	MVE
float16x8_t [_arm_]vcvtq_m_n[f16_u16](float16x8_t inactive, uint16x8_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm6 <= 16 p -> Rp	VMSR P0,Rp VPST VCVTT.F16.U16 Qd,Qm,imm6	Qd -> result	MVE
float32x4_t [_arm_]vcvtq_m_n[_f32_s32](float32x4_t inactive, int32x4_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm6 <= 32 p -> Rp	VMSR P0.Rp VPST VCVTT.F32.S32 Qd,Qm,imm6	Qd -> result	MVE
float32x4_t [_arm_]vcvtq_m_n[_f32_u32](float32x4_t inactive, uint32x4_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm6 <= 32 p -> Rp	VMSR P0.Rp VPST VCVTT.F32.U32 Qd,Qm,imm6	Qd -> result	MVE
float16x8_t [_arm_]vcvtq_x_n[_f16_s16](int16x8_t a, const int imm6, mve_pred16_t p)	a -> Qm 1 <= imm6 <= 16 p -> Rp	VMSR P0,Rp VPST VCVTT.F16.S16 Qd,Qm,imm6	Qd -> result	MVE
float16x8_t [_arm_]vcvtq_x_n[_f16_u16](uint16x8_t a, const int imm6, mve_pred16_t p)	a -> Qm 1 <= imm6 <= 16 p -> Rp	VMSR P0,Rp VPST VCVTT.F16.U16 Qd,Qm,imm6	Qd -> result	MVE
float32x4_t [_arm_]vcvtq_x_n[_f32_s32](int32x4_t a, const int imm6, mve_pred16_t p)	a -> Qm 1 <= imm6 <= 32 p -> Rp	VMSR P0,Rp VPST VCVTT.F32.S32 Qd,Qm,imm6	Qd -> result	MVE

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	VPST VCVTT.F32.U32 Qd,Qm,imm6		
	p -> Rp			
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) int16x8_t [_arm_]vcvtq_m[_s16_f16](int16x8_t inactive,	a -> Qm inactive -> Qd	VCVT.U32.F32 Qd,Qm VMSR P0,Rp	Qd -> result Qd -> result	MVE/NEON MVE
float16x8_t a, mve_pred16_t p)	a -> Qm	VMSK PO,KP VPST	Qu -> resuit	NIVE
noutrono_t u, m/o_preuro_t p/	p -> Rp	VCVTT.S16.F16 Qd,Qm		
int32x4_t [_arm_]vcvtq_m[_s32_f32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCVTT.S32.F32 Qd,Qm		
uint16x8_t [_arm_]vcvtq_m[_u16_f16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VCVTT.U16.F16 Qd,Qm		
uint32x4_t [arm_]vcvtq_m[_u32_f32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Qu => result	WIVE
macuve, noaco za i_t a, m ve_prod ro_t p)	p -> Rp	VCVTT.U32.F32 Qd,Qm		
int16x8_t [arm_]vcvtq_x_s16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	`	
		VCVTT.S16.F16 Qd,Qm		
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) a m
uint16x8_t [arm_]vcvtq_x_u16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VCVTT.U16.F16 Qd,Qm		
uint32x4_t [arm_]vcvtq_x_u32_f32(float32x4_t a,	a -> Om	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu -> result	MVL
	r ·r	VCVTT.U32.F32 Qd,Qm		
int16x8_t [arm_]vcvtq_n_s16_f16(float16x8_t a, const	a -> Qm	VCVT.S16.F16 Qd,Qm,imm6	Qd -> result	MVE/NEON
int imm6)	1 <= imm6 <=			
	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 <=			
uint16x8_t [arm_]vcvtq_n_u16_f16(float16x8_t a, const	32 a -> Qm	VCVT.U16.F16 Qd,Qm,imm6	Qd -> result	MVE/NEON
int imm6)	1 <= imm6 <=	VCV1.016.F16 Qa,Qili,lilililib	Qu -> resuit	M V E/INEOIN
int inimio)	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 <=			
	32			
int16x8_t [arm_]vcvtq_m_n[_s16_f16](int16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, const int imm6, mve_pred16_t p)	a -> Qm	VPST		
	1 <= imm6 <=	VCVTT.S16.F16 Qd,Qm,imm6		
	16			
int32x4_t [arm_]vcvtq_m_n[_s32_f32](int32x4_t	p -> Rp inactive -> Od	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, const int imm6, mve_pred16_t p)	a -> Qm	VPST	Qu => result	WIVE
mactive, noats2x+_t a, const int immo, inve_prearo_t p)	1 <= imm6 <=	VCVTT.S32.F32 Qd,Qm,imm6		
	32	(4,7,0)		
	p -> Rp			
uint16x8_t [arm_]vcvtq_m_n[_u16_f16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, const int imm6, mve_pred16_t p)	a -> Qm	VPST		
	1 <= imm6 <=	VCVTT.U16.F16 Qd,Qm,imm6		
	16			
uint32x4_t [arm_]vcvtq_m_n[_u32_f32](uint32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, const int imm6, mve_pred16_t p)	a -> Om	VMSK FO,KP VPST	Qu -> resuit	MIVE
mactive, notice 2241_t at, const in mino, inve_prearo_t p)	1 <= imm6 <=	VCVTT.U32.F32 Qd,Qm,imm6		
	32	2.5.2.2.2.2.2.2.3.2,		
	p -> Rp			
int16x8_t [arm_]vcvtq_x_n_s16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
const int imm6 myo prod16 +=)		LVDCT		
const int imm6, mve_pred16_t p)	1 <= imm6 <=	VPST		
const int ininio, inve_pred to_t p)	1 <= imm6 <= 16	VCVTT.S16.F16 Qd,Qm,imm6		
	1 <= imm6 <= 16 p -> Rp	VCVTT.S16.F16 Qd,Qm,imm6) Arm
int32x4_t [arm_]vcvtq_x_n_s32_f32(float32x4_t a,	1 <= imm6 <= 16 p -> Rp a -> Qm	VCVTT.S16.F16 Qd,Qm,imm6 VMSR P0,Rp	Qd -> result	MVE
	1 <= imm6 <= 16 p -> Rp a -> Qm 1 <= imm6 <=	VCVTT.S16.F16 Qd,Qm,imm6 VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t [arm_]vcvtq_x_n_s32_f32(float32x4_t a,	1 <= imm6 <= 16 p -> Rp a -> Qm 1 <= imm6 <= 32	VCVTT.S16.F16 Qd,Qm,imm6 VMSR P0,Rp	Qd -> result	MVE
int32x4_t [_arm_]vcvtq_x_n_s32_f32(float32x4_t a, const int imm6, mve_pred16_t p)	1 <= imm6 <= 16 p -> Rp a -> Qm 1 <= imm6 <= 32 p -> Rp	VCVTT.S16.F16 Qd,Qm,imm6 VMSR P0,Rp VPST VCVTT.S32.F32 Qd,Qm,imm6		
int32x4_t [arm_]vcvtq_x_n_s32_f32(float32x4_t a,	1 <= imm6 <= 16 p -> Rp a -> Qm 1 <= imm6 <= 32	VCVTT.S16.F16 Qd,Qm,imm6 VMSR P0,Rp VPST	Qd -> result Qd -> result	MVE MVE
int32x4_t [_arm_]vcvtq_x_n_s32_f32(float32x4_t a, const int imm6, mve_pred16_t p) uint16x8_t [_arm_]vcvtq_x_n_u16_f16(float16x8_t a,	1 <= imm6 <= 16 p -> Rp a -> Qm 1 <= imm6 <= 32 p -> Rp a -> Qm	VCVTT.S16.F16 Qd,Qm,imm6 VMSR P0,Rp VPST VCVTT.S32.F32 Qd,Qm,imm6 VMSR P0,Rp		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [arm_]vcvtq_x_n_u32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
const int imm6, mve_pred16_t p)	1 <= imm6 <= 32	VPST VCVTT.U32.F32 Qd,Qm,imm6		
	p -> Rp			
float16x8_t [_arm_]vrndq[_f16](float16x8_t a)	a -> Qm	VRINTZ.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vrndq[_f32](float32x4_t a)	a -> Qm	VRINTZ.F32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vrndq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
noacrox8_t a, nive_pred10_t p)	p -> Rp	VRINTZT.F16 Qd,Qm		
float32x4_t [arm_]vrndq_m[_f32](float32x4_t inactive,	inactive -> Od	VMSR P0,Rp	Od -> result	MVE
float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VRINTZT.F32 Qd,Qm		
float16x8_t [arm_]vrndq_x[_f16](float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
float32x4_t [arm_]vrndq_x[_f32](float32x4_t a,	a -> Om	VRINTZT.F16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VMSK FU,KP VPST	Qu -> resuit	MIVE
inve_pred10_t p)	p -> Kp	VRINTZT.F32 Qd,Qm		
float16x8_t [arm_]vrndnq[_f16](float16x8_t a)	a -> Qm	VRINTN.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vrndnq[_f32](float32x4_t a)	a -> Qm	VRINTN.F32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vrndnq_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	VRINTNT.F16 Qd,Qm		
float32x4_t [arm_]vrndnq_m[_f32](float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
float16x8 t [arm]vrndnq x[f16](float16x8 t a,	p -> Rp a -> Qm	VRINTNT.F32 Qd,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VMSK PO,KP VPST	Qu -> resuit	IVI V E
inve_pred10_t p)	p -> Kp	VRINTNT.F16 Qd,Qm		
float32x4 t [arm]vrndnq x[f32](float32x4 t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VRINTNT.F32 Qd,Qm		
float16x8_t [arm_]vrndmq[_f16](float16x8_t a)	a -> Qm	VRINTM.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [arm_]vrndmq[_f32](float32x4_t a)	a -> Qm	VRINTM.F32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vrndmq_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_]vrndmq_m[_f32](float32x4_t	p -> Rp inactive -> Od	VRINTMT.F16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VRINTMT.F32 Qd,Qm	Qu -> resuit	WIVE
float16x8_t [arm_]vrndmq_x[_f16](float16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VRINTMT.F16 Qd,Qm		
float32x4_t [arm_]vrndmq_x[_f32](float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
	_	VRINTMT.F32 Qd,Qm		
float16x8_t [_arm_]vrndpq[_f16](float16x8_t a)	a -> Qm	VRINTP.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [arm_]vrndpq[_f32](float32x4_t a)	a -> Qm	VRINTP.F32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vrndpq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, noatroxo_t a, mve_picuro_t p)	n -> Rn	VRINTPT.F16 Od.Om		
float32x4_t [arm_]vrndpq_m[_f32](float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VRINTPT.F32 Qd,Qm		
float16x8_t [arm_]vrndpq_x[_f16](float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
G (22.4) I I I MOVG (22.4)	. 0	VRINTPT.F16 Qd,Qm	0.1 1:	MATE
float32x4_t [arm_]vrndpq_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
mvc_picuro_t p)	p -> Kp	VRINTPT.F32 Qd,Qm		
float16x8_t [arm_]vrndaq[_f16](float16x8_t a)	a -> Qm	VRINTA.F16 Qd,Qm	Qd -> result	MVE
float32x4 t [arm]vrndaq[f32](float32x4 t a)	a -> Qm	VRINTA.F32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vrndaq_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	VRINTAT.F16 Qd,Qm	1	
float32x4_t [arm_]vrndaq_m[_f32](float32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
floation of complements of flores are a	p -> Rp	VRINTAT.F32 Qd,Qm	04 5 10	MVE
float16x8_t [arm_]vrndaq_x[_f16](float16x8_t a,	a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VRINTAT.F16 Qd,Qm		
float32x4_t [arm_]vrndaq_x[_f32](float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		1
	1 1	VRINTAT.F32 Qd,Qm	<u> </u>	
float16x8_t [arm_]vrndxq[_f16](float16x8_t a)	a -> Qm	VRINTX.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [arm_]vrndxq[_f32](float32x4_t a)	a -> Qm	VRINTX.F32 Qd,Qm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [arm_]vrndxq_m[_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VRINTXT.F16 Od,Om		
float32x4_t [arm_]vrndxq_m[_f32](float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
float16x8_t [arm_]vrndxq_x[_f16](float16x8_t a,	p -> Rp a -> Qm	VRINTXT.F32 Qd,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Ç	
float32x4 t [arm]vrndxq x[f32](float32x4 t a,	a -> Qm	VRINTXT.F16 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu -> resuit	WIVE
int8x16_t [arm_]vandq[_s8](int8x16_t a, int8x16_t b)	0	VRINTXT.F32 Qd,Qm VAND Qd,Qn,Qm	Od -> result	MVE/NEON
intex10_t [arm_jvandq[_s8](intex10_t a, intex10_t b)	a -> Qn b -> Qm	VAND Qu,Qii,Qiii	Qu -> resuit	WIVE/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	a -> Qn	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
b) uint16x8 t [arm]vandq[u16](uint16x8 t a, uint16x8 t	b -> Qm a -> Qn	VAND Qd,Qn,Qm	Od -> result	MVE/NEON
b)	b -> Qm	VAND Qu,Qii,Qiii	Qu -> result	
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,	a -> Qn	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t b) float32x4 t [arm]vandq[f32](float32x4 t a,	b -> Qm a -> On	VAND Qd,Qn,Qm	Od -> result	MVE/NEON
float32x4_t b)	b -> Qm	7 . 7 . 7		
int8x16_t [_arm_]vandq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
moxro_t u, moxro_t o, mve_preuro_t p)	b -> Qm	VANDT Qd,Qn,Qm		
int16x8_t [arm_]vandq_m[_s16](int16x8_t inactive,	p -> Rp inactive -> Od	VMSR P0,Rp	Od -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qu	VMSK PO,KP VPST	Qu -> resuit	WIVE
	b -> Qm	VANDT Qd,Qn,Qm		
int32x4_t [_arm_]vandq_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 -> Qn	VPST		
	b -> Qm p -> Rp	VANDT Qd,Qn,Qm		
uint8x16_t [arm_]vandq_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 b -> Qm	VPST VANDT Qd,Qn,Qm		
	p -> Rp	2 . 2 . 2		
uint16x8_t [arm_]vandq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
unicroxo_t a, unicroxo_t o, inve_predio_t p)	b -> Qm	VANDT Qd,Qn,Qm		
uint32x4_t [_arm_]vandq_m[_u32](uint32x4_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
uint32x4_t [atm_]vandq_m[_u32](uint32x4_t mactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn	VMSK FO,Kp VPST	Qu -> result	WIVE
	b -> Qm	VANDT Qd,Qn,Qm		
float16x8_t [arm_]vandq_m[_f16](float16x8_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VANDT Qd,Qn,Qm		
float32x4_t [_arm_]vandq_m[_f32](float32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VANDT Qd,Qn,Qm		
	p -> Rp	2 . 2 . 2		
int8x16_t [arm_]vandq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VANDT Qd,Qn,Qm		
int16x8_t [arm_]vandq_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	VANDT Qd,Qn,Qm		
int32x4_t [arm_]vandq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
· -• -• -• -• -• -• -• -• -• -• -• -• -•	p -> QIII p -> Rp	VANDT Qd,Qn,Qm		
uint8x16_t [_arm_]vandq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VANDT Qd,Qn,Qm		
uint16x8_t [_arm_]vandq_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 VANDT Qd,Qn,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [arm_]vandq_x[_u32](uint32x4_t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Qu y rosun	111112
float16x8_t [arm_]vandq_x[_f16](float16x8_t a,	p -> Rp a -> On	VANDT Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VMSK PO,KP VPST	Qu -> resuit	NIVE
noutrono_t o, mre_preuro_t p/	p -> Rp	VANDT Qd,Qn,Qm		
float32x4_t [arm_]vandq_x[_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VANDT Qd,Qn,Qm	01	MUTATION
int8x16_t [arm_]vbicq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Om	VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vbicq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vbicq[_s32](int32x4_t a, int32x4_t b)	b -> Qm a -> Qn	VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vbicq[_u8](uint8x16_t a, uint8x16_t	b -> Qm a -> Qn	VBIC Qd,Qn,Qm	Od -> result	MVE/NEON
b)	b -> Qm	V DIC Qu,QII,QIII	Qu > resun	W E TEOT
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	a -> Qn	VBIC Qd,Qn,Qm	Qd -> result	MVE/NEON
b) float16x8_t [arm_]vbicq[_f16](float16x8_t a,	b -> Qm a -> Qn	VBIC Qd,Qn,Qm	Od -> result	MVE/NEON
float16x8_t b)	b -> Qm	2 . 2 . 2	`	
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,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VBICT Qd,Qn,Qm		
int16x8_t [arm_]vbicq_m[_s16](int16x8_t inactive,	inactive -> Od	VMSR P0,Rp	Od -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST	(a : 110mm	
	b -> Qm	VBICT Qd,Qn,Qm		
int32x4_t [arm_]vbicq_m[_s32](int32x4_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t [atmjvbleq_ml_s52](mt52x4_t mactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> On	VPST VPST	Qu -> resuit	MIVE
	b -> Qm	VBICT Qd,Qn,Qm		
	p -> Rp			
uint8x16_t [arm_]vbicq_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 b -> Qm	VPST VBICT Qd,Qn,Qm		
	p -> Rp	VBICT Qu,QII,QIII		
uint16x8_t [arm_]vbicq_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	VBICT Qd,Qn,Qm		
uint32x4 t [arm]ybicq m[u32](uint32x4 t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn	VPST	(a : 110mm	
	b -> Qm	VBICT Qd,Qn,Qm		
float16x8_t [arm_]vbicq_m[_f16](float16x8_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn	VPST	Qu -> resuit	MIVE
	b -> Qm	VBICT Qd,Qn,Qm		
	p -> Rp			
float32x4_t [_arm_]vbicq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
noat32x4_t a, noat32x4_t b, nive_pred16_t p)	b -> Qm	VBICT Qd,Qn,Qm		
	p -> Rp			
int8x16_t [arm_]vbicq_x[_s8](int8x16_t a, int8x16_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Qm	VPST		
int16x8_t [arm_]vbicq_x[_s16](int16x8_t a, int16x8_t	p -> Rp a -> Qn	VBICT Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
b, mve pred16 t p)	b -> Qm	VMSK FO,KP VPST	Qu -> icsuit	141 4 12
, 4 -1/	p -> Rp	VBICT Qd,Qn,Qm		
int32x4_t [arm_]vbicq_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 VBICT Qd,Qn,Qm		
uint8x16_t [arm_]vbicq_x[_u8](uint8x16_t a,	p -> Kp a -> On	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST	Qu -> resuit	
	p -> Rp	VBICT Qd,Qn,Qm		
uint16x8_t [arm_]vbicq_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 VBICT Qd,Qn,Qm		
uint32x4_t [arm_]vbicq_x[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
·	p -> Rp	VBICT Qd,Qn,Qm		

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' > Tesun	11112
116016	p -> Rp	VBICT Qd,Qn,Qm	0.1) am
int16x8_t [arm_]vbicq[_n_s16](int16x8_t a, const int16_t imm)	a -> Qda imm in	VBIC.I16 Qda,#imm	Qda -> result	MVE
	AdvSIMDExpa			
	ndImm	AMBAG YAO O L III	0.1) am
nt32x4_t [arm_]vbicq[_n_s32](int32x4_t a, const nt32_t imm)	a -> Qda imm in	VBIC.I32 Qda,#imm	Qda -> result	MVE
	AdvSIMDExpa			
	ndImm	***************************************		1
uint16x8_t [arm_]vbicq[_n_u16](uint16x8_t a, const uint16_t imm)	a -> Qda imm in	VBIC.I16 Qda,#imm	Qda -> result	MVE
untro_t mmi)	AdvSIMDExpa			
	ndImm			
uint32x4_t [_arm_]vbicq[_n_u32](uint32x4_t a, const uint32_t imm)	a -> Qda imm in	VBIC.I32 Qda,#imm	Qda -> result	MVE
umc32_t mmn)	AdvSIMDExpa			
	ndImm			
int16x8_t [arm_]vbicq_m_n[_s16](int16x8_t a, const int16_t imm, mve_pred16_t p)	a -> Qda imm in	VMSR P0,Rp VPST	Qda -> result	MVE
mero_t mini, mve_predro_t p)	AdvSIMDExpa	VBICT.I16 Qda,#imm		
	ndImm			
int32x4_t [arm_]vbicq_m_n[_s32](int32x4_t a, const	p -> Rp a -> Oda	VMSR P0,Rp	Oda -> result	MVE
int32x4_t [arm_jvoicq_m_n[_s52](int32x4_t a, const int32_t imm, mve_pred16_t p)	imm in	VMSK PO,KP VPST	Qua -> resuit	MVE
, , , , , , , , , , , , , , , , , , ,	AdvSIMDExpa	VBICT.I32 Qda,#imm		
	ndImm			
uint16x8_t [arm_]vbicq_m_n[_u16](uint16x8_t a, const	p -> Rp a -> Oda	VMSR P0,Rp	Qda -> result	MVE
uint16_t imm, mve_pred16_t p)	imm in	VPST	V	1
	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 AdvSIMDExpa	VPST VBICT.I32 Qda,#imm		
	ndImm	VBIC1.132 Qua,#IIIIIII		
	p -> Rp			
int8x16_t [arm_]vbrsrq[_n_s8](int8x16_t a, int32_t b)	a -> Qn b -> Rm	VBRSR.8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vbrsrq[_n_s16](int16x8_t a, int32_t b)	a -> On	VBRSR.16 Qd,Qn,Rm	Qd -> result	MVE
	b -> Rm			
int32x4_t [arm_]vbrsrq[_n_s32](int32x4_t a, int32_t b)	a -> Qn	VBRSR.32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vbrsrq[_n_u8](uint8x16_t a, int32_t	b -> Rm a -> Qn	VBRSR.8 Od,On,Rm	Qd -> result	MVE
b)	b -> Rm		Q	
uint16x8_t [arm_]vbrsrq[_n_u16](uint16x8_t a, int32_t	a -> Qn	VBRSR.16 Qd,Qn,Rm	Qd -> result	MVE
b) uint32x4_t [arm_]vbrsrq[_n_u32](uint32x4_t a, int32_t	b -> Rm a -> Qn	VBRSR.32 Qd,Qn,Rm	Qd -> result	MVE
b)	b -> Rm	2,2,		·
float16x8_t [arm_]vbrsrq[_n_f16](float16x8_t a, int32_t	a -> Qn	VBRSR.16 Qd,Qn,Rm	Qd -> result	MVE
float32x4 t [arm]vbrsrq[n f32](float32x4 t a, int32 t	b -> Rm a -> Qn	VBRSR.32 Qd,Qn,Rm	Od -> result	MVE
b)	b -> Rm	2,2,		
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 -> Rp			
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 b -> Rm	VPST VBRSRT.16 Qd,Qn,Rm		
	p -> Rp	. DROKT.TO QU,QII,KIII		
int32x4_t [_arm_]vbrsrq_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	VPST VBRSRT.32 Qd,Qn,Rm		
	p -> Rm	TENSICI .52 QU,QII,KIII		
uint8x16_t [arm_]vbrsrq_m[_n_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qn	VPST VBRSRT.8 Qd,Qn,Rm		1
- · · · ·	b -> Rm			

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 p -> Rp	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)	a -> Qn b -> Qm p -> Rp	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) uint32x4 t[_arm_]veorq_x[_u32](uint32x4_t a,	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm VMSR P0,Rp	Qd -> result Od -> result	MVE MVE
uint32x4_t [arm_]veorq_x[_us2](uint32x4_t a, uint32x4_t b, mve_pred16_t p) float16x8_t [arm_]veorq_x[_f16](float16x8_t a,	a -> Qn b -> Qm p -> Rp a -> Qn	VMSR PO,RP VPST VEORT Qd,Qn,Qm VMSR P0,Rp	Qd -> result Od -> result	MVE
float16x8_t b, mve_pred16_t p) float32x4_t [_arm_]veorq_x[_f32](float32x4_t a,	b -> Qm p -> Rp a -> On	VMSR PO,RP VPST VEORT Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
float32x4_t b, mve_pred16_t p) int16x8_t [arm\vmovlbq[_s8](int8x16_t a)	b -> Qm p -> Rp a -> Qm	VMSK FO,RP VPST VEORT Qd,Qn,Qm VMOVLB.S8 Qd,Qm	Qd -> result	MVE
int32x4_t [arm_]vmovlbq[_s16](int16x8_t a)	a -> Qm	VMOVLB.S16 Qd,Qm	Od -> result	MVE
uint16x8_t [arm_]vmovlbq[_u8](uint8x16_t a)	a -> Qm	VMOVLB.U8 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmovlbq[_u16](uint16x8_t a) int16x8_t [_arm_]vmovlbq_m[_s8](int16x8_t inactive, int8x16_t a, mve_pred16_t p)	a -> Qm inactive -> Qd a -> Om	VMOVLB.U16 Qd,Qm VMSR P0,Rp VPST	Qd -> result Qd -> result	MVE MVE
int32x4_t [_arm_]vmovlbq_m[_s16](int32x4_t inactive, int16x8_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm	VMOVLBT.S8 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
uint16x8_t [_arm_]vmovlbq_m[_u8](uint16x8_t inactive, uint8x16_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm	VMOVLBT.S16 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t [_arm_]vmovlbq_m[_u16](uint32x4_t inactive, uint16x8_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm p -> Rp	VMOVLBT.U8 Qd,Qm 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 Od,Om	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 Qd,Qm	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 p -> Rp	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)	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 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 p -> Rp	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)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int32x4_t [arm_]vmvnq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	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 Qd,Qm	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)	$\begin{array}{l} a -> Qm \\ p -> Rp \end{array}$	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

BRNS 6.2	Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
move_predic_t_p p	int8x16_t [arm_]vpselq[_s8](int8x16_t a, int8x16_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
		`	VPSEL Qd,Qn,Qm		
Doc	1.150.150.150.150.1150.1		VII (OD DO D	0.1) am
Description Part				Qd -> result	MVE
mi324_1_mm_lynelg[_s32](mi324_1 a, ini324_1 b)	mive_pred10_t p)		VFSEL Qu,Qii,Qiii		
December	int32x4 t [arm]vpselq[s32](int32x4 t a, int32x4 t b,		VMSR P0,Rp	Od -> result	MVE
min452.f_l_arm_lypselg_is4 tim6452_t a, init645_t b		b -> Qm		`	
Description					
		-		Qd -> result	MVE
minst	mve_pred16_t p)		VPSEL Qd,Qn,Qm		
b. me_pred16_tp b. > Qm	uint8x16 t [arm]vpselq[u8](uint8x16 t a, uint8x16 t		VMSR P0.Rp	Od -> result	MVE
Mint					
wint584_t am ypecia[_u32](wint32x4_t a, wint582x4_t am ypecia[_u32](wint32x4_t a, wint582x4_t am ypecia[_u42](wint54x2_t a, wint582x4_t am ypecia[_u64](wint64x2_t a, was 2 on					
min32s4_t_L_arm_lyopselq_Lu32[quin32s4_t_a, a > 0n		`		Qd -> result	MVE
min3244_1 amm_\prosel_[_u32](uim3234_1_a, a > 0_n VMSR_POR_p Day	uint16x8_t b, mve_pred16_t p)	-	VPSEL Qd,Qn,Qm		
wint642_t Lamm_lypesclq_tof (wint64s2_t a, wint64s2_t lamm_lypesclq_tof)(wint64s2_t a, was percent) b > 0m	nint32x4 t [arm]vpsela[n32](nint32x4 t a		VMSR P0 Rn	Od -> result	MVE
p > Rp				Qu' > Tesan	111.12
b > 0m					
P - Rp				Qd -> result	MVE
	uint64x2_t b, mve_pred16_t p)		VPSEL Qd,Qn,Qm		
	float16x8 t [arm]vpselo[f16](float16x8 t a		VMSR P0 Rn	Od -> result	MVF
D > Rp				Qu -> result	MVL
Dotal fo 82, 1 arm vornq[_f16](float16x8_t a, a > Qn		a -> Qn		Qd -> result	MVE
	float32x4_t b, mve_pred16_t p)		VPSEL Qd,Qn,Qm		
	float16v9 t f ama lyama f f161/float16v9 to		VORN Od On Om	Od > moult	MVE
		`	VOKN Qu,Qii,Qiii	Qu -> resuit	MIVE
Int8x16_t [_arm_ vomq[_s8](int8x16_t a, int8x16_t b) a -> Qn b -> Qm VORN Qd,Qn,Qm Qd -> result MVE/NEON b -> Qm VORN Qd,Qn,Qm P -> Rp VORN Qd,Qn,Qm VORN Qd,Qn,Qm VORN Qd,Qn,Qm P -> Rp VORN Qd,Qn,Qm VORN Qd,Q			VORN Qd,Qn,Qm	Qd -> result	MVE
b > Qm		b -> Qm		ì	
Int16x8_t [_arm_ vornq[_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 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 Po > Rp VORN Qd,Qn,Qm VORN Qd,Qn,Qm VORN Qd,Qn,Qm VORN Qd,Qn,Qm VORN Qd,Qn,Qm V	int8x16_t [arm_]vornq[_s8](int8x16_t a, int8x16_t b)		VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
b > Qm	::416-0 4 []	1	VODN 04 0 - 0	0.1 >14	MVEATEON
int32x4_t[_arm_ vornq[_s32 (int32x4_t a, int32x4_t b) a -> On b -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON b -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON b -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON D -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON D -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON D -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON D -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON D -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON D -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON D -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON D -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON D -> Om VORN Qd,Qn,Qm Qd -> result MVE/NEON D -> Om VPST VORNT Qd,Qn,Qm Qd -> result MVE/NEON D -> Om VPST VORNT Qd,Qn,Qm P -> Rp VPST VORNT Qd,Qn,Qm VPST VORNT Q	int16x8_t [arm_jvornq[_s16](int16x8_t a, int16x8_t b)		VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
b > Qm	int32x4 t[arm]vorng[s32](int32x4 t a, int32x4 t b)	1	VORN Od.On.Om	Od -> result	MVE/NEON
b > Qm	12 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	`			
uint16x8_t [_arm_ vornq[u16](uint16x8_t a, uint16x8_t b)	uint8x16_t [arm_]vornq[_u8](uint8x16_t a, uint8x16_t	a -> Qn	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
b > Qm			WORN OLO	0.1	MENTON
uint32x4_t [_arm_ vornq[u32](uint32x4_t a, uint32x4_t b)		-	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
b D D D D D D D D D		_	VORN Od On Om	Od -> result	MVF/NFON
float16x8_t a, float16x8_t b, mve_pred16_t p)			Voter Qu,Qn,Qm	Qu > resuit	W VE/IVEOIV
b -> Qm		inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
Float32x4_t t	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)			VORNT Qd,Qn,Qm		
float32x4_t a, float32x4_t b, mve_pred16_t p)	float32x4 t [arm]vorng m[f32](float32x4 t inactive	 	VMSR P0 Rn	Od -> result	MVE
b -> Qm				Qu > resun	
int8x16_t [_arm_]vornq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)		b -> Qm			
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	1940-16 4 F 1 - 1 - 01/2 +0 - 15 - 1 - 2		VMCD DO D	01.	Myr
b -> Qm				Qd -> result	MVE
p -> Rp	miox10_t a, miox10_t b, mve_pred10_t p)				
int16x8_t [_arm_]vornq_m[_s16](int16x8_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 VPST V		`	- (-/ (, (
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$		inactive -> Qd		Qd -> result	MVE
p -> Rp	int16x8_t a, int16x8_t b, mve_pred16_t p)				
int32x4_t [_arm_]vornq_m[_s32](int32x4_t inactive, inactive -> Qd		-	VOKNI Qa,Qn,Qm		
$ \begin{array}{c} 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\ [_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\ p) \\ \\ uint16x8_t\ a, uint16x8_t\ b, mve_pred16_t\ p) \\ \\ uint16x8_t\ b, mve_pred16_t\ b, mve_pred16_t\ p) \\ \\ uint16x8_t\ b, mve_pred16_t\ b, mve_pred16_t\ b, mve_pred16_t\ b, mve_pred16_t\ b, mve_pr$	int32x4 t[arm]vorng m[s32](int32x4 t inactive.		VMSR P0,Rp	Od -> result	MVE
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$					
$ \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 \\ p -> Rp \end{array} \qquad \begin{array}{c} VMSR \ P0,Rp \\ VPST \\ VORNT \ Qd,Qn,Qm \\ p -> Rp \end{array} \qquad \begin{array}{c} Qd -> result \\ MVE \end{array}$			VORNT Qd,Qn,Qm		
$\begin{array}{c} uint8x16_t\ a,\ uint8x16_t\ b,\ mve_pred16_t\ p) \\ & a -> Qn \\ & b -> Qm \\ & VPST \\ VORNT\ Qd,Qn,Qm \\ & p -> Rp \\ \\ uint16x8_t\ [_arm_]vormq_m[_u16](uint16x8_t\ inactive, \\ & uint16x8_t\ a,\ uint16x8_t\ b,\ mve_pred16_t\ p) \\ & a -> Qn \\ & uint16x8_t\ b,\ mve_pred16_t\ p) \\ & a -> Qn \\ & b -> Qm \\ & VPST \\ & b -> Qm \\ & VORNT\ Qd,Qn,Qm \\ \end{array} \begin{array}{c} Qd -> result \\ MVE \\ \\ MVE \\ \\ A -> Qn \\ \\ B -> Qn \\ \\ CRNT\ Qd,Qn,Qm \\ \\ C$		 	VMCD DO D	01.	Myr
b -> Qm				Qa -> result	MVE
p -> Rp	annostro_t a, annostro_t o, mvc_predio_t p)				
uint16x8_t a, uint16x8_t b, mve_pred16_t p) a -> Qn VPST b -> Qm VORNT Qd,Qn,Qm			Ç-, Ç-, Ç		
b -> Qm VORNT Qd,Qn,Qm				Qd -> result	MVE
	uint16x8_t a, uint16x8_t b, mve_pred16_t p)	-			
		b -> Qm p -> Rp	VOKN1 Qa,Qn,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vornq_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 VORNT Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vornq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORNT Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vornq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORNT Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vornq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORNT Qd,Qn,Qm	Qd -> result	MVE
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 p -> Rp	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)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORNT Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vornq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORNT Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vorrq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> 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)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [_arm_]vorrq_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 VORRT Qd,Qn,Qm	Qd -> result	MVE
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 b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vorrq_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 VORRT Qd,Qn,Qm	Qd -> result	MVE
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	VMSR P0,Rp VPST	Qd -> result	MVE
float32x4_t [_arm_]vorrq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VORRT Qd,Qn,Qm VMSR P0,Rp VPST VORRT Qd, Qn, Qm	Qd -> result	MVE
int8x16_t [arm_]vorrq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm p -> Rp	VORRT Qd,Qn,Qm 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	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 -> Qd	VQMOVNB.U16 Qd,Qm VOMOVNB.U32 Qd,Qm	Qd -> result Od -> result	MVE MVE
uint3x8-t [arm_]vqmovnbq_m[_s16](int8x16_t a,	b -> Qm a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VQMOVNBT.S16 Qd,Qm		. =
int16x8_t [arm_]vqmovnbq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	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, uint32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int8x16_t [_arm_]vqmovntq[_s16](int8x16_t a, int16x8_t	p -> Rp a -> Qd	VQMOVNBT.U32 Qd,Qm VQMOVNT.S16 Qd,Qm	Qd -> result	MVE
b) int16x8_t [_arm_]vqmovntq[_s32](int16x8_t a, int32x4_t b)	b -> Qm a -> Qd b -> Qm	VQMOVNT.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vqmovntq[_u16](uint8x16_t a, uint16x8_t b)	a -> Qd b -> Qm	VQMOVNT.U16 Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vqmovntq[_u32](uint16x8_t a, uint32x4_t b)	a -> Qd b -> Qm	VQMOVNT.U32 Qd,Qm	Qd -> result	MVE
int8x16_t [_arm_]vqmovntq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t [arm_]vqmovntq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p)	p -> Rp a -> Qd b -> Qm	VQMOVNTT.S16 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
uint8x16_t [_arm_]vqmovntq_m[_u16](uint8x16_t a, uint16x8_t b, mve_pred16_t p)	p -> Rp a -> Qd b -> Qm	VQMOVNTT.S32 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
uint16x8_t [_arm_]vqmovntq_m[_u32](uint16x8_t a, uint32x4_t b, mve_pred16_t p)	p -> Rp a -> Qd b -> Qm	VQMOVNTT.U16 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
uint8x16_t [_arm_]vqmovunbq[_s16](uint8x16_t a,	p -> Rp a -> Qd	VQMOVNTT.U32 Qd,Qm VQMOVUNB.S16 Qd,Qm	Qd -> result	MVE
int16x8_t b) uint16x8_t [arm_]vqmovunbq[_s32](uint16x8_t a, int32x4_t b)	b -> Qm a -> Qd b -> Qm	VQMOVUNB.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vqmovunbq_m[_s16](uint8x16_t a, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVUNBT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vqmovunbq_m[_s32](uint16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVUNBT.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vqmovuntq[_s16](uint8x16_t a, int16x8_t b)	a -> Qd b -> Qm	VQMOVUNT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vqmovuntq[_s32](uint16x8_t a, int32x4_t b)	a -> Qd b -> Qm	VQMOVUNT.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vqmovuntq_m[_s16](uint8x16_t a, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVUNTT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vqmovuntq_m[_s32](uint16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVUNTT.S32 Qd,Qm	Qd -> result	MVE
int8x16_t [_arm_]vqrshlq[_n_s8](int8x16_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [_arm_]vqrshlq[_n_s16](int16x8_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [_arm_]vqrshlq[_n_s32](int32x4_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [arm_]vqrshlq[_n_u8](uint8x16_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [_arm_]vqrshlq[_n_u16](uint16x8_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vqrshlq[_n_u32](uint32x4_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [_arm_]vqrshlq_m_n[_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [_arm_]vqrshlq_m_n[_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vqrshlq_m_n[_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vqrshlq_m_n[_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
uint16x8_t [arm_]vqrshlq_m_n[_u16](uint16x8_t a, int32_t b, mve_pred16_t p)	p -> Rp a -> Qda b -> Rm p -> Rp	VQRSHLT.U8 Qda,Rm VMSR P0,Rp VPST VQRSHLT.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vqrshlq_m_n[_u32](uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [arm_]vqrshlq[_s8](int8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VQRSHL.S8 Qd,Qm,Qn	Qd -> result	MVE/NEON

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 -> Qn 0 <= imm <= 7	VQSHL.S8 Qd,Qn,#imm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vqshlq_n[_s16](int16x8_t a, const int imm)	a -> Qn 0 <= imm <= 15	VQSHL.S16 Qd,Qn,#imm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vqshlq_n[_s32](int32x4_t a, const int imm)	a -> Qn 0 <= imm <= 31	VQSHL.S32 Qd,Qn,#imm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vqshlq_n[_u8](uint8x16_t a, const int imm)	a -> Qn 0 <= imm <= 7	VQSHL.U8 Qd,Qn,#imm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vqshlq_n[_u16](uint16x8_t a, const int imm)	a -> Qn 0 <= imm <= 15	VQSHL.U16 Qd,Qn,#imm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vqshlq_n[_u32](uint32x4_t a, const int imm)	a -> Qn 0 <= imm <= 31	VQSHL.U32 Qd,Qn,#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 -> Qn 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VQSHLT.S8 Qd,Qn,#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 -> Qn 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VQSHLT.S16 Qd,Qn,#imm	Qd -> result	MVE
int32x4_t [_arm_]vqshlq_m_n[_s32](int32x4_t inactive, int32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qn 0 <= imm <= 31	VMSR P0,Rp VPST VQSHLT.S32 Qd,Qn,#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 -> Qn 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VQSHLT.U8 Qd,Qn,#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 -> Qn 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VQSHLT.U16 Qd,Qn,#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 -> Qn 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VQSHLT.U32 Qd,Qn,#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	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VQSHLT.S16 Qda,Rm		
int32x4 t [arm vqshlq m r[s32](int32x4 t a, int32 t	a -> Oda	VMSR P0,Rp	Oda -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST	Q	1
	p -> Rp	VQSHLT.S32 Qda,Rm		
uint8x16_t [arm_]vqshlq_m_r[_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
int32_t b, nive_pied10_t p)	p -> Rn	VQSHLT.U8 Qda,Rm		
uint16x8_t [arm_]vqshlq_m_r[_u16](uint16x8_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VQSHLT.U16 Qda,Rm	0.1	NOTE
uint32x4_t [arm_]vqshlq_m_r[_u32](uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
mio2_t o, mvc_picaro_t p)	p -> Rn	VQSHLT.U32 Qda,Rm		
uint8x16_t [arm_]vqshluq[_n_s8](int8x16_t a, const int	a -> Qn	VQSHLU.S8 Qd,Qn,#imm	Qd -> result	MVE
imm)	0 <= imm <= 7	YYORW YY GLEO LO W) (T) (T)
uint16x8_t [arm_]vqshluq[_n_s16](int16x8_t a, const int imm)	a -> Qn 0 <= imm <=	VQSHLU.S16 Qd,Qn,#imm	Qd -> result	MVE
iiit iiiiiii)	15			
uint32x4_t [arm_]vqshluq[_n_s32](int32x4_t a, const	a -> Qn	VQSHLU.S32 Qd,Qn,#imm	Qd -> result	MVE
int imm)	0 <= imm <=			
20140-16 4 f - 2014 location and a 2014-in-0-16 4	31	VMCD DO D.	0.1	MVE
uint8x16_t [arm_]vqshluq_m[_n_s8](uint8x16_t inactive, int8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, medicate at const me mini, mvc_predio_t p)	$0 \le imm \le 7$	VQSHLUT.S8 Qd,Qn,#imm		
	p -> Rp	7 . 7 . 7		
uint16x8_t [arm_]vqshluq_m[_n_s16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int16x8_t a, const int imm, mve_pred16_t p)	a -> Qn 0 <= imm <=	VPST VQSHLUT.S16 Qd,Qn,#imm		
	15	VQSIIEC1.S10 Qu,Qii,#iiliili		
	p -> Rp			
uint32x4_t [arm_]vqshluq_m[_n_s32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int32x4_t a, const int imm, mve_pred16_t p)	a -> Qn 0 <= imm <=	VPST VQSHLUT.S32 Qd,Qn,#imm		
	31	VQSHLU1.332 Qd,Qii,#iiliili		
	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	W V E
	1 <= imm <=			
	16	VOCUDNID LITE OF ON #:	Od -> result	MVE
uint8x16_t [arm_]vqshrnbq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm	VQSHRNB.U16 Qd,Qm,#imm	Qu -> result	MVE
difference of court in mining	1 <= imm <= 8			
uint16x8_t [arm_]vqshrnbq[_n_u32](uint16x8_t a,	a -> Qd	VQSHRNB.U32 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t b, const int imm)	b -> Qm			
	1 <= imm <= 16			
int8x16_t [arm_]vqshrnbq_m[_n_s16](int8x16_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, const int imm, mve_pred16_t p)	b -> Qm	VPST		
	1 <= imm <= 8	VQSHRNBT.S16 Qd,Qm,#imm		
int16x8_t [arm_]vqshrnbq_m[_n_s32](int16x8_t a,	p -> Rp a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, const int imm, mve_pred16_t p)	b -> Qu b -> Qm	VPST VPST	Qu -> resuit	WIVE
	1 <= imm <=	VQSHRNBT.S32 Qd,Qm,#imm		
	16			
pint0v16 tf arm brochmba and a v161/vint0v16 t	p -> Rp	VMSP DO Po	Od > =====14	MVE
uint8x16_t [arm_]vqshrnbq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	1 <= imm <= 8	VQSHRNBT.U16 Qd,Qm,#imm		1
	p -> Rp			1
uint16x8_t [_arm_]vqshrnbq_m[_n_u32](uint16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, const int imm, mve_pred16_t p)	b -> Qm 1 <= imm <=	VPST VQSHRNBT.U32 Qd,Qm,#imm		
	16	. Qorna 15 1.032 Qu,Qm,#illilli		1
	p -> Rp			
int8x16_t [arm_]vqshrntq[_n_s16](int8x16_t a,	a -> Qd	VQSHRNT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t b, const int imm)	b -> Qm			
int16x8_t [arm_]vqshrntq[_n_s32](int16x8_t a,	1 <= imm <= 8 a -> Od	VQSHRNT.S32 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t b, const int imm)	b -> Qm	. goria (1.552 Qu,Qiii,#iiiiii	Qu > resuit	
	1 <= imm <=			
	16			

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		
14604504504504	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_predro_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		
uint16x8_t [arm_]vrev32q_m[_u16](uint16x8_t	p -> Rp inactive -> Od	VREV32T.8 Qd,Qm VMSR P0,Rp	0.1	MVE
inactive, uint16x8_t a, mve_pred16_t p)	a -> Qm	VMSR PU,RP VPST	Qd -> result	MVE
mactive, unitroxo_t a, mve_pred1o_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		
1 0 16 (F) 20 F 01/1 (0 16)	p -> Rp	VREV32T.16 Qd,Qm	0.1	MATE
int8x16_t [arm_]vrev32q_x[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
nive_picuro_t p)	р -> Кр	VREV32T.8 Qd,Qm		
int16x8_t [arm_]vrev32q_x[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> 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 VPST	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	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		
		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 VREV32T.16 Qd,Qm		
int8x16_t [arm_]vrev64q[_s8](int8x16_t a)	a -> Qm	VREV64.8 Od,Om	Od -> result	MVE/NEON
int16x8_t [arm_]vrev64q[_s16](int16x8_t a)	a -> Qm	VREV64.16 Qd,Qm	Qd -> result	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) float32x4_t [arm_]vrev64q[_f32](float32x4_t a)	a -> Qm a -> Qm	VREV64.16 Qd,Qm VREV64.32 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
int8x16_t [arm_]vrev64q_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE/NEON
int8x16_t a, mve_pred16_t p)	a -> Qm	VPST	Qu'y resun	1,1,2
	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		
int32x4_t [arm_]vrev64q_m[_s32](int32x4_t inactive,	p -> Rp inactive -> Qd	VREV64T.16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
int32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Qu -> resuit	MVL
	p -> Rp	VREV64T.32 Qd,Qm		
uint8x16_t [arm_]vrev64q_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
uint16x8_t [arm_]vrev64q_m[_u16](uint16x8_t	p -> Rp inactive -> Qd	VREV64T.8 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
inactive, uint16x8_t a, mve_pred16_t p)	a -> Qm	VPST	Qu -> resuit	WIVE
· · ·	p -> Rp	VREV64T.16 Qd,Qm		
uint32x4_t [arm_]vrev64q_m[_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
float16x8_t [arm_]vrev64q_m[_f16](float16x8_t	p -> Rp inactive -> Qd	VREV64T.32 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Om	VMSK FO,KP VPST	Qu -> resuit	MIVE
	p -> Rp	VREV64T.16 Qd,Qm		
float32x4_t [arm_]vrev64q_m[_f32](float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
int8x16_t [arm_]vrev64q_x[_s8](int8x16_t a,	p -> Rp a -> Qm	VREV64T.32 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	a -> Qm p -> Rp	VMSR PU,RP VPST	Qu -> resuit	IVI V E
- <u>r</u> - r - r - r - r - r - r - r - r - r -	1	VREV64T.8 Qd,Qm		
int16x8_t [arm_]vrev64q_x[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
: 1200-4 4 F	0	VREV64T.16 Qd,Qm	0.1 : 1:	MVE
int32x4_t [arm_]vrev64q_x[_s32](int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
mro_prouto_t p)	h -> Kh	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		
		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		
uint32x4_t [arm_]vrev64q_x[_u32](uint32x4_t a,	a -> Qm	VREV64T.16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	$p \rightarrow Rp$	VPST	Qu'y Tesun	11172
		VREV64T.32 Qd,Qm	0.1	MALE
float16x8_t [arm_]vrev64q_x[_f16](float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
	PP	VREV64T.16 Qd,Qm		
float32x4_t [_arm_]vrev64q_x[_f32](float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VREV64T.32 Qd,Qm		
int8x16_t [arm_]vrshlq[_n_s8](int8x16_t a, int32_t b)	a -> Qda	VRSHL.S8 Qda,Rm	Qda -> result	MVE
116.0 15 1 115 1676 16.0 1 120 11	b -> Rm	ADOM GICOL D	0.1	MATE
int16x8_t [arm_]vrshlq[_n_s16](int16x8_t a, int32_t b)	a -> Qda b -> Rm	VRSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vrshlq[_n_s32](int32x4_t a, int32_t b)	a -> Qda	VRSHL.S32 Qda,Rm	Qda -> result	MVE
10.16.16	b -> Rm	AND OWN AND OLD TO	0.1) am
uint8x16_t [arm_]vrshlq[_n_u8](uint8x16_t a, int32_t b)	a -> Qda b -> Rm	VRSHL.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [arm_]vrshlq[_n_u16](uint16x8_t a, int32_t	a -> Qda	VRSHL.U16 Qda,Rm	Qda -> result	MVE
b)	b -> Rm	AMBORN TAGS OF 2		NOW
uint32x4_t [arm_]vrshlq[_n_u32](uint32x4_t a, int32_t b)	a -> Qda b -> Rm	VRSHL.U32 Qda,Rm	Qda -> result	MVE
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	VPST		
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	VPST	Qua -> result	MVE
	p -> Rp	VRSHLT.S16 Qda,Rm		
int32x4_t [arm_]vrshlq_m_n[_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
	p -> Rn	VRSHLT.S32 Qda,Rm		
uint8x16_t [arm_]vrshlq_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 VRSHLT.U8 Qda,Rm		
uint16x8_t [arm_]vrshlq_m_n[_u16](uint16x8_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
uint32x4_t [arm_]vrshlq_m_n[_u32](uint32x4_t a,	p -> Rp a -> Qda	VRSHLT.U16 Qda,Rm VMSR P0,Rp	Oda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST	Qua -> result	MVE
	p -> Rp	VRSHLT.U32 Qda,Rm		
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	VRSHL.S16 Qd,Qm,Qn	Qd -> result	MVE/NEON
	b -> Qn			
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	VRSHL.U8 Qd,Qm,Qn	Qd -> result	MVE/NEON
-	b -> Qn			
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	a -> Qm	VRSHL.U32 Qd,Qm,Qn	Qd -> result	MVE/NEON
b)	b -> Qn	7 . 7 . 7		
int8x16_t [arm_]vrshlq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mioxio_t u, mioxio_t o, mive_predio_t p)	b -> Qn	VRSHLT.S8 Qd,Qm,Qn		
1.150.15	p -> Rp	VINCOR DO D	0.1) am
int16x8_t [arm_]vrshlq_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
	b -> Qn	VRSHLT.S16 Qd,Qm,Qn		
in(22=4.4 f]	p -> Rp	VMCD DO D.	01 > 1:	MVE
int32x4_t [arm_]vrshlq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qn	VRSHLT.S32 Qd,Qm,Qn		
nint0-16 of any levels of 02/2 of 16 of	p -> Rp	VMCD DO D.	01 - 1	MVE
uint8x16_t [_arm_]vrshlq_m[_u8](uint8x16_t inactive, uint8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qn	VRSHLT.U8 Qd,Qm,Qn		
pintfey0 tf ann lymbla mf -167/civife-0 til	p -> Rp	VMCD DO Do	04 >14	MVE
uint16x8_t [_arm_]vrshlq_m[_u16](uint16x8_t inactive, uint16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qn	VRSHLT.U16 Qd,Qm,Qn		
	p -> Rp			

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 p -> Rp	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)	a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.S8 Qd,Qm,Qn	Qd -> result	MVE
int16x8_t [arm_]vrshlq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qm b -> Qn p -> Rp	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 <= 16	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)	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 p -> Rp	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)	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 int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHLLTT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vshlltq_x[_n_u16](uint16x8_t a, const int imm, mve_pred16_t p)	a -> Qm 1 <= imm <= 16	VMSR P0,Rp VPST VSHLLTT.U16 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vshlq[_s8](int8x16_t a, int8x16_t b)	p -> Rp a -> Qm b -> Qn	VSHL.S8 Qd,Qm,Qn	Qd -> result	MVE/NEON
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 b -> On	VSHL.S32 Qd,Qm,Qn	Qd -> result	MVE/NEON
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 b)	a -> Qm b -> Qn	VSHL.U16 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint32x4_t [arm_]vshlq[_u32](uint32x4_t a, int32x4_t b)	a -> Qm b -> Qn	VSHL.U32 Qd,Qm,Qn	Qd -> result	MVE/NEON
int8x16_t [arm_]vshlq_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 VSHLT.S8 Qd,Qm,Qn	Qd -> result	MVE
int16x8_t [arm_]vshlq_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 VSHLT.S16 Qd,Qm,Qn	Qd -> result	MVE
int32x4_t [arm_]vshlq_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 VSHLT.S32 Qd,Qm,Qn	Qd -> result	MVE
uint8x16_t [_arm_]vshlq_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 VSHLT.U8 Qd,Qm,Qn	Qd -> result	MVE
uint16x8_t [_arm_]vshlq_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 VSHLT.U16 Qd,Qm,Qn	Qd -> result	MVE
uint32x4_t [_arm_]vshlq_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 VSHLT.U32 Qd,Qm,Qn	Qd -> result	MVE
int8x16_t [arm_]vshlq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.S8 Qd,Qm,Qn	Qd -> result	MVE
int16x8_t [arm_]vshlq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.S16 Qd,Qm,Qn	Qd -> result	MVE
int32x4_t [arm_]vshlq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.S32 Qd,Qm,Qn	Qd -> result	MVE
uint8x16_t [_arm_]vshlq_x[_u8](uint8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.U8 Qd,Qm,Qn	Qd -> result	MVE
uint16x8_t [arm_]vshlq_x[_u16](uint16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.U16 Od,Om,On	Qd -> result	MVE
uint32x4_t [_arm_]vshlq_x[_u32](uint32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.U32 Qd,Qm,Qn	Qd -> result	MVE
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 imm)	a -> Qm 0 <= imm <= 15	VSHL.S16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshlq_n[_s32](int32x4_t a, const int imm)	a -> Qm 0 <= imm <= 31	VSHL.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vshlq_n[_u8](uint8x16_t a, const int imm)	a -> Qm 0 <= imm <= 7	VSHL.U8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshlq_n[_u16](uint16x8_t a, const int imm)	a -> Qm 0 <= imm <= 15	VSHL.U16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [arm_]vshlq_n[_u32](uint32x4_t a, const int imm)	a -> Qm 0 <= imm <= 31	VSHL.U32 Qd,Qm,#imm	Qd -> result	MVE

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.II6 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.132 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	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)	p -> Rp 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.I32 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.I32 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.I32 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_]vshmbq[_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.I16 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	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)	p -> Rp 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)	$a \rightarrow Qn$ $0 <= idx <= 3$	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)	$a \rightarrow Qn$ $0 <= idx <= 7$	VMOV.S16 Rt,Qn[idx]	Rt -> result	MVE/NEON
int32_t [_arm_]vgetq_lane[_s32](int32x4_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 3$	VMOV.32 Rt,Qn[idx]	Rt -> result	MVE/NEON
int64_t [_arm_]vgetq_lane[_s64](int64x2_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 1$	VMOV Rt1,Rt2,D(2*n+idx)	[Rt1,Rt2] -> result	MVE/NEON
uint8_t [_arm_]vgetq_lane[_u8](uint8x16_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 15$	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_u6(void)	<u> </u>		Od -> result	MVE
uint32x4_t [arm_]vuninitializedq_u32(void)		<u>† </u>	Od -> result	MVE
uint64x2_t [arm_]vuninitializedq_u64(void)			Od -> result	MVE
float16x8_t [arm_]vuninitializedq_f16(void)			Qd -> result	MVE
float32x4_t [arm_]vuninitializedq_f32(void)			Qd -> result	MVE
int8x16_t [_arm_]vuninitializedq(int8x16_t t)	t -> Do Not Evaluate		Qd -> result	MVE
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	Od -> 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 -> Qd a -> Qd	NOP	Qd -> result	MVE/NEON
a) uint64x2_t [_arm_]vreinterpretq_u64[_u16](uint16x8_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)	01	NOD	01 : 1:	MUEATEON
int64x2_t [_arm_]vreinterpretq_s64[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_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)				
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 -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [arm_]vreinterpretq_s16[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [arm_]vreinterpretq_s32[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_u64](uint64x2_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)				
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	Qd -> result	MVE/NEON
a)				
uint32x4_t [arm_]vreinterpretq_u32[_u64](uint64x2_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)		1100	1	
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)			1	L
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)				
uint8x16_t [arm_]vreinterpretq_u8[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	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	Qd -> result	MVE/NEON
a)				
uint64x2_t [arm_]vreinterpretq_u64[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [arm_]vreinterpretq_s64[_f16](float16x8_t a)	a -> Qd	NOP	Od -> result	MVE/NEON
uint64_t [arm_]\telf\telf\telf\telf\telf\telf\telf\telf	value ->	LSLL RdaLo,RdaHi,Rm	[RdaHi,RdaLo]	MVE/NEON MVE
umto+_t [arm_jisn(umto+_t value, filt32_t siliit)	[RdaHi,RdaLo]	LOLL NUALU, NUAFII, KIII	-> result	1V1 V 12
	shift -> Rm		-> icouit	
int64_t [arm_]asrl(int64_t value, int32_t shift)	value ->	ASRL RdaLo,RdaHi,Rm	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]	. LONE ROLLO, ROLLII, RIII	-> result	111 7 12
	shift -> Rm		, result	
uint64_t [arm_]uqrshll(uint64_t value, int32_t shift)	value ->	UORSHLL	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]	RdaLo,RdaHi,#64,Rm	-> result	
	shift -> Rm	, , , , , , , , , , , , , , , , , , , ,		
uint64_t [arm_]ugrshll_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		<u> </u>	<u></u>
int64_t [arm_]sqrshrl_sat48(int64_t value, int32_t shift)	value ->	SQRSHRL	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]	RdaLo,RdaHi,#48,Rm	-> result	
	shift -> Rm			
uint64_t [arm_]uqshll(uint64_t value, const int shift)	value ->	UQSHLL RdaLo,RdaHi,#shift	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]		-> result	
	1 <= shift <= 32			<u></u>
uint64_t [arm_]urshrl(uint64_t value, const int shift)	value ->	URSHRL RdaLo,RdaHi,#shift	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]		-> result	
		1	i i	
<u></u>	1 <= shift <= 32			
int64_t [arm_]srshrl(int64_t value, const int shift)	value ->	SRSHRL RdaLo,RdaHi,#shift	[RdaHi,RdaLo]	MVE
int64_t [_arm_]srshrl(int64_t value, const int shift)		SRSHRL RdaLo,RdaHi,#shift	[RdaHi,RdaLo] -> result	MVE

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