

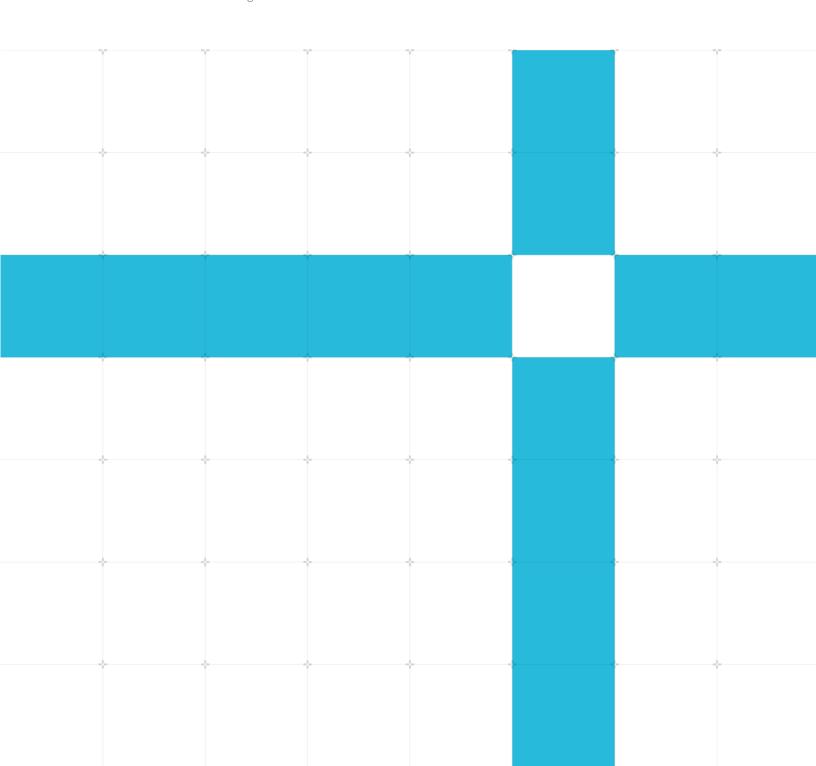
Arm MVE Intrinsics Reference for ACLE Q2 2020

Non-Confidential

Copyright © 2019-2020 Arm Limited (or its affiliates). All rights reserved.

Issue Q220-00

101809



Arm MVE Intrinsics

Reference

Copyright © 2019-2020 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
Q219- 00	30 June 2019	Non-Confidential	Version ACLE Q2 2019.
Q319- 00	30 September 2019	Non-Confidential	Version ACLE Q3 2019
Q419- 00	31 December 2019	Non-Confidential	Version ACLE Q4 2019
Q220- 00	30 May 2020	Non-Confidential	Version ACLE Q2 2020

Non-Confidential Proprietary Notice

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

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

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

This document may include technical inaccuracies or typographical errors.

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

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation

of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

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

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

Copyright © 2019-2020 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LFS-PRF-20349

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

http://www.arm.com

About this document

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

List of Intrinsics

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

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	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)	imm in [1,2,4,8] *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
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	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
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	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
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	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]	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)	p -> Rp *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]	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)	p -> 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]	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 -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32_t a, uint32_t b, const int imm,	a -> Rn	VPST		
mve_pred16_t p)	b -> Rm	VIWDUPT.U16 Qd,Rn,Rm,imm		
	imm in [1,2,4,8] p -> Rp			
uint32x4_t [arm_]viwdupq_m[_n_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32_t a, uint32_t b, const int imm,	a -> Rn	VPST	Qu > resurt	141 4 12
mve_pred16_t p)	b -> Rm	VIWDUPT.U32 Qd,Rn,Rm,imm		
-1 - 1/	imm in [1,2,4,8]			
	p -> Rp			
uint8x16_t [arm_]viwdupq_m[_wb_u8](uint8x16_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
nactive, uint32_t * a, uint32_t b, const int imm,	*a -> Rn	VPST	Rn -> *a	
mve_pred16_t p)	b -> Rm	VIWDUPT.U8 Qd,Rn,Rm,imm		
	imm in [1,2,4,8] p -> Rp			
uint16x8_t [arm_]viwdupq_m[_wb_u16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, uint32_t * a, uint32_t b, const int imm,	*a -> Rn	VPST	Rn -> *a	MAL
nve_pred16_t p)	b -> Rm	VIWDUPT.U16 Qd,Rn,Rm,imm	14.1 / 11	
	imm in [1,2,4,8]			
	p -> Rp			
uint32x4_t [arm_]viwdupq_m[_wb_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32_t * a, uint32_t b, const int imm,	*a -> Rn	VPST	Rn -> *a	
nve_pred16_t p)	b -> Rm	VIWDUPT.U32 Qd,Rn,Rm,imm		
	imm in [1,2,4,8]			
-i-49-16 4 []-i	p -> Rp	VMCD DO D.:	0.1 >16	MAZE
uint8x16_t [_arm_]viwdupq_x[_n]_u8(uint32_t a,	a -> Rn b -> Rm	VMSR P0,Rp VPST	Qd -> result	MVE
uint32_t b, const int imm, mve_pred16_t p)	imm in [1,2,4,8]	VIWDUPT.U8 Qd,Rn,Rm,imm		
	p -> Rp	VIWDOI 1.00 Qu,Kii,Kiii,iiiiii		
uint16x8_t [arm_]viwdupq_x[_n]_u16(uint32_t a,	a -> Rn	VMSR P0,Rp	Od -> result	MVE
uint32_t b, const int imm, mve_pred16_t p)	b -> Rm	VPST	(1
	imm in [1,2,4,8]	VIWDUPT.U16 Qd,Rn,Rm,imm		
	p -> Rp	-		
uint32x4_t [arm_]viwdupq_x[_n]_u32(uint32_t a,	a -> Rn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, const int imm, mve_pred16_t p)	b -> Rm	VPST		
	imm in [1,2,4,8]	VIWDUPT.U32 Qd,Rn,Rm,imm		
1.0.15.1	p -> Rp	VILLED DO D	0.1) a m
uint8x16_t [_arm_]viwdupq_x[_wb]_u8(uint32_t * a,	*a -> Rn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, const int imm, mve_pred16_t p)	b -> Rm	VPST VIWDUPT.U8 Qd,Rn,Rm,imm	Rn -> *a	
	imm in [1,2,4,8] p -> Rp	VIWDOF 1.08 Qu,Kii,Kiii,iiiiii		
uint16x8_t [arm_]viwdupq_x[_wb]_u16(uint32_t * a,	*a -> Rn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, const int imm, mve_pred16_t p)	b -> Rm	VPST	Rn -> *a	141 4 12
amio2_t o, const int immi, invo_preuro_t p)	imm in [1,2,4,8]	VIWDUPT.U16 Qd,Rn,Rm,imm	14.1	
	p -> Rp			
	. D	VMSR P0,Rp	Od -> result	
uint32x4_t [arm_]viwdupq_x[_wb]_u32(uint32_t * a,	*a -> Rn	v Misik i O,kp	Qu > resure	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	VPST	Rn -> *a	MVE
	b -> Rm imm in [1,2,4,8]			MVE
uint32_t b, const int imm, mve_pred16_t p)	b -> Rm imm in [1,2,4,8] p -> Rp	VPST VIWDUPT.U32 Qd,Rn,Rm,imm	Rn -> *a	
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [arm_]vdupq_n_s8(int8_t a)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [arm_]vdupq_n_s8(int8_t a) int16x8_t [arm_]vdupq_n_s16(int16_t a)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt	Rn -> *a Qd -> result Qd -> result	MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s32(int32_t a)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt	Rn -> *a Qd -> result Qd -> result Qd -> result Qd -> result	MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s32(int32_t a) uint8x16_t [_arm_]vdupq_n_u8(uint8_t a)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt a -> Rt a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.8 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s32(int32_t a) uint8x16_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt a -> Rt a -> Rt a -> Rt a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.8 Qd,Rt VDUP.16 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s32(int32_t a) uint8x16_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a) uint32x4_t [_arm_]vdupq_n_u32(uint32_t a)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt a -> Rt a -> Rt a -> Rt a -> Rt a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [arm_]vdupq_n_s8(int8_t a) int16x8_t [arm_]vdupq_n_s16(int16_t a) int132x4_t [arm_]vdupq_n_s32(int32_t a) uint8x16_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a) uint32x4_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s32(int32_t a) uint8x16_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a) uint32x4_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) float32x4_t [_arm_]vdupq_n_f32(float32_t a)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s32(int32_t a) uint8x16_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a) uint16x8_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) float32x4_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[n_s8](int8x16_t inactive,	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s32(int32_t a) uint8x16_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a) uint32x4_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) float32x4_t [_arm_]vdupq_n_f32(float32_t a)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.39 Qd,Rt VDUP.39 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a) uint32x4_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) float32x4_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[_n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s32(int32_t a) uint8x16_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a) uint16x8_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) float32x4_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[n_s8](int8x16_t inactive,	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.39 Qd,Rt VDUP.39 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_u8(uint3_t a) uint8x16_t [_arm_]vdupq_n_u16(uint16_t a) uint16x8_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) float32x4_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_n_f38[(int8x16_t inactive, int8_t a, mve_pred16_t p) int16x8_t [_arm_]vdupq_m[_n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt inactive -> Qd a -> Rt p -> Rp inactive -> Qd a -> Rt p -> Rp	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a) uint32x4_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) int8x16_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p) int16x8_t [_arm_]vdupq_m[_n_s16](int16x8_t inactive, int16x8_t [_arm_]vdupq_m[_n_s16](int16x8_t inactive,	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt inactive -> Qd a -> Rp inactive -> Qd a -> Rp	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_u8(uint3_t a) uint8x16_t [_arm_]vdupq_n_u16(uint16_t a) uint16x8_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) float32x4_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_n_f38[(int8x16_t inactive, int8_t a, mve_pred16_t p) int16x8_t [_arm_]vdupq_m[_n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt inactive -> Qd a -> Rt p -> Rp inactive -> Qd a -> Rt p -> Rp inactive -> Qd a -> Rt p -> Rp inactive -> Qd a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST	Rn -> *a Qd -> result	MVE/NEON
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_u8(uint8_t a) uint18x16_t [_arm_]vdupq_n_u8(uint8_t a) uint18x8_t [_arm_]vdupq_n_u16(uint16_t a) uint32x4_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) int8x16_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p) int16x8_t [_arm_]vdupq_m[n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p) int32x4_t [_arm_]vdupq_m[n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt inactive -> Qd a -> Rt p -> Rp inactive -> Qd a -> Rt p -> Rp inactive -> Qd a -> Rt p -> Rp	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE MVE MVE
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s32(int32_t a) uint8x16_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a) uint32x4_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) float32x4_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[_n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p) int16x8_t [_arm_]vdupq_m[_n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p) int32x4_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_u8](uint8x16_t inactive, int32_t a, mve_pred16_t p)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt p -> Rp inactive -> Qd a -> Rt p -> Rp inactive -> Qd	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt VMSR P0,Rp	Rn -> *a Qd -> result	MVE/NEON MVE MVE
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_u8(uint8_t a) uint18x16_t [_arm_]vdupq_n_u8(uint8_t a) uint18x8_t [_arm_]vdupq_n_u16(uint16_t a) uint32x4_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) int8x16_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p) int16x8_t [_arm_]vdupq_m[n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p) int32x4_t [_arm_]vdupq_m[n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt inactive -> Qd a -> Rt p -> Rp inactive -> Qd a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE MVE MVE
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_u8(uint8_t a) uint8x16_t [_arm_]vdupq_n_u16(uint16_t a) uint16x8_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) float32x4_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[_n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p) int16x8_t [_arm_]vdupq_m[_n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p) int32x4_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_u8](uint8x16_t inactive, uint8x16_t [_arm_]vdupq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, mve_pred16_t p)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt inactive -> Qd a -> Rt p -> Rp inactive -> Qd a -> Rt p -> Rp	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE MVE MVE
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s2(int32_t a) uint8x16_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a) uint32x4_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[_n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p) int16x8_t [_arm_]vdupq_m[_n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p) int32x4_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int16_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_s16](uint8x16_t inactive, int32_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_u8](uint8x16_t inactive, uint8_t a, mve_pred16_t p) uint16x8_t [_arm_]vdupq_m[_n_u16](uint16x8_t t) uint16x8_t [_arm_]vdupq_m[_n_u16](uint16x8_t t)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rp inactive -> Qd a -> Rt p -> Rp	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE MVE MVE
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_u8(uint8_t a) uint8x16_t [_arm_]vdupq_n_u16(uint16_t a) uint16x8_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f16(float16_t a) float32x4_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[_n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p) int16x8_t [_arm_]vdupq_m[_n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p) int32x4_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_u8](uint8x16_t inactive, uint8x16_t [_arm_]vdupq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, mve_pred16_t p)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt inactive -> Qd a -> Rt p -> Rp inactive -> Qd a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE MVE MVE
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s22(int32_t a) uint8x16_t [_arm_]vdupq_n_u16(uint16_t a) uint16x8_t [_arm_]vdupq_n_u32(uint32_t a) ffloat16x8_t [_arm_]vdupq_n_f16(float16_t a) int8x16_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[_n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p) int16x8_t [_arm_]vdupq_m[_n_s16](int16x8_t inactive, int32_t a, mve_pred16_t p) uint32x4_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_u8](uint8x16_t inactive, uint8_t a, mve_pred16_t p) uint16x8_t [_arm_]vdupq_m[_n_u8](uint8x16_t inactive, uint8_t a, mve_pred16_t p) uint16x8_t [_arm_]vdupq_m[_n_u16](uint16x8_t inactive, uint16x8_t [_arm_]vdupq_m[_n_u16](uint16x8_t inactive, uint16x8_t [_arm_]vdupq_m[_n_u16](uint16x8_t inactive, uint16_t a, mve_pred16_t p)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt inactive -> Qd a -> Rt p -> Rp inactive -> Qd a -> Rt p -> Rp	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE/NEON MVE MVE MVE MVE MVE
uint32_t b, const int imm, mve_pred16_t p) int8x16_t [_arm_]vdupq_n_s8(int8_t a) int16x8_t [_arm_]vdupq_n_s16(int16_t a) int32x4_t [_arm_]vdupq_n_s2(int32_t a) uint8x16_t [_arm_]vdupq_n_u8(uint8_t a) uint16x8_t [_arm_]vdupq_n_u16(uint16_t a) uint32x4_t [_arm_]vdupq_n_u32(uint32_t a) float16x8_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[_n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p) int16x8_t [_arm_]vdupq_m[_n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p) int32x4_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int16_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_s16](uint8x16_t inactive, int32_t a, mve_pred16_t p) uint8x16_t [_arm_]vdupq_m[_n_u8](uint8x16_t inactive, uint8_t a, mve_pred16_t p) uint16x8_t [_arm_]vdupq_m[_n_u16](uint16x8_t t) uint16x8_t [_arm_]vdupq_m[_n_u16](uint16x8_t t)	b -> Rm imm in [1,2,4,8] p -> Rp a -> Rt a -> Rt inactive -> Qd a -> Rt p -> Rp inactive -> Qd a -> Rt	VPST VIWDUPT.U32 Qd,Rn,Rm,imm VDUP.8 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.16 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VDUP.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt VMSR P0,Rp VPST VDUPT.32 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.8 Qd,Rt	Rn -> *a Qd -> result	MVE/NEON MVE MVE MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [arm_]vdupq_m[_n_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16_t a, mve_pred16_t p)	a -> Rt	VPST		
float32x4_t [_arm_]vdupq_m[_n_f32](float32x4_t	p -> Rp inactive -> Od	VDUPT.16 Qd,Rt VMSR P0,Rp	Od -> result	MVE
inactive, float32 t a, mve_pred16_t p)	a -> Rt	VMSK FO,KP VPST	Qu -> resuit	IVI V E
	p -> Rp	VDUPT.32 Qd,Rt		
int8x16_t [arm_]vdupq_x_n_s8(int8_t a, mve_pred16_t	a -> Rt	VMSR P0,Rp	Qd -> result	MVE
p)	p -> Rp	VPST		
int16x8 t[arm]vdupq x n s16(int16 t a,	a -> Rt	VDUPT.8 Qd,Rt VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu -> result	WYL
	r	VDUPT.16 Qd,Rt		
int32x4_t [arm_]vdupq_x_n_s32(int32_t a,	a -> Rt	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
uint8x16_t [arm_]vdupq_x_n_u8(uint8_t a,	a -> Rt	VDUPT.32 Qd,Rt VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu -> result	WIVE
	r	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
nve pred16 tp)	p -> Rp	VPST	Qu -> result	IVIVE
	r ·r'	VDUPT.32 Qd,Rt		
loat16x8_t [arm_]vdupq_x_n_f16(float16_t a,	a -> Rt	VMSR P0,Rp	Qd -> result	MVE
nve_pred16_t p)	p -> Rp	VPST		
Haat22v4 t [amm lydyna v m f22/flaat22 t a	a -> Rt	VDUPT.16 Qd,Rt VMSR P0,Rp	Od -> result	MVE
float32x4_t [arm_]vdupq_x_n_f32(float32_t a, nve_pred16_t p)	p -> Rp	VMSK PO,KP VPST	Qu -> resuit	MVE
inve_prearo_t p)	P × MP	VDUPT.32 Qd,Rt		
mve_pred16_t [arm_]vcmpeqq[_f16](float16x8_t a,	a -> Qn	VCMP.F16 eq,Qn,Qm	Rd -> result	MVE
float16x8_t b)	b -> Qm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq[_f32](float32x4_t a,	a -> Qn	VCMP.F32 eq,Qn,Qm	Rd -> result	MVE
float32x4_t b) mve_pred16_t [arm_]vcmpeqq[_s8](int8x16_t a,	b -> Qm a -> On	VMRS Rd,P0 VCMP.I8 eq,Qn,Qm	Rd -> result	MVE
int8x16_t b)	b -> Qm	VMRS Rd,P0	Ru -> resuit	WYL
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 -> Qn	VMRS Rd,P0 VCMP.I8 eq,Qn,Qm	Rd -> result	MVE
uint8x16_t b)	b -> Qm	VMRS Rd,P0	Ru -> resuit	WYL
mve_pred16_t [arm_]vcmpeqq[_u16](uint16x8_t a,	a -> Qn	VCMP.I16 eq,Qn,Qm	Rd -> result	MVE
uint16x8_t b)	b -> Qm	VMRS Rd,P0		
nve_pred16_t [arm_]vcmpeqq[_u32](uint32x4_t a,	a -> Qn	VCMP.I32 eq,Qn,Qm	Rd -> result	MVE
nint32x4_t b) nve pred16 t [arm]vcmpeqq[n f16](float16x8 t a,	b -> Qm a -> Qn	VMRS Rd,P0 VCMP.F16 eq,Qn,Rm	Rd -> result	MVE
loat16_t b)	b -> Rm	VMRS Rd,P0	Ku => icsuit	IVIVE
nve_pred16_t [arm_]vcmpeqq[_n_f32](float32x4_t a,	a -> Qn	VCMP.F32 eq,Qn,Rm	Rd -> result	MVE
loat32_t b)	b -> Rm	VMRS Rd,P0		
nve_pred16_t [arm_]vcmpeqq[_n_s8](int8x16_t a,	a -> Qn	VCMP.I8 eq,Qn,Rm	Rd -> result	MVE
nt8_t b) nve_pred16_t [arm_]vcmpeqq[_n_s16](int16x8_t a,	b -> Rm a -> Qn	VMRS Rd,P0 VCMP.I16 eq,Qn,Rm	Rd -> result	MVE
nve_pred16_t [arm_jvcmpeqq[_n_s16](int16x8_t a, nt16_t b)	a -> Qn b -> Rm	VCMP.116 eq,Qn,Rm VMRS Rd,P0	Nu -> resuit	IVI V E
nve_pred16_t [arm_]vcmpeqq[_n_s32](int32x4_t a,	a -> Qn	VCMP.I32 eq,Qn,Rm	Rd -> result	MVE
nt32_t b)	b -> Rm	VMRS Rd,P0		
nve_pred16_t [arm_]vcmpeqq[_n_u8](uint8x16_t a,	a -> Qn	VCMP.I8 eq,Qn,Rm	Rd -> result	MVE
uint8_t b) mve pred16 t [arm]vcmpeqq[n u16](uint16x8 t a,	b -> Rm a -> Qn	VMRS Rd,P0 VCMP.I16 eq,Qn,Rm	Rd -> result	MVE
uint16_t b)	b -> Rm	VCMP.116 eq,Qii,Riii VMRS Rd,P0	Nu -> resuit	141 4 17
nve_pred16_t [arm_]vcmpeqq[_n_u32](uint32x4_t a,	a -> Qn	VCMP.I32 eq,Qn,Rm	Rd -> result	MVE
nint32_t b)	b -> Rm	VMRS Rd,P0		
nve_pred16_t [arm_]vcmpeqq_m[_f16](float16x8_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
loat16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCMPT.F16 eq,Qn,Qm		
	h -> Kh	VCMP1.F16 eq,Qn,Qm VMRS Rd,P0		
nve_pred16_t [arm_]vcmpeqq_m[_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.F32 eq,Qn,Qm		
myo prod16 + f orm lyomposs and s01/:n40v16 + -	0 > Or	VMRS Rd,P0	Dd > manule	MVE
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	IVI V E
	p -> Rp	VCMPT.I8 eq,Qn,Qm		
	1 * *	VMRS Rd,P0		1

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpeqq_m[_s16](int16x8_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.I16 eq,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_s32](int32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
nt32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Ttu > Tesur	1,1,12
- · ·	p -> Rp	VCMPT.I32 eq,Qn,Qm		
		VMRS Rd,P0		
nve_pred16_t [arm_]vcmpeqq_m[_u8](uint8x16_t a,	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rd -> result	MVE
uint8x16_t b, mve_pred16_t p)	p -> Rp	VCMPT.I8 eq.On,Om		
	r · · · · ·	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.I16 eq,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.I32 eq,Qn,Qm		
	0::	VMRS Rd,P0	D.1 >16	MVE
mve_pred16_t [arm_]vcmpeqq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Rd -> result	MVE
a, noatro_t o, invo_prearo_t p)	p -> Rp	VCMPT.F16 eq,Qn,Rm		
		VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_n_f32](float32x4_t	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
a, float32_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.F32 eq,Qn,Rm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_n_s8](int8x16_t a,	a -> On	VMSR P0,Rp	Rd -> result	MVE
int8_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.I8 eq,Qn,Rm		
116 (1) 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	. 0	VMRS Rd,P0	70.1	MATE
mve_pred16_t [arm_]vcmpeqq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Rd -> result	MVE
miro_t o, mve_predro_t p)	p -> Rn	VCMPT.I16 eq,Qn,Rm		
	r	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_n_s32](int32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.I32 eq,Qn,Rm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_n_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
uint8_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.I8 eq,Qn,Rm		
mve_pred16_t [arm_]vcmpeqq_m[_n_u16](uint16x8_t	a -> On	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
a, uint16_t b, mve_pred16_t p)	b -> Rm	VPST	Ku -> icsuit	MVE
.,,F F/	p -> Rp	VCMPT.I16 eq,Qn,Rm		
		VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_n_u32](uint32x4_t	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
a, uint32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VCMPT.I32 eq,Qn,Rm		
	p -> Kp	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpneq[_f16](float16x8_t a,	a -> Qn	VCMP.F16 ne,Qn,Qm	Rd -> result	MVE
float16x8_t b)	b -> Qm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpneq[_f32](float32x4_t a,	a -> Qn	VCMP.F32 ne,Qn,Qm	Rd -> result	MVE
float32x4_t b) mve_pred16_t [arm_]vcmpneq[_s8](int8x16_t a,	b -> Qm a -> Qn	VMRS Rd,P0 VCMP.I8 ne,Qn,Qm	Rd -> result	MVE
int8x16_t b)	b -> Qm	VMRS Rd,P0	Nu -/ ICSUIT	141 4 15
mve_pred16_t [arm_]vcmpneq[_s16](int16x8_t a,	a -> Qn	VCMP.I16 ne,Qn,Qm	Rd -> result	MVE
int16x8_t b)	b -> Qm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpneq[_s32](int32x4_t a,	a -> Qn	VCMP.I32 ne,Qn,Qm	Rd -> result	MVE
int32x4_t b) mve_pred16_t [arm_]vcmpneq[_u8](uint8x16_t a,	b -> Qm a -> Qn	VMRS Rd,P0 VCMP.I8 ne,Qn,Qm	Rd -> result	MVE
uint8x16_t b)	b -> Qn	VMRS Rd,P0	Ku -> resuit	141 4 17
mve_pred16_t [arm_]vcmpneq[_u16](uint16x8_t a,	a -> Qn	VCMP.I16 ne,Qn,Qm	Rd -> result	MVE
uint16x8_t b)	b -> Qm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpneq[_u32](uint32x4_t a,	a -> Qn	VCMP.I32 ne,Qn,Qm	Rd -> result	MVE
uint32x4_t b) mve_pred16_t [arm_]vcmpneq_m[_f16](float16x8_t a,	b -> Qm a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR PU,RP VPST	Ku -> resuit	MIVE
	p -> Rp	VCMPT.F16 ne,Qn,Qm		
		VMRS Rd,P0		
mve_pred16_t [arm_]vcmpneq_m[_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
float32x4_t b, mve_pred16_t p)	1 . 0			
	b -> Qm p -> Rp	VPST VCMPT.F32 ne,Qn,Qm		

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.I8 ne,Qn,Qm	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	VMRS Rd,P0 VMSR P0,Rp VPST VCMPT.I16 ne,Qn,Qm	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	VMRS Rd,P0 VMSR P0,Rp VPST VCMPT.I32 ne,Qn,Qm	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	VMRS Rd,P0 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.116 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.132 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	Rd -> result	MVE
$\label{eq:mve_pred16_t} \begin{split} & mve_pred16_t \ [_arm_]vcmpneq_m[_n_f32](float32x4_t \\ & a, \ float32_t \ b, \ mve_pred16_t \ p) \end{split}$	a -> Qn b -> Rm p -> Rp	VMRS Rd,P0 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
$\label{eq:mve_pred16_t} $$mve_pred16_t [_arm_]vcmpneq_m[_n_s16](int16x8_t \ a, int16_t \ b, mve_pred16_t \ p)$$	a -> Qn b -> Rm p -> Rp	VMRS Rd,P0 VMSR P0,Rp VPST VCMPT.I16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
$\label{eq:mve_pred16_t} $$mve_pred16_t [_arm_]vcmpneq_m[_n_s32](int32x4_t \ a, int32_t \ b, mve_pred16_t \ p)$$	a -> Qn b -> Rm p -> Rp	VMRS R0,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	VMRS R0,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 -> Qm	VCMP.S16 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgeq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VCMP.S32 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgeq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 ge,Qn,Qm	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMRS Rd,P0 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	VMRS RQ,F0 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,	a -> Qn b -> Rm	VCMP.S32 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
int32_t b) mve_pred16_t [arm_]vcmpgeq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMRS RU,FO VMSR PO,Rp VPST VCMPT.F16 ge,Qn,Rm VMRS Rd,PO	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 PO,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 RO,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 PO,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	VMRS RQ,10 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	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpgtq_m[_s8](int8x16_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.S8 gt,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpgtq_m[_s16](int16x8_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.S16 gt,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpgtq_m[_s32](int32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
nt32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Tu > Tobali	111 / 12
	p -> Rp	VCMPT.S32 gt,Qn,Qm		
116 (1) (16)(0) (16) (. 0	VMRS Rd,P0	D1 . 1	MIE
mve_pred16_t [arm_]vcmpgtq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VCMP.F16 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_n_f32](float32x4_t a,	a -> Qn	VCMP.F32 gt,Qn,Rm	Rd -> result	MVE
float32_t b)	b->Rm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpgtq[_n_s8](int8x16_t a,	a -> Qn	VCMP.S8 gt,Qn,Rm	Rd -> result	MVE
int8_t b) mve_pred16_t [arm_]vcmpgtq[_n_s16](int16x8_t a,	b -> Rm a -> On	VMRS Rd,P0 VCMP.S16 gt,Qn,Rm	Rd -> result	MVE
int16_t b)	b -> Rm	VMRS Rd,P0	Ku -> iesuit	MIVE
mve_pred16_t [arm_]vcmpgtq[_n_s32](int32x4_t a,	a -> Qn	VCMP.S32 gt,Qn,Rm	Rd -> result	MVE
int32_t b)	b -> Rm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpgtq_m[_n_f16](float16x8_t	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
a, float16_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VCMPT.F16 gt,Qn,Rm		
	p -> Kp	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpgtq_m[_n_f32](float32x4_t	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
a, float32_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.F32 gt,Qn,Rm		
mve_pred16_t [arm_]vcmpgtq_m[_n_s8](int8x16_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
int8_t b, mve_pred16_t p)	b -> Rm	VPST	Ru -> result	MVE
- ' - ' - '	p -> Rp	VCMPT.S8 gt,Qn,Rm		
		VMRS Rd,P0	2.1	2.07.00
mve_pred16_t [arm_]vcmpgtq_m[_n_s16](int16x8_t a,	a -> Qn b -> Rm	VMSR P0,Rp VPST	Rd -> result	MVE
int16_t b, mve_pred16_t p)	p -> Rp	VCMPT.S16 gt,Qn,Rm		
	p > Kp	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpgtq_m[_n_s32](int32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.S32 gt,Qn,Rm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpleq[_f16](float16x8_t a,	a -> Qn	VCMP.F16 le,On,Om	Rd -> result	MVE
float16x8_t b)	b->Qm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpleq[_f32](float32x4_t a,	a -> Qn	VCMP.F32 le,Qn,Qm	Rd -> result	MVE
float32x4_t b) mve_pred16_t [arm_]vcmpleq[_s8](int8x16_t a,	b -> Qm a -> On	VMRS Rd,P0 VCMP.S8 le,Qn,Qm	Rd -> result	MVE
int8x16 t b)	b -> Om	VMRS Rd,P0	Ku -> iesuit	MIVE
mve_pred16_t [arm_]vcmpleq[_s16](int16x8_t a,	a -> Qn	VCMP.S16 le,Qn,Qm	Rd -> result	MVE
int16x8_t b)	b -> Qm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpleq[_s32](int32x4_t a,	a -> Qn	VCMP.S32 le,Qn,Qm	Rd -> result	MVE
int32x4_t b) mve_pred16_t [arm_]vcmpleq_m[_f16](float16x8_t a,	b -> Qm a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
float16x8 t b, mve pred16 t p)	b -> Qm	VPST	Ku -> icsuit	MIVE
	p -> Rp	VCMPT.F16 le,Qn,Qm		
		VMRS Rd,P0	-	2.07.00
mve_pred16_t [arm_]vcmpleq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rd -> result	MVE
noat32x4_t b, nive_pred16_t p)	p -> QIII	VCMPT.F32 le,Qn,Qm		
	F	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpleq_m[_s8](int8x16_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.S8 le,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpleq_m[_s16](int16x8_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.S16 le,Qn,Qm		
myo prod16 t [orm lyomples of s201/int22v4 t -	0 > 0=	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	VMSR P0,Rp VPST	Ku -> resuit	1V1 V E
	p -> Rp	VCMPT.S32 le,Qn,Qm		
		VMRS Rd,P0		
mve_pred16_t [arm_]vcmpleq[_n_f16](float16x8_t a,	a -> Qn	VCMP.F16 le,Qn,Rm	Rd -> result	MVE
float16_t b) mve_pred16_t [arm_]vcmpleq[_n_f32](float32x4_t a,	b -> Rm	VMRS Rd,P0 VCMP.F32 le,Qn,Rm	Dd > manula	MVE
nive predito () ann jvempled n 132 (110at32X4 ta,	a -> Qn	v Civir.roz ie,Qn,Km	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	VMSR P0,Rp VPST	Rd -> result	MVE
a, noatro_t b, nive_predito_t p)	p -> Rp	VCMPT.F16 le,Qn,Rm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpleq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Rd -> result	MVE
a, noat52_t b, nive_pred16_t p)	p -> Rm p -> Rp	VCMPT.F32 le,Qn,Rm		
mve_pred16_t [arm_]vcmpleq_m[_n_s8](int8x16_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
int8_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VCMPT.S8 le,Qn,Rm		
mve_pred16_t [arm_]vcmpleq_m[_n_s16](int16x8_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
int16_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VCMPT.S16 le,Qn,Rm		
mve_pred16_t [arm_]vcmpleq_m[_n_s32](int32x4_t a,	a -> On	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VCMPT.S32 le,Qn,Rm		
mve_pred16_t [arm_]vcmpltq[_f16](float16x8_t a,	a -> Qn	VMRS Rd,P0 VCMP.F16 lt,On,Om	Rd -> result	MVE
float16x8_t b)	b -> Qm	VMRS Rd,P0		
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	VMSR P0,Rp VPST	Rd -> result	MVE
noatroxo_t o, mvc_predio_t p)	p -> Rp	VCMPT.F16 lt,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpltq_m[_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCMPT.F32 lt,Qn,Qm		
mve_pred16_t [arm_]vcmpltq_m[_s8](int8x16_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
int8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCMPT.S8 lt,Qn,Qm		
mve_pred16_t [arm_]vcmpltq_m[_s16](int16x8_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCMPT.S16 lt,Qn,Qm		
mve_pred16_t [arm_]vcmpltq_m[_s32](int32x4_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCMPT.S32 lt,Qn,Qm	Ku -> resuit	WIVE
and the state of t		VMRS Rd,P0	D.1	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	VMSR P0,Rp VPST	Rd -> result	MVE
nouro_t o, invo_picuro_t p)	p -> Rp	VCMPT.F16 lt,Qn,Rm		
mve_pred16_t [_arm_]vcmpltq_m[_n_f32](float32x4_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
float32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VCMPT.F32 lt,Qn,Rm		
		VMRS Rd,P0		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpltq_m[_n_s8](int8x16_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int8_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.S8 lt,Qn,Rm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpltq_m[_n_s16](int16x8_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int16_t b, mve_pred16_t p)	b -> Rm	VPST		·
	p -> Rp	VCMPT.S16 lt,Qn,Rm		
116.15		VMRS Rd,P0	D1 1	No.
nve_pred16_t [arm_]vcmpltq_m[_n_s32](int32x4_t a, nt32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Rd -> result	MVE
its2_t b, inve_pred16_t p)	p -> Rm	VCMPT.S32 lt,On,Rm		
	Prop	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpcsq[_u8](uint8x16_t a,	a -> Qn	VCMP.U8 cs,Qn,Qm	Rd -> result	MVE
uint8x16_t b)	b -> Qm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpcsq[_u16](uint16x8_t a,	a -> Qn	VCMP.U16 cs,Qn,Qm	Rd -> result	MVE
uint16x8_t b) mve_pred16_t [arm_]vcmpcsq[_u32](uint32x4_t a,	b -> Qm a -> On	VMRS Rd,P0 VCMP.U32 cs,Qn,Qm	Rd -> result	MVE
uint32x4_t b)	b -> Qm	VMRS Rd,P0	Ru -> icsuit	WIVE
mve_pred16_t [arm_]vcmpcsq_m[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.U8 cs,Qn,Qm		
116 (1	a -> On	VMRS Rd,P0	D.1 1:	MVE
mve_pred16_t [arm_]vcmpcsq_m[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rd -> result	MVE
unitroxo_t b, inve_predio_t p)	p -> Rp	VCMPT.U16 cs,Qn,Qm		
	r · ··r	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpcsq_m[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.U32 cs,Qn,Qm		
mve_pred16_t [arm_]vcmpcsq[_n_u8](uint8x16_t a,	a -> Qn	VMRS Rd,P0 VCMP.U8 cs,Qn,Rm	Rd -> result	MVE
uint8_t b)	b -> Rm	VMRS Rd,P0	Ku -> icsuit	IVIVE
mve_pred16_t [arm_]vcmpcsq[_n_u16](uint16x8_t a,	a -> Qn	VCMP.U16 cs,Qn,Rm	Rd -> result	MVE
uint16_t b)	b -> Rm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpcsq[_n_u32](uint32x4_t a,	a -> Qn	VCMP.U32 cs,Qn,Rm	Rd -> result	MVE
uint32_t b)	b -> Rm	VMRS Rd,P0	D.1 1	MATE
mve_pred16_t [arm_]vcmpcsq_m[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Rd -> result	MVE
unito_t b, inve_pred10_t p)	p -> Rp	VCMPT.U8 cs,Qn,Rm		
	r · ··r	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpcsq_m[_n_u16](uint16x8_t	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
a, uint16_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.U16 cs,Qn,Rm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpcsq_m[_n_u32](uint32x4_t	a -> On	VMSR P0,Rp	Rd -> result	MVE
a, uint32_t b, mve_pred16_t p)	b -> Rm	VPST	rta > result	WY E
	p -> Rp	VCMPT.U32 cs,Qn,Rm		
		VMRS Rd,P0		
mve_pred16_t [arm_]vcmphiq[_u8](uint8x16_t a,	a -> Qn	VCMP.U8 hi,Qn,Qm	Rd -> result	MVE
uint8x16_t b) mve_pred16_t [arm_]vcmphiq[_u16](uint16x8_t a,	b -> Qm a -> Qn	VMRS Rd,P0 VCMP.U16 hi,Qn,Qm	Rd -> result	MVE
uint16x8_t b)	b -> Qm	VMRS Rd,P0	Ku -> iesuit	MIVE
mve_pred16_t [arm_]vcmphiq[_u32](uint32x4_t a,	a -> Qn	VCMP.U32 hi,Qn,Qm	Rd -> result	MVE
uint32x4_t b)	b -> Qm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmphiq_m[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST VCMPT.U8 hi,On,Om		
	p -> Rp	VMRS Rd,P0		
mve_pred16_t [arm_]vcmphiq_m[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VCMPT.U16 hi,Qn,Qm		
	0	VMRS Rd,P0	D.I. I	MVE
mve_pred16_t [arm_]vcmphiq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rd -> result	MVE
unito 2x-1 to, nive_procito_t p)	p -> QIII	VCMPT.U32 hi,Qn,Qm		
	rr	VMRS Rd,P0		
mve_pred16_t [arm_]vcmphiq[_n_u8](uint8x16_t a,	a -> Qn	VCMP.U8 hi,Qn,Rm	Rd -> result	MVE
uint8_t b)	b -> Rm	VMRS Rd,P0		1.57
mve_pred16_t [arm_]vcmphiq[_n_u16](uint16x8_t a,	a -> Qn	VCMP.U16 hi,Qn,Rm	Rd -> result	MVE
uint16_t b) mve_pred16_t [arm_]vcmphiq[_n_u32](uint32x4_t a,	b -> Rm a -> Qn	VMRS Rd,P0 VCMP.U32 hi,Qn,Rm	Rd -> result	MVE
mve_pred16_t [arm_jvcmpmq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VMRS Rd,P0	Ku -> resuit	IVI V L
mve_pred16_t [arm_]vcmphiq_m[_n_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
uint8_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VCMPT.U8 hi,Qn,Rm		
		VMRS Rd,P0		

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)	b -> Rm	VPST		
	p -> Rp	VCMPT.U16 hi,Qn,Rm VMRS Rd,P0		
mve_pred16_t [arm_]vcmphiq_m[_n_u32](uint32x4_t	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
a, uint32_t b, mve_pred16_t p)	b -> Rm	VPST		1
	p -> Rp	VCMPT.U32 hi,Qn,Rm		
int8x16_t [arm_]vminq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VMRS Rd,P0 VMIN.S8 Qd,Qn,Qm	Od -> result	MVE/NEON
mtox10_t [amjvmmq[_so](mtox10_t a, mtox10_t b)	b -> Qm	VIVIIIV.36 Qu,Qii,Qiii	Qu -> result	WIVE/INDOIN
int16x8_t [arm_]vminq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VMIN.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
22.4.5.	b->Qm	VI (IV (22) 0.1 0.0	01 1	MENERY
nt32x4_t [arm_]vminq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Om	VMIN.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vminq[_u8](uint8x16_t a, uint8x16_t	a -> Qn	VMIN.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
p)	b -> Qm			
uint16x8_t [arm_]vminq[_u16](uint16x8_t a, uint16x8_t	a -> Qn	VMIN.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
b) uint32x4_t [arm_]vminq[_u32](uint32x4_t a, uint32x4_t	b -> Qm a -> Qn	VMIN.U32 Qd,Qn,Qm	Od -> result	MVE/NEON
mit.52x4_t [arm_jvininq[_u52](umt.52x4_t a, umt.52x4_t b)	b -> Qm	VIVIIIV.032 Qd,Qii,Qiii	Qu -> resuit	WIVE/NEON
nt8x16_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	VPST		
	b -> Qm p -> Rp	VMINT.S8 Qd,Qn,Qm		
nt16x8_t [arm_]vminq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
nt16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST	Qu	1
	b -> Qm	VMINT.S16 Qd,Qn,Qm		
	p -> Rp	VMCD DO D.	0.1	MVE
nt32x4_t [_arm_]vminq_m[_s32](int32x4_t inactive, nt32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
me_nte_t a, me_nte_predio_t p/	b -> Qm	VMINT.S32 Qd,Qn,Qm		
	p -> Rp			
uint8x16_t [arm_]vminq_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VMINT.U8 Qd,Qn,Qm		
	p -> Rp	VIIII VI . 00 Qu, QII, QIII		
uint16x8_t [arm_]vminq_m[_u16](uint16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VMINT.U16 Qd,Qn,Qm		
uint32x4_t [arm_]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	VPST		
	b -> Qm	VMINT.U32 Qd,Qn,Qm		
int8x16_t [arm_]vminq_x[_s8](int8x16_t a, int8x16_t b,	p -> Rp a -> Qn	VMSR P0,Rp	Od -> result	MVE
mtox10_t [atm]vmmq_x[_so](mtox10_t a, mtox10_t b, mve_pred16_t p)	b -> Qm	VPST	Qu -> resurt	WYE
	p -> Rp	VMINT.S8 Qd,Qn,Qm		
nt16x8_t [arm_]vminq_x[_s16](int16x8_t a, int16x8_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
o, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINT.S16 Qd,Qn,Qm		
nt32x4_t [arm_]vminq_x[_s32](int32x4_t a, int32x4_t	a -> Qn	VMSR P0,Rp	Od -> result	MVE
o, mve_pred16_t p)	b -> Qm	VPST		·
	p -> Rp	VMINT.S32 Qd,Qn,Qm		
tint8x16_t [arm_]vminq_x[_u8](uint8x16_t a, tint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mitox10_t b, nive_pred10_t p)	p -> Rp	VMINT.U8 Od,On,Om		
nint16x8_t [arm_]vminq_x[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
nint16x8_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
$\lim_{z \to +} 2x + \lim_{z \to +} \lim_{z \to +} 2x + \lim_{z \to +}$	b -> Qm	VMSK FO,KP VPST	Qu -> resuit	WIVE
2.1)	p -> Rp	VMINT.U32 Qd,Qn,Qm		
nint8x16_t [arm_]vminaq[_s8](uint8x16_t a, int8x16_t	a -> Qda	VMINA.S8 Qda,Qm	Qda -> result	MVE
b) iint16x8_t [arm_]vminaq[_s16](uint16x8_t a, int16x8_t	b -> Qm	VMINA.S16 Qda,Qm	Ode > regult	MVE
iintiox8_t [arm_jvminaq[_s16](uint16x8_t a, int16x8_t	a -> Qda b -> Qm	viviliva.510 Qda,Qiii	Qda -> result	IVI V E
nint32x4_t [arm_]vminaq[_s32](uint32x4_t a, int32x4_t	a -> Qda	VMINA.S32 Qda,Qm	Qda -> result	MVE
n)	b -> Qm			
uint8x16_t [_arm_]vminaq_m[_s8](uint8x16_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
nt8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINAT.S8 Qda,Qm		
uint16x8_t [arm_]vminaq_m[_s16](uint16x8_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
nt16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VMINAT.S16 Qda,Qm		1
uint32x4_t [arm_]vminaq_m[_s32](uint32x4_t a, nt32x4_t b, mve_pred16_t p)	a -> Qda b -> Qm	VMSR P0,Rp VPST	Qda -> result	MVE
	1 D = 2 UIII	1 1191	1	1

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	VMINV.S32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vminvq[_u8](uint8_t a, uint8x16_t b)	b -> Qm 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,	b -> Qm a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINVT.S8 Rda,Qm		
int16_t [arm_]vminvq_p[_s16](int16_t a, int16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
int32_t [arm_]vminvq_p[_s32](int32_t a, int32x4_t b,	p -> Rp a -> Rda	VMINVT.S16 Rda,Qm 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,	a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
mve_pred16_t p)	p -> Rp	VMINVT.U8 Rda,Qm		
uint16_t [_arm_]vminvq_p[_u16](uint16_t a, uint16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
uint32_t [arm_]vminvq_p[_u32](uint32_t a, uint32x4_t	p -> Rp a -> Rda	VMINVT.U16 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST VMINVT.U32 Rda,Om	raa > resur	W. C.
uint8_t [arm_]vminavq[_s8](uint8_t a, int8x16_t b)	p -> Rp a -> Rda	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,Qm	Rda -> result	MVE
uint8_t [arm_]vminavq_p[_s8](uint8_t a, int8x16_t b,	b -> Qm a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINAVT.S8 Rda,Qm		
uint16_t [_arm_]vminavq_p[_s16](uint16_t a, int16x8_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINAVT.S16 Rda,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	Od -> result	MVE/NEON
float16x8_t b)	b -> Qm			
float32x4_t [_arm_]vminnmq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VMINNM.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [_arm_]vminnmq_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	VMINNMT.F16 Qd,Qn,Qm		
float32x4_t [_arm_]vminnmq_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 VMINNMT.F32 Qd,Qn,Qm		
float16x8_t [arm_]vminnmq_x[_f16](float16x8_t a,	p -> Rp a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMINNMT.F16 Qd,Qn,Qm		
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
_ · · -• - • · · ·	p -> Rp	VMINNMT.F32 Qd,Qn,Qm		
float16x8_t [arm_]vminnmaq[_f16](float16x8_t a, float16x8_t b)	a -> Qda b -> Qm	VMINNMA.F16 Qda,Qm	Qda -> result	MVE
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
	p -> Rp	VMINNMAT.F16 Qda,Qm		1
float32x4_t [_arm_]vminnmaq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qda b -> Qm	VMSR P0,Rp VPST	Qda -> result	MVE
float16_t [arm_]vminnmvq[_f16](float16_t a,	p -> Rp a -> Rda	VMINNMAT.F32 Qda,Qm VMINNMV.F16 Rda,Qm	Rda -> result	MVE
float16x8_t b)	b -> Qm	, ,		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32_t [_arm_]vminnmvq[_f32](float32_t a, float32x4_t b)	a -> Rda b -> Qm	VMINNMV.F32 Rda,Qm	Rda -> result	MVE
float16_t [_arm_]vminnmvq_p[_f16](float16_t a, float16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINNMVT.F16 Rda,Qm	Rda -> result	MVE
float32_t [arm_]vminnmvq_p[_f32](float32_t a, float32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINNMVT.F32 Rda,Qm	Rda -> result	MVE
float16_t [_arm_]vminnmavq[_f16](float16_t a, float16x8_t b)	a -> Rda b -> Qm	VMINNMAV.F16 Rda,Qm	Rda -> result	MVE
float32_t [_arm_]vminnmavq[_f32](float32_t a, float32x4_t b)	a -> Rda b -> Qm	VMINNMAV.F32 Rda,Qm	Rda -> result	MVE
float16_t [_arm_]vminnmavq_p[_f16](float16_t a, 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	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 p -> Rp	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)	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 -> Qm	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 p -> Rp	VMSR P0,Rp VPST VMAXAVT.S16 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vmaxavq_p[_s32](uint32_t a, int32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	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	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)	p -> Rp a -> Qda b -> Qm	VMSR P0,Rp VPST	Qda -> result	MVE
float16_t [arm_]vmaxnmvq[_f16](float16_t a, float16x8_t b)	p -> Rp a -> Rda b -> Qm	VMAXNMAT.F32 Qda,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	VMSR P0,Rp VPST	Rda -> result	MVE
float16_t [_arm_]vmaxnmavq[_f16](float16_t a, float16x8_t b)	p -> Rp a -> Rda b -> Qm	VMAXNMVT.F32 Rda,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	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)	p -> Rp 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)	p -> Rp 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 -> Qm	VABD.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vabdq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vabdq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vabdq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [arm_]vabdq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	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)	p -> Rp 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)	a -> Qm	VABS.S8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vabsq[_s16](int16x8_t a)	a -> Qm	VABS.S16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vabsq[_s32](int32x4_t a)	a -> Qm	VABS.S32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vabsq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VABST.532 Qd,Qm VMSR P0,Rp VPST VABST.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vabsq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VABST.F10 Qd,Qlll VMSR P0,Rp VPST VABST.F32 Qd,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vabsq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t [_arm_]vabsq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm p -> Rp	VABST.S8 Qd,Qm VMSR P0,Rp VPST VABST.S16 Qd,Qm	Qd -> result	MVE
int32x4_t [arm_]vabsq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S32 Qd,Qm	Qd -> result	MVE
float16x8_t [_arm_]vabsq_x[_f16](float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vabsq_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.F32 Qd,Qm	Qd -> result	MVE
int8x16_t [_arm_]vabsq_x[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S8 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vabsq_x[_s16](int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S16 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.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
uint32x4_t [_arm_]vadciq[_u32](uint32x4_t a, uint32x4_t b, unsigned * carry_out)	a -> Qn b -> Qm	VADCI.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
int32x4_t [_arm_]vadciq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, unsigned * carry_out, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADCIT.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
uint32x4_t [_arm_]vadciq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, unsigned * carry_out, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADCIT.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_nzevqe BFI Rs,Rt,#29,#1 VMSR FPSCR_nzevqe,Rs VADC.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqe 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 PO,Rp VPST VADCT.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
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.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
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	VMSR P0,Rp VPST	Qd -> result	MVE
noacrono_t a, noacrono_t o, inve_predio_t p)	b -> Qm p -> Rp	VADDT.F16 Qd,Qn,Qm		
float32x4_t [_arm_]vaddq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
noacs2x4_t a, noacs2x4_t o, inve_pred1c_t p)	b -> Qm p -> Rp	VADDT.F32 Qd,Qn,Qm		
float16x8_t [_arm_]vaddq_m[_n_f16](float16x8_t inactive, float16x8_t a, float16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
macure, neartono_t a, nearto_t e, mre_preare_t p/	b -> Rm p -> Rp	VADDT.F16 Qd,Qn,Rm		
float32x4_t [arm_]vaddq_m[_n_f32](float32x4_t inactive, float32x4_t a, float32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Rm p -> Rp	VADDT.F32 Qd,Qn,Rm		
int8x16_t [_arm_]vaddq_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
	b -> Qm p -> Rp	VADDT.I8 Qd,Qn,Qm		
int16x8_t [_arm_]vaddq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qm p -> Rp	VADDT.I16 Qd,Qn,Qm		
int32x4_t [_arm_]vaddq_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
2.0, 2.0, 2.1,	b -> Qm p -> Rp	VADDT.I32 Qd,Qn,Qm		
int8x16_t [_arm_]vaddq_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 p -> Rp	VADDT.I8 Qd,Qn,Rm		
int16x8_t [_arm_]vaddq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
- / - / - / - F/	b -> Rm p -> Rp	VADDT.I16 Qd,Qn,Rm		
int32x4_t [_arm_]vaddq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
- / - / - / - F/	b -> Rm p -> Rp	VADDT.I32 Qd,Qn,Rm		
uint8x16_t [_arm_]vaddq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qm p -> Rp	VADDT.I8 Qd,Qn,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vaddq_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	VADDT.I16 Qd,Qn,Qm		
	p -> Rp			
uint32x4_t [arm_]vaddq_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	VPST		
	b -> Qm	VADDT.I32 Qd,Qn,Qm		
	p -> Rp			
uint8x16_t [_arm_]vaddq_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	VADDT.I8 Qd,Qn,Rm		
		VMCD DO D.	Od -> result	MAZE
uint16x8_t [_arm_]vaddq_m[_n_u16](uint16x8_t	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qu -> resuit	MVE
nactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	b -> Rm	VADDT.I16 Qd,Qn,Rm		
	p -> Rp	VADD1:110 Qu,Qii,Kiii		
uint32x4_t [arm_]vaddq_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 -> On	VPST	Qu -> result	MIVE
maeave, amaeza i_t a, amaez_t o, mve_prearo_t p	b -> Rm	VADDT.I32 Qd,Qn,Rm		
	p -> Rp	77155 7.152 Qu,Qu,tem		
float16x8_t [arm_]vaddq_x[_f16](float16x8_t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Zu > Icsuit	,.
	p -> Rp	VADDT.F16 Qd,Qn,Qm		
float32x4_t [arm_]vaddq_x[_f32](float32x4_t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Qu > resun	11111
	p -> Rp	VADDT.F32 Qd,Qn,Qm		
float16x8_t [arm_]vaddq_x[_n_f16](float16x8_t a,	a -> On	VMSR P0.Rp	Od -> result	MVE
float16x6_t [armjvaddq_x[_n_110](float16x8_t a, float16_t b, mve_pred16_t p)	b -> Rm	VMSK FO,KP VPST	Zu -> Icsuit	171 7 L
mourro_t b, mvc_predro_t p)	p -> Rp	VADDT.F16 Qd,Qn,Rm		
float32x4_t [arm_]vaddq_x[_n_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Od -> result	MVE
float32_t b, mve_pred16_t p)	b -> Rm	VPST	Qu > result	
moat32_t b, mvc_picu10_t p)	p -> Rp	VADDT.F32 Qd,Qn,Rm		
int8x16_t [arm_]vaddq_x[_s8](int8x16_t a, int8x16_t b,	a -> Qn	VMSR P0,Rp	Od -> result	MVE
mtox10_t [arm_]vaddq_x[_so](mtox10_t a, mtox10_t b, mve_pred16_t p)	b -> Qm	VPST	Qu -> result	WIVE
mve_pred16_t p)	p -> Rp	VADDT.I8 Qd,Qn,Qm		
int16x8_t [arm_]vaddq_x[_s16](int16x8_t a, int16x8_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST	Qu => resuit	WIVE
b, filve_pred16_t p)	p -> Rp	VADDT.I16 Qd,Qn,Qm		
int32x4_t [arm_]vaddq_x[_s32](int32x4_t a, int32x4_t	a -> On	VMSR P0,Rp	Od -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST	Qu => result	WIVE
o, mve_predro_t p/	p -> Rp	VADDT.I32 Qd,Qn,Qm		
int8x16_t [arm_]vaddq_x[_n_s8](int8x16_t a, int8_t b,	a -> On	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	b -> Rm	VPST	Qu > result	III V E
m,e_predic_t p/	p -> Rp	VADDT.I8 Qd,Qn,Rm		
int16x8 t [arm]vaddq x[n s16](int16x8 t a, int16 t	a -> On	VMSR P0,Rp	Od -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST	Qu' > Tesant	111,12
o, mve_predro_tp)	p -> Rp	VADDT.I16 Qd,Qn,Rm		
int32x4_t [arm_]vaddq_x[_n_s32](int32x4_t a, int32_t	a -> On	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST	Q	1
	p -> Rp	VADDT.I32 Qd,Qn,Rm		
uint8x16_t [arm_]vaddq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST	Q	
- / I/	p -> Rp	VADDT.I8 Qd,Qn,Qm		
uint16x8_t [arm_]vaddq_x[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Od -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VADDT.I16 Qd,Qn,Qm		
uint32x4_t [arm_]vaddq_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	VADDT.I32 Qd,Qn,Qm		
uint8x16_t [arm_]vaddq_x[_n_u8](uint8x16_t a, uint8_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VADDT.I8 Qd,Qn,Rm		
uint16x8_t [arm_]vaddq_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	VADDT.I16 Qd,Qn,Rm		
uint32x4_t [arm_]vaddq_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/	p -> Rp	VADDT.I32 Qd,Qn,Rm		
int8x16_t [arm_]vclsq[_s8](int8x16_t a)	a -> Qm	VCLS.S8 Qd,Qm	Qd -> result	MVE/NEON
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 -> Om	VPST	2 - 105uit	1
	p -> Rp	VCLST.S8 Qd,Qm		
				-1
int16x8 t[arm lyclsq mf s16l(int16x8 tinactive		VMSR P0 Rn	Od -> result	MVE
int16x8_t [_arm_]vclsq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE

ntrinsic	Argument Preparation	Instruction	Result	Supported Architecture
nt32x4_t [arm_]vclsq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
nt32x4_t a, mve_pred16_t p)	a -> Om	VPST	Qu > resurt	W. C.
mozni i di miro prodro i pi	p -> Rp	VCLST.S32 Qd,Qm		
nt8x16_t [arm_]vclsq_x[_s8](int8x16_t a,	a -> Om	VMSR P0,Rp	Od -> result	MVE
nve_pred16_t p)	p -> Rp	VPST	Qu > resun	111 1 2
	r ·r	VCLST.S8 Qd,Qm		
nt16x8_t [arm_]vclsq_x[_s16](int16x8_t a,	a -> Om	VMSR P0,Rp	Od -> result	MVE
ntiox8_t [arm_jvcisq_x[_sioj(intiox8_t a, nve_pred16_t p)	p -> Rp	VPST	Qu -> resurt	IVI V L
nve_predio_t p)	p -> Kp	VCLST.S16 Od,Om		
nt32x4 t [arm]vclsq x[s32](int32x4 t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
nve pred16 t p)	p -> Rp	VPST	Qu -> resurt	IVI V L
nve_pred10_t p)	p -> Kp	VCLST.S32 Qd,Qm		
nt8x16 t[arm]vclzq[s8](int8x16 t a)	a > Om	VCLZ.I8 Od,Om	Qd -> result	MVE/NEON
	a -> Qm			_
nt16x8_t [_arm_]vclzq[_s16](int16x8_t a)	a -> Qm	VCLZ.I16 Qd,Qm	Qd -> result	MVE/NEON
nt32x4_t [arm_]vclzq[_s32](int32x4_t a)	a -> Qm	VCLZ.I32 Qd,Qm	Qd -> result	MVE/NEON
nint8x16_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
nint32x4_t [arm_]vclzq[_u32](uint32x4_t a)	a -> Qm	VCLZ.I32 Qd,Qm	Qd -> result	MVE/NEON
nt8x16_t [arm_]vclzq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
nt8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCLZT.I8 Qd,Qm		1
nt16x8_t [arm_]vclzq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
nt16x8_t a, mve_pred16_t p)	a -> Qm	VPST	, , , , , ,	1
	p -> Rp	VCLZT.I16 Qd,Qm		1
nt32x4_t [arm_]vclzq_m[_s32](int32x4_t inactive,	inactive -> Od	VMSR P0,Rp	Od -> result	MVE
nt32x4_t [armyerzq_m[_s32](mt32x4_t maetive, nt32x4_t a, mve_pred16_t p)	a -> Qm	VMSK FO,KP VPST	Qu =/ Icsuit	141 4 15
nt32x4_t a, nive_pied10_t p)	p -> Rp	VCLZT.I32 Qd,Qm		
iint8x16 t [arm]vclzq m[u8](uint8x16 t inactive,	inactive -> Od		Od -> result	MVE
	,	VMSR P0,Rp	Qd -> result	MVE
iint8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCLZT.I8 Qd,Qm		
iint16x8_t [arm_]vclzq_m[_u16](uint16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
iint16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCLZT.I16 Qd,Qm		
int32x4_t [arm_]vclzq_m[_u32](uint32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCLZT.I32 Qd,Qm		
nt8x16_t [arm_]vclzq_x[_s8](int8x16_t a,	a -> Om	VMSR P0,Rp	Od -> result	MVE
nve_pred16_t p)	p -> Rp	VPST		
_1 _ 1/	1 .	VCLZT.I8 Qd,Qm		
nt16x8_t [arm_]vclzq_x[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
nve_pred16_t p)	p -> Rp	VPST		
	r·r	VCLZT.I16 Qd,Qm		
nt32x4_t [arm_]vclzq_x[_s32](int32x4_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
nve_pred16_t p)	p -> Rp	VPST	Qu -> resurt	WIVE
nve_pred1o_t p)	p -> Kp	VCLZT.I32 Od,Om		
.int0v16 t [.amm]volga v[.v0](vint0v16 t a	a > Om		Qd -> result	MVE
tint8x16_t [arm_]vclzq_x[_u8](uint8x16_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
nve_pred16_t p)	p -> Rp	VPST		
		VCLZT.I8 Qd,Qm		
int16x8_t [arm_]vclzq_x[_u16](uint16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
nve_pred16_t p)	p -> Rp	VPST		
	1	VCLZT.I16 Qd,Qm		
int32x4_t [arm_]vclzq_x[_u32](uint32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
nve_pred16_t p)	p -> Rp	VPST		
		VCLZT.I32 Qd,Qm		
loat16x8_t [arm_]vnegq[_f16](float16x8_t a)	a -> Qm	VNEG.F16 Qd,Qm	Qd -> result	MVE/NEON
loat32x4_t [arm_]vnegq[_f32](float32x4_t a)	a -> Qm	VNEG.F32 Qd,Qm	Qd -> result	MVE/NEON
nt8x16_t [arm_]vnegq[_s8](int8x16_t a)	a -> Qm	VNEG.S8 Qd,Qm	Qd -> result	MVE/NEON
nt16x8_t [arm_]vnegq[_s16](int16x8_t a)	a -> Qm	VNEG.S16 Qd,Qm	Qd -> result	MVE/NEON
nt32x4_t [arm_]vnegq[_s32](int32x4_t a)	a -> Om	VNEG.S32 Qd,Qm	Qd -> result	MVE/NEON
loat16x8_t [arm_]vnegq_m[_f16](float16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE/NEON
loat16x8_t a, mve_pred16_t p)	a -> Qm	VMSK PO,KP VPST	Qu -> icsuit	1V1 V 15
toatrono_t a, mvc_preuro_t μ)	p -> Rp	VNEGT.F16 Qd,Qm		
00122v4 + [0rm]vmoqq m[f221/floot22v4 + intime			Od > =====14	MVE
loat32x4_t [_arm_]vnegq_m[_f32](float32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
loat32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VNEGT.F32 Qd,Qm		
nt8x16_t [arm_]vnegq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
	a -> Qm	VPST		
nt8x16_t a, mve_pred16_t p)		VNEGT.S8 Qd,Qm		
	p -> Rp	VIVEOT.50 Qu,QIII		
	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
nt16x8_t [arm_]vnegq_m[_s16](int16x8_t inactive,			Qd -> result	MVE
nt16x8_t [arm_]vnegq_m[_s16](int16x8_t inactive,	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
nt16x8_t [arm_]vnegq_m[_s16](int16x8_t inactive, nt16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VNEGT.S16 Qd,Qm		
nt8x16_t a, mve_pred16_t p) nt16x8_t [_arm_]vnegq_m[_s16](int16x8_t inactive, nt16x8_t a, mve_pred16_t p) nt32x4_t [_arm_]vnegq_m[_s32](int32x4_t inactive, nt32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result Qd -> result	MVE MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [arm_]vnegq_x[_f16](float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
		VNEGT.F16 Qd,Qm		
float32x4_t [arm_]vnegq_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VNEGT.F32 Qd,Qm		
int8x16_t [arm_]vnegq_x[_s8](int8x16_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VNEGT.S8 Qd,Qm		
int16x8_t [_arm_]vnegq_x[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPS1 VNEGT.S16 Qd,Qm		
int32x4_t [arm_]vnegq_x[_s32](int32x4_t a,	a -> Om	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		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
int16x8_t [arm_]vmulhq[_s16](int16x8_t a, int16x8_t b)	b -> Qm a -> On	VMULH.S16 Qd,Qn,Qm	Od -> result	MVE
introxo_t [arm_]vinumq[_sro](introxo_t a, introxo_t b)	b -> Qm	VWICEII.310 Qu,Qii,Qiii	Qu => result	WIVE
int32x4_t [arm_]vmulhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn	VMULH.S32 Qd,Qn,Qm	Qd -> result	MVE
	b -> Qm			
uint8x16_t [arm_]vmulhq[_u8](uint8x16_t a, uint8x16_t	a -> Qn	VMULH.U8 Qd,Qn,Qm	Qd -> result	MVE
b) uint16x8_t [arm_]vmulhq[_u16](uint16x8_t a,	b -> Qm a -> On	VMULH.U16 Qd,Qn,Qm	Od -> result	MVE
uint16x8_t [arm_]vmuinq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMULH.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4 t [arm]vmulhq[u32](uint32x4 t a,	a -> On	VMULH.U32 Qd,Qn,Qm	Od -> result	MVE
uint32x4_t b)	b -> Qm		Z	
int8x16_t [arm_]vmulhq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VMULHT.S8 Qd,Qn,Qm		
int16x8_t [_arm_]vmulhq_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	VMSK FO,KP VPST	Qu -> resuit	MIVE
introxo_t a, introxo_t b, invo_prearo_t p)	b -> Qm	VMULHT.S16 Qd,Qn,Qm		
	p -> Rp			
int32x4_t [arm_]vmulhq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VMULHT.S32 Qd,Qn,Qm		
uint8x16_t [arm_]vmulhq_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	Z	
	b -> Qm	VMULHT.U8 Qd,Qn,Qm		
	p -> Rp			
uint16x8_t [arm_]vmulhq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
umtrox8_t a, umtrox8_t b, mve_pred16_t p)	a -> Qn b -> Om	VMULHT.U16 Od,On,Om		
	p -> Rp	111021111010 Qu,Qu,Qu		
uint32x4_t [arm_]vmulhq_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	VPST		
	b -> Qm	VMULHT.U32 Qd,Qn,Qm		
int8x16_t [arm_]vmulhq_x[_s8](int8x16_t a, int8x16_t	p -> Rp a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qn	VPST	Qu => result	MIVE
	p -> Rp	VMULHT.S8 Qd,Qn,Qm		
int16x8_t [arm_]vmulhq_x[_s16](int16x8_t a, int16x8_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
:	p -> Rp	VMULHT.S16 Qd,Qn,Qm	0.1 >16	MVE
int32x4_t [_arm_]vmulhq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
b, mvc_picuro_t p)	p -> Rp	VMULHT.S32 Qd,Qn,Qm		
uint8x16_t [arm_]vmulhq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VMULHT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vmulhq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
umitrozo_t o, mvc_preuro_t p)		VPS1 VMULHT.U16 Qd,Qn,Qm		
			1	+
uint32x4_t [arm_]vmulhq_x[_u32](uint32x4_t a,	p -> Rp a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t [_arm_]vmulhq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULHT.U32 Qd,Qn,Qm		
uint32x4_t b, mve_pred16_t p) uint16x8_t [arm_]vmullbq_poly[_p8](uint8x16_t a,	p -> Rp a -> Qn b -> Qm p -> Rp a -> Qn	VMSR P0,Rp VPST	Qd -> result Qd -> result	MVE MVE
uint32x4_t b, mve_pred16_t p) uint16x8_t [arm_]vmullbq_poly[_p8](uint8x16_t a, uint8x16_t b)	p -> Rp a -> Qn b -> Qm p -> Rp a -> Qn b -> Qm	VMSR P0,Rp VPST VMULHT.U32 Qd,Qn,Qm VMULLB.P8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p) uint16x8_t [arm_]vmullbq_poly[_p8](uint8x16_t a, uint8x16_t b) uint32x4_t [arm_]vmullbq_poly[_p16](uint16x8_t a,	p -> Rp a -> Qn b -> Qm p -> Rp a -> Qn b -> Qm a -> Qn b -> Qm	VMSR P0,Rp VPST VMULHT.U32 Qd,Qn,Qm		
uint32x4_t b, mve_pred16_t p) uint16x8_t [arm_]vmullbq_poly[_p8](uint8x16_t a, uint8x16_t b) uint32x4_t [arm_]vmullbq_poly[_p16](uint16x8_t a, uint16x8_t b)	p -> Rp a -> Qn b -> Qm p -> Rp a -> Qn b -> Qm	VMSR P0,Rp VPST VMULHT.U32 Qd,Qn,Qm VMULLB.P8 Qd,Qn,Qm VMULLB.P16 Qd,Qn,Qm	Qd -> result Qd -> result	MVE MVE
uint32x4_t b, mve_pred16_t p) uint16x8_t [arm_]vmullbq_poly[_p8](uint8x16_t a, uint8x16_t b) uint32x4_t [arm_]vmullbq_poly[_p16](uint16x8_t a,	p -> Rp a -> Qn b -> Qm p -> Rp a -> Qn b -> Qm b -> Qm b -> Qm	VMSR P0,Rp VPST VMULHT.U32 Qd,Qn,Qm VMULLB.P8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p) uint16x8_t [_arm_]vmullbq_poly[_p8](uint8x16_t a, uint8x16_t b) uint32x4_t [_arm_]vmullbq_poly[_p16](uint16x8_t a, uint16x8_t b) int16x8_t [_arm_]vmullbq_int[_s8](int8x16_t a,	p -> Rp a -> Qn b -> Qm p -> Rp a -> Qn b -> Qm p -> Rp a -> Qn b -> Qm b -> Qm a -> Qn b -> Qm	VMSR P0,Rp VPST VMULHT.U32 Qd,Qn,Qm VMULLB.P8 Qd,Qn,Qm VMULLB.P16 Qd,Qn,Qm	Qd -> result Qd -> result	MVE MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64x2_t [arm_]vmullbq_int[_s32](int32x4_t a,	a -> Qn	VMULLB.S32 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t b) uint16x8_t [arm_]vmullbq_int[_u8](uint8x16_t a,	b -> Qm a -> Qn	VMULLB.U8 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t b) uint32x4_t [_arm_]vmullbq_int[_u16](uint16x8_t a,	b -> Qm a -> Qn	VMULLB.U16 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t b) uint64x2_t [_arm_]vmullbq_int[_u32](uint32x4_t a,	b -> Qm a -> Qn	VMULLB.U32 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t b)	b -> Qm			
uint16x8_t [arm_]vmullbq_poly_m[_p8](uint16x8_t inactive, uint8x16 t a, uint8x16 t b, mve pred16 t p)	inactive -> Qd a -> On	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 -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VMULLBT.P16 Qd,Qn,Qm		
int16x8_t [arm_]vmullbq_int_m[_s8](int16x8_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn	VPST	Qd -> resuit	MVE
	b -> Qm p -> Rp	VMULLBT.S8 Qd,Qn,Qm		
int32x4_t [_arm_]vmullbq_int_m[_s16](int32x4_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 VMULLBT.S16 Qd,Qn,Qm		
int64x2 t [arm vmullbq int m[s32](int64x2 t	p -> Rp inactive -> Od	VMSR P0,Rp	Od -> result	MVE
inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn	VPST	Qu > result	III V E
	b -> Qm p -> Rp	VMULLBT.S32 Qd,Qn,Qm		
uint16x8_t [arm_]vmullbq_int_m[_u8](uint16x8_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
macuve, umtox10_t a, umtox10_t b, mve_preu10_t p)	b -> Qm	VMULLBT.U8 Qd,Qn,Qm		
uint32x4_t [arm_]vmullbq_int_m[_u16](uint32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VMULLBT.U16 Qd,Qn,Qm		
	p -> Rp	7 . 7 . 7		
uint64x2_t [arm_]vmullbq_int_m[_u32](uint64x2_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qm p -> Rp	VMULLBT.U32 Qd,Qn,Qm		
uint16x8_t [arm_]vmullbq_poly_x[_p8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMULLBT.P8 Qd,Qn,Qm		
uint32x4_t [arm_]vmullbq_poly_x[_p16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VMULLBT.P16 Qd,Qn,Qm		
int16x8_t [arm_]vmullbq_int_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	VMULLBT.S8 Qd,Qn,Qm		100
int32x4_t [arm_]vmullbq_int_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int64x2 t [arm]vmullbq int x[s32](int32x4 t a,	p -> Rp a -> Qn	VMULLBT.S16 Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Qu -> resuit	WYL
uint16x8_t [arm_]vmullbq_int_x[_u8](uint8x16_t a,	p -> Rp a -> Qn	VMULLBT.S32 Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMULLBT.U8 Qd,Qn,Qm		
uint32x4_t [arm_]vmullbq_int_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 VMULLBT.U16 Qd,Qn,Qm		
uint64x2_t [_arm_]vmullbq_int_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 VMULLBT.U32 Qd,Qn,Qm		
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,	a -> Qn	VMULLT.P16 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t b) int16x8_t [_arm_]vmulltq_int[_s8](int8x16_t a, int8x16_t	b -> Qm a -> Qn	VMULLT.S8 Qd,Qn,Qm	Qd -> result	MVE
b) int32x4_t [_arm_]vmulltq_int[_s16](int16x8_t a,	b -> Qm a -> Qn	VMULLT.S16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t b)	b->Qm	2 . 2 . 2		
int64x2_t [arm_]vmulltq_int[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMULLT.S32 Qd,Qn,Qm	Qd -> result	MVE
111(32x4_(10)				

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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,	a -> Qn	VMULLT.U32 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t b) uint16x8_t [_arm_]vmulltq_poly_m[_p8](uint16x8_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	b -> Qm 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 tb)	a -> Qn b -> Om	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	VMUL.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vmulq[_s16](int16x8_t a, int16x8_t b)	b -> Qm a -> Qn	VMUL.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vmulq[_s32](int32x4_t a, int32x4_t b)	b -> Qm a -> Qn	VMUL.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vmulq[_n_s8](int8x16_t a, int8_t b)	b -> Qm a -> Qn b -> Rm	VMUL.I8 Qd,Qn,Rm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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	a -> Qn b -> Qm	VMUL.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
b) uint16x8_t [arm_]vmulq[_u16](uint16x8_t a, uint16x8_t	a -> Qn	VMUL.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
b) uint32x4_t [_arm_]vmulq[_u32](uint32x4_t a, uint32x4_t	b -> Qm a -> Qn	VMUL.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
b) uint8x16_t [arm_]vmulq[_n_u8](uint8x16_t a, uint8_t	b -> Qm a -> Qn	VMUL.I8 Qd,Qn,Rm	Qd -> result	MVE/NEON
b)	b -> Rm a -> Qn	VMUL.I16 Qd,Qn,Rm	Qd -> result	MVE/NEON
b)	b -> Rm a -> Qn	VMUL.I32 Qd,Qn,Rm	Qd -> result	MVE/NEON
b) float16x8_t [_arm_]vmulq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	b -> Rm inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VMULT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [arm_]vmulq_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 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
uint8x16_t [_arm_]vmulq_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 VMULT.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [arm_]vmulq_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 VMULT.I16 Qd,Qn,Rm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vmulq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm	VMSR P0,Rp VPST VMULT.I32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [arm_]vmulq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vmulq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [arm_]vmulq_x[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.F16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [_arm_]vmulq_x[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.F32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vmulq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vmulq_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vmulq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vmulq_x[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vmulq_x[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vmulq_x[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vmulq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmulq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmulq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vmulq_x[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vmulq_x[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vmulq_x[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I32 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vsbciq[_s32](int32x4_t a, int32x4_t b, unsigned * carry_out)	a -> Qn b -> Qm	VSBCI.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
uint32x4_t [_arm_]vsbciq[_u32](uint32x4_t a, uint32x4_t b, unsigned * carry_out)	a -> Qn b -> Qm	VSBCI.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
int32x4_t [arm_]vsbciq_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 VSBCIT.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
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_nzevqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vsubq_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	VSUBT.I8 Qd,Qn,Rm		
int16x8_t [arm_]vsubq_m[_n_s16](int16x8_t inactive,	p -> Rp inactive -> Od	VMSR P0,Rp	Od -> result	MVE
int16x8_t a, int16_t b, mve_pred16_t p)	a -> On	VPST	Qu => result	WIVE
	b -> Rm	VSUBT.I16 Qd,Qn,Rm		
	p -> Rp			
$int32x4_t \ [_arm_]vsubq_m[_n_s32] (int32x4_t \ inactive,$	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm p -> Rp	VSUBT.I32 Qd,Qn,Rm		
uint8x16_t [arm_]vsubq_m[_u8](uint8x16_t inactive,	inactive -> Od	VMSR P0,Rp	Od -> result	MVE
uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn	VPST	Qu'y resun	11112
- · · - · - · - ·	b -> Qm	VSUBT.I8 Qd,Qn,Qm		
	p -> Rp			
uint16x8_t [arm_]vsubq_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 VSUBT.I16 Qd,Qn,Qm		
	p -> Rp	V30B1.110 Qu,Qii,Qiii		
uint32x4 t [arm]vsubq m[u32](uint32x4 t inactive,	inactive -> Od	VMSR P0,Rp	Od -> result	MVE
uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VSUBT.I32 Qd,Qn,Qm		
	p -> Rp			
uint8x16_t [_arm_]vsubq_m[_n_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VSUBT.I8 Qd,Qn,Rm		
	p -> Rp	V3CB1.18 Qu,Qii,Kiii		
uint16x8 t [arm]vsubq 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		
	b -> Rm	VSUBT.I16 Qd,Qn,Rm		
	p -> Rp			
uint32x4_t [arm_]vsubq_m[_n_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VSUBT.I32 Qd,Qn,Rm		
	p -> Rp	VSCB1:132 Qu,Qii,Kiii		
float16x8_t [arm_]vsubq_m[_f16](float16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VSUBT.F16 Qd,Qn,Qm		
float32x4_t [_arm_]vsubq_m[_f32](float32x4_t inactive,	p -> Rp inactive -> Qd	VMCD DO D.	Od -> result	MVE
float32x4_t iarm_jvsubq_m[_132](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
noat32x4_t a, noat32x4_t b, nive_pica10_t p)	b -> Qm	VSUBT.F32 Qd,Qn,Qm		
	p -> Rp	27272		
float16x8_t [arm_]vsubq_m[_n_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm p -> Rp	VSUBT.F16 Qd,Qn,Rm		
float32x4 t [arm]vsubq m[n f32](float32x4 t	inactive -> Od	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn	VPST	Qu => result	IVIVE
	b -> Rm	VSUBT.F32 Qd,Qn,Rm		
	p -> Rp			
int8x16_t [arm_]vsubq_x[_s8](int8x16_t a, int8x16_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Qm	VPST		
intleve to amorphory of alelintleve to intleve t	p -> Rp	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	VMSR P0,Rp VPST	Qd -> result	MVE
o, mve_predro_t p)	p -> Rp	VSUBT.I16 Qd,Qn,Qm		
int32x4_t [arm_]vsubq_x[_s32](int32x4_t a, int32x4_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VSUBT.I32 Qd,Qn,Qm		
int8x16_t [_arm_]vsubq_x[_n_s8](int8x16_t a, int8_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Rm p -> Rp	VPST VSUBT.I8 Qd,Qn,Rm		
int16x8 t [arm]vsubq x[n s16](int16x8 t a, int16 t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VSUBT.I16 Qd,Qn,Rm		
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		
nint0v16 +f orm bronks of n01/nint0-16 +-	p -> Rp	VSUBT.I32 Qd,Qn,Rm	04 > #14	MVE
uint8x16_t [arm_]vsubq_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
amoxio_t o, mvc_picaro_t p)	p -> Rp	VSUBT.I8 Qd,Qn,Qm		
uint16x8_t [arm_]vsubq_x[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
	b -> Qm	VPST	-	1
uint16x8_t b, mve_pred16_t p)	b->QIII	VI 51		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [arm_]vsubq_x[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
uint8x16 t [arm]vsubq x[n u8](uint8x16 t a, uint8 t	p -> Rp a -> On	VSUBT.I32 Qd,Qn,Qm VMSR P0.Rp	Od -> result	MVE
b, mve_pred16_t p)	a -> Qn b -> Rm	VMSK PO,KP VPST	Qu -> resuit	MVE
	p -> Rp	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 p -> Rp	VPST VSUBT.I16 Qd,Qn,Rm		
uint32x4 t [arm]vsubq x[n u32](uint32x4 t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, mve_pred16_t p)	b -> Rm	VPST		
flord (c.0 + f]	p -> Rp	VSUBT.I32 Qd,Qn,Rm	0.1 >16	MVE
float16x8_t [arm_]vsubq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
nourono_t o, mve_prouro_t p/	p -> Rp	VSUBT.F16 Qd,Qn,Qm		
float32x4_t [arm_]vsubq_x[_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST VSUBT.F32 Qd,Qn,Qm		
float16x8_t [arm_]vsubq_x[_n_f16](float16x8_t a,	p -> Rp a -> Qn	VMSR P0,Rp	Od -> result	MVE
float16_t b, mve_pred16_t p)	b -> Rm	VPST	L	/
	p -> Rp	VSUBT.F16 Qd,Qn,Rm		
float32x4_t [arm_]vsubq_x[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Qd -> result	MVE
moat32_t b, mve_pred10_t p)	p -> Rn	VSUBT.F32 Qd,Qn,Rm		
float16x8_t [_arm_]vcaddq_rot90[_f16](float16x8_t a,	a -> Qn	VCADD.F16 Qd,Qn,Qm,#90	Qd -> result	MVE/NEON
float16x8_t b) float32x4_t [arm_]vcaddq_rot90[_f32](float32x4_t a,	b -> Qm a -> Qn	VCADD.F32 Qd,Qn,Qm,#90	Qd -> result	MVE/NEON
float32x4_t b) int8x16 t [arm]vcaddq rot90[s8](int8x16 t a,	b -> Qm a -> Qn	VCADD.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
int8x16_t b)	b -> Qm			
int16x8_t [arm_]vcaddq_rot90[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VCADD.I16 Qd,Qn,Qm,#90	Qd -> result	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, uint8x16_t b)	a -> Qn b -> Qm	VCADD.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
uint16x8_t [_arm_]vcaddq_rot90[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VCADD.I16 Qd,Qn,Qm,#90	Qd -> result	MVE
uint32x4_t [arm_]vcaddq_rot90[_u32](uint32x4_t a,	a -> Qn	VCADD.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
uint32x4_t b) float16x8_t [_arm_]vcaddq_rot270[_f16](float16x8_t a,	b -> Qm a -> Qn	VCADD.F16 Qd,Qn,Qm,#270	Qd -> result	MVE/NEON
float16x8_t b) float32x4_t [_arm_]vcaddq_rot270[_f32](float32x4_t a,	b -> Qm 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	Qd -> result	MVE
int8x16_t b) int16x8_t [arm_]vcaddq_rot270[_s16](int16x8_t a,	b -> Qm a -> On	VCADD.I16 Qd,Qn,Qm,#270	Qd -> result	MVE
int16x8_t b) int32x4_t [arm_]vcaddq_rot270[_s32](int32x4_t a,	b -> Qm a -> Qn	VCADD.I32 Qd,Qn,Qm,#270	Qd -> result	MVE
int32x4_t b)	b -> Qm			
uint8x16_t [arm_]vcaddq_rot270[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VCADD.I8 Qd,Qn,Qm,#270	Qd -> result	MVE
uint16x8_t [arm_]vcaddq_rot270[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VCADD.I16 Qd,Qn,Qm,#270	Qd -> result	MVE
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	Qd -> result	MVE
inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VCADDT.F16 Qd,Qn,Qm,#90		
float32x4_t [arm_]vcaddq_rot90_m[_f32](float32x4_t	inactive -> Od	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn	VPST		·
	b -> Qm	VCADDT.F32 Qd,Qn,Qm,#90		
int8x16_t [arm_]vcaddq_rot90_m[_s8](int8x16_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn	VPST VPST	Qu > resuit	,
	b -> Qm	VCADDT.I8 Qd,Qn,Qm,#90		
intlend the own broader more of alcoholical	p -> Rp	VMCD DO Do	04 >14	MVE
int16x8_t [arm_]vcaddq_rot90_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
,	b -> Qm	VCADDT.I16 Qd,Qn,Qm,#90		
	p -> Rp	7 . 7 . 7 .		
int32x4_t [_arm_]vcaddq_rot90_m[_s32](int32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VCADDT.I32 Qd,Qn,Qm,#90		
	p -> Rp	. 5/1551.152 Qu,Qii,Qiii,#90		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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	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)	p -> Rp inactive -> Qd a -> Qn b -> Qm	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)	p -> Rp inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
float16x8_t [_arm_]vcaddq_rot270_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm	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)	p -> Rp inactive -> Qd a -> Qn b -> Qm	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)	p -> Rp inactive -> Qd a -> Qn b -> Qm	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)	p -> Rp 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	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)	p -> Rp 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	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) $	p -> Rp a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.F16 Qd,Qn,Qm,#90	Qd -> result	MVE
float32x4_t [_arm_]vcaddq_rot90_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.F32 Qd,Qn,Qm,#90	Qd -> result	MVE
int8x16_t [_arm_]vcaddq_rot90_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
int16x8_t [_arm_]vcaddq_rot90_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p) int32x4 t [_arm_]vcaddq_rot90_x[_s32](int32x4_t a,	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I16 Qd,Qn,Qm,#90	Qd -> result Qd -> result	MVE
int32x4_t = arm_ vcaddq_rot90_x[_s52](int32x4_t a, int32x4_t b, mve_pred16_t p) uint8x16_t [_arm_ vcaddq_rot90_x[_u8](uint8x16_t a,	a -> Qn b -> Qm p -> Rp a -> Qn	VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#90 VMSR P0.Rp	Qd -> result	MVE MVE
uint8x16_t b, mve_pred16_t p) uint16x8_t [arm_]vcaddq_rot90_x[_u16](uint16x8_t a,	b -> Qm p -> Rp a -> Qn	VPST VCADDT.I8 Qd,Qn,Qm,#90 VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p) uint32x4_t [arm_]vcaddq_rot90_x[_u32](uint32x4_t a,	b -> Qm p -> Rp a -> Qn	VPST VCADDT.I16 Qd,Qn,Qm,#90 VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p) float16x8_t [arm_]vcaddq_rot270_x[_f16](float16x8_t	b -> Qm p -> Rp a -> Qn	VPST VCADDT.I32 Qd,Qn,Qm,#90 VMSR P0,Rp	Qd -> result	MVE
a, float16x8_t b, mve_pred16_t p) float32x4_t [arm_]vcaddq_rot270_x[_f32](float32x4_t	b -> Qm p -> Rp a -> Qn	VPST VCADDT.F16 Qd,Qn,Qm,#270 VMSR P0,Rp	Qd -> result	MVE
a, float32x4_t b, mve_pred16_t p) int8x16_t [_arm_]vcaddq_rot270_x[_s8](int8x16_t a,	b -> Qm p -> Rp a -> Qn	VPST VCADDT.F32 Qd,Qn,Qm,#270 VMSR P0,Rp	Qd -> result	MVE
int8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCADDT.I8 Qd,Qn,Qm,#270		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vcaddq_rot270_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t [arm_]vcaddq_rot270_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VCADDT.I16 Qd,Qn,Qm,#270 VMSR P0,Rp VPST	Qd -> result	MVE
uint8x16_t [_arm_]vcaddq_rot270_x[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm	VCADDT.I32 Qd,Qn,Qm,#270 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 -> Om	VCADDT.18 Qd,Qn,Qm,#270 VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t [_arm_]vcaddq_rot270_x[_u32](uint32x4_t a,	p -> Rp a -> Qn	VCADDT.I16 Qd,Qn,Qm,#270 VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p) float16x8_t [arm_]vcmlaq[_f16](float16x8_t a,	b -> Qm p -> Rp a -> Qda	VPST VCADDT.I32 Qd,Qn,Qm,#270 VCMLA.F16 Qda,Qn,Qm,#0	Qda -> result	MVE/NEON
float16x8_t b, float16x8_t c) float32x4_t [arm_]vcmlaq[_f32](float32x4_t a,	b -> Qn c -> Qm a -> Qda	VCMLA.F32 Qda,Qn,Qm,#0	Qda -> result	MVE/NEON
float32x4_t b, float32x4_t c) float16x8_t [arm_]vcmlaq_rot90[_f16](float16x8_t a,	b -> Qn c -> Qm a -> Qda	VCMLA.F16 Qda,Qn,Qm,#90	Qda -> result	MVE/NEON
float16x8_t b, float16x8_t c)	b -> Qn c -> Qm			
float32x4_t [arm_]vcmlaq_rot90[_f32](float32x4_t a, 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 p -> Rp	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)	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
float16x8_t [arm_]vcmlaq_rot270_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,#270	Qda -> result	MVE
float32x4_t [arm_]vcmlaq_rot270_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,#270	Qda -> result	MVE
float16x8_t [_arm_]vcmulq[_f16](float16x8_t a, float16x8_t b)	p -> Rp a -> Qn b -> Qm	VCMUL.F16 Qd,Qn,Qm,#0	Qd -> result	MVE
float32x4_t [arm_]vcmulq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMUL.F32 Qd,Qn,Qm,#0	Qd -> result	MVE
float16x8_t [arm_]vcmulq_rot90[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMUL.F16 Qd,Qn,Qm,#90	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [_arm_]vcmulq_rot90[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMUL.F32 Qd,Qn,Qm,#90	Qd -> result	MVE
float16x8_t [arm_]vcmulq_rot180[_f16](float16x8_t a,	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	Qd -> result	MVE
float32x4_t b) float16x8_t [arm_]vcmulq_rot270[_f16](float16x8_t a,	b -> Qm a -> Qn	VCMUL.F16 Qd,Qn,Qm,#270	Qd -> result	MVE
float16x8_t b) float32x4_t [arm_]vcmulq_rot270[_f32](float32x4_t a,	b -> Qm 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	Od -> result	MVE
inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VPST VCMULT.F16 Qd,Qn,Qm,#0	Qu' > Tesun	III VE
float32x4_t [_arm_]vcmulq_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 VCMULT.F32 Qd,Qn,Qm,#0	Qd -> result	MVE
float16x8_t [_arm_]vcmulq_rot90_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 VCMULT.F16 Qd,Qn,Qm,#90	Qd -> result	MVE
float32x4_t [_arm_]vcmulq_rot90_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 VCMULT.F32 Qd,Qn,Qm,#90	Qd -> result	MVE
float16x8_t [_arm_]vcmulq_rot180_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F16 Qd,Qn,Qm,#180	Qd -> result	MVE
float32x4_t [_arm_]vcmulq_rot180_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 VCMULT.F32 Qd,Qn,Qm,#180	Qd -> result	MVE
float16x8_t [_arm_]vcmulq_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 VCMULT.F16 Qd,Qn,Qm,#270	Qd -> result	MVE
float32x4_t [_arm_]vcmulq_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 VCMULT.F32 Qd,Qn,Qm,#270	Qd -> result	MVE
float16x8_t [_arm_]vcmulq_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F16 Qd,Qn,Qm,#0	Qd -> result	MVE
float32x4_t [_arm_]vcmulq_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F32 Qd,Qn,Qm,#0	Qd -> result	MVE
float16x8_t [_arm_]vcmulq_rot90_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F16 Qd,Qn,Qm,#90	Qd -> result	MVE
float32x4_t [_arm_]vcmulq_rot90_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F32 Qd,Qn,Qm,#90	Qd -> result	MVE
float16x8_t [arm_]vcmulq_rot180_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F16 Qd,Qn,Qm,#180	Qd -> result	MVE
float32x4_t [_arm_]vcmulq_rot180_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F32 Qd,Qn,Qm,#180	Qd -> result	MVE
float16x8_t [arm_]vcmulq_rot270_x[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F16 Qd,Qn,Qm,#270	Qd -> result	MVE
float32x4_t [_arm_]vcmulq_rot270_x[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F32 Qd,Qn,Qm,#270	Qd -> result	MVE
int8x16_t [arm_]vqabsq[_s8](int8x16_t a) int16x8_t [arm_]vqabsq[_s16](int16x8_t a)	a -> Qm a -> Qm	VQABS.S8 Qd,Qm VQABS.S16 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
int32x4_t [arm_]vqabsq[_s32](int32x4_t a)	a -> Qm	VQABS.S32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vqabsq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t [_arm_]vqabsq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm p -> Rp	VQABST.S8 Qd,Qm VMSR P0,Rp VPST VQABST.S16 Qd,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [arm_]vqabsq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
int8x16_t [arm_]vqaddq[_n_s8](int8x16_t a, int8_t b)	p -> Rp a -> Qn	VQABST.S32 Qd,Qm VQADD.S8 Qd,Qn,Rm	Qd -> result	MVE
::416-0 4 []	b -> Rm	VOADD 616 O4 On Day	0.1	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 b -> Rm	VQADD.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vqaddq[_n_u8](uint8x16_t a, uint8_t	a -> Qn	VQADD.U8 Qd,Qn,Rm	Qd -> result	MVE
b) uint16x8_t [_arm_]vqaddq[_n_u16](uint16x8_t a,	b -> Rm 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 -> Qn	VQADD.U32 Qd,Qn,Rm	Qd -> result	MVE
uint32_t b) int8x16_t [arm_]vqaddq[_s8](int8x16_t a, int8x16_t b)	b -> Rm a -> Qn	VQADD.S8 Qd,Qn,Qm	Od -> result	MVE/NEON
	b->Qm	2 7 7 7 7		
int16x8_t [arm_]vqaddq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQADD.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
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 -> Qn	VQADD.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
b) uint16x8_t [_arm_]vqaddq[_u16](uint16x8_t a,	b -> Qm a -> Qn	VQADD.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t b) uint32x4_t [arm_]vqaddq[_u32](uint32x4_t a,	b -> Qm 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	Qd -> result	MVE
int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn	VPST	Qu > resurt	I WAY E
	b -> Rm p -> Rp	VQADDT.S8 Qd,Qn,Rm		
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	2		
int32x4_t [arm_]vqaddq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
miszar_t a, misz_t o, mvc_predro_t p)	b -> Rm	VQADDT.S32 Qd,Qn,Rm		
uint8x16_t [arm_]vqaddq_m[_n_u8](uint8x16_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn	VPST	Qu > resurt	141 4 12
	b -> Rm p -> Rp	VQADDT.U8 Qd,Qn,Rm		
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 b -> Rm	VPST VQADDT.U16 Qd,Qn,Rm		
	p -> Rp	VQADD1.010 Qu,Qii,Kiii		
uint32x4_t [_arm_]vqaddq_m[_n_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VQADDT.U32 Qd,Qn,Rm		
	p -> Rp	VALCE DO D	01	NOTE
int8x16_t [_arm_]vqaddq_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
	b -> Qm	VQADDT.S8 Qd,Qn,Qm		
int16x8_t [arm_]vqaddq_m[_s16](int16x8_t inactive,	p -> Rp inactive -> Od	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST	Qu -> resurt	WIVE
	b -> Qm p -> Rp	VQADDT.S16 Qd,Qn,Qm		
int32x4_t [arm_]vqaddq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VQADDT.S32 Qd,Qn,Qm		<u> </u>
uint8x16_t [_arm_]vqaddq_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 VQADDT.U8 Qd,Qn,Qm		
	p -> Rp	2 . 2 . 2		
uint16x8_t [arm_]vqaddq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
amerono_t a, amerono_t o, mvc_prearo_t p)	b -> Qm	VQADDT.U16 Qd,Qn,Qm		
uint32x4_t [arm_]vqaddq_m[_u32](uint32x4_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
	a -> Qn	VMSK PU,KP VPST	Qu -> resuit	IVI V E
uint32x4_t a, uint32x4_t b, mve_pred16_t p)	/ 2	VQADDT.U32 Qd,Qn,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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 -> Qm	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 -> Om	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 -> Om	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
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 -> Qm	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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [_arm_]vqrdmladhxq_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 VQRDMLADHXT.S16		
	p -> Rp	Qd,Qn,Qm		
int32x4_t [_arm_]vqrdmladhxq_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	VQRDMLADHXT.S32		
0.16.1	p -> Rp	Qd,Qn,Qm		
int8x16_t [_arm_]vqdmlahq[_n_s8](int8x16_t add, int8x16_t m1, int8_t m2)	add -> Qda m1 -> On	VQDMLAH.S8 Qda,Qn,Rm	Qda -> result	MVE
	m2 -> Rm			
int16x8_t [arm_]vqdmlahq[_n_s16](int16x8_t add,	add -> Qda	VQDMLAH.S16 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t m1, int16_t m2)	m1 -> Qn m2 -> Rm			
int32x4_t [arm_]vqdmlahq[_n_s32](int32x4_t add,	add -> Qda	VQDMLAH.S32 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t m1, int32_t m2)	m1 -> Qn			
int8x16_t [arm_]vqdmlahq_m[_n_s8](int8x16_t add,	m2 -> Rm add -> Oda	VMSR P0,Rp	Qda -> result	MVE
int8x16_t m1, int8_t m2, mve_pred16_t p)	m1 -> Qn	VPST	Qua -> resurt	WVL
	m2 -> Rm	VQDMLAHT.S8 Qda,Qn,Rm		
int16x8_t [arm_]vqdmlahq_m[_n_s16](int16x8_t add,	p -> Rp add -> Qda	VMSR P0,Rp	Qda -> result	MVE
int16x8_t m1, int16_t m2, mve_pred16_t p)	m1 -> Qn	VPST	Qua -> resurt	WVL
	m2 -> Rm	VQDMLAHT.S16 Qda,Qn,Rm		
int32x4_t [arm_]vqdmlahq_m[_n_s32](int32x4_t add,	p -> Rp add -> Qda	VMSR P0,Rp	Oda -> result	MVE
int32x4_t m1, int32_t m2, mve_pred16_t p)	m1 -> Qua	VPST	Qua -> resuit	WIVE
_ · · _ · · _ · · _ · · · · ·	m2 -> Rm	VQDMLAHT.S32 Qda,Qn,Rm		
int8x16_t [arm_]vqrdmlahq[_n_s8](int8x16_t add,	p -> Rp add -> Qda	VORDMLAH.S8 Qda,Qn,Rm	Oda -> result	MVE
int8x16_t m1, int8_t m2)	m1 -> Qua	VQKDIVILAH.38 Qua,Qii,Kiii	Qua -> resuit	IVI V E
	m2 -> Rm			
int16x8_t [arm_]vqrdmlahq[_n_s16](int16x8_t add,	add -> Qda	VQRDMLAH.S16 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t m1, int16_t m2)	m1 -> Qn m2 -> Rm			
int32x4_t [arm_]vqrdmlahq[_n_s32](int32x4_t add,	add -> Qda	VQRDMLAH.S32 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t m1, int32_t m2)	m1 -> Qn			
int8x16_t [arm_]vqrdmlahq_m[_n_s8](int8x16_t add,	m2 -> Rm add -> Qda	VMSR P0,Rp	Qda -> result	MVE
int8x16_t m1, int8_t m2, mve_pred16_t p)	m1 -> Qn	VPST	Qua > resurt	III V E
	m2 -> Rm	VQRDMLAHT.S8 Qda,Qn,Rm		
int16x8_t [arm_]vqrdmlahq_m[_n_s16](int16x8_t add,	p -> Rp add -> Qda	VMSR P0,Rp	Qda -> result	MVE
int16x8_t m1, int16_t m2, mve_pred16_t p)	m1 -> Qn	VPST	Qua > resurt	III V E
	m2 -> Rm	VQRDMLAHT.S16 Qda,Qn,Rm		
int32x4_t [arm_]vqrdmlahq_m[_n_s32](int32x4_t add,	p -> Rp add -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32x4_t m1, int32_t m2, mve_pred16_t p)	m1 -> Qn	VPST	Qua y resun	
	m2 -> Rm	VQRDMLAHT.S32 Qda,Qn,Rm		
int8x16_t [arm_]vqdmlashq[_n_s8](int8x16_t m1,	p -> Rp m1 -> Qda	VODMLASH.S8 Oda,On,Rm	Oda -> result	MVE
int8x16_t m2, int8_t add)	m2 -> Qn		Q	1
. 16 0 16 1 1 1 1 1 1 1 16 16 0 1 1	add -> Rm	VODM + CH CL CO L O D	0.1	NOTE
int16x8_t [arm_]vqdmlashq[_n_s16](int16x8_t m1, int16x8_t m2, int16_t add)	m1 -> Qda m2 -> Qn	VQDMLASH.S16 Qda,Qn,Rm	Qda -> result	MVE
merono_e m2, mero_e add/	add -> Rm			
int32x4_t [arm_]vqdmlashq[_n_s32](int32x4_t m1,	m1 -> Qda	VQDMLASH.S32 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t m2, int32_t add)	m2 -> Qn add -> Rm			
int8x16_t [arm_]vqdmlashq_m[_n_s8](int8x16_t m1,	m1 -> Qda	VMSR P0,Rp	Qda -> result	MVE
int8x16_t m2, int8_t add, mve_pred16_t p)	m2 -> Qn	VPST		
	add -> Rm p -> Rp	VQDMLASHT.S8 Qda,Qn,Rm		
int16x8_t [arm_]vqdmlashq_m[_n_s16](int16x8_t m1,	m1 -> Qda	VMSR P0,Rp	Qda -> result	MVE
int16x8_t m2, int16_t add, mve_pred16_t p)	m2 -> Qn	VPST		
	add -> Rm p -> Rp	VQDMLASHT.S16 Qda,Qn,Rm		
int32x4_t [arm_]vqdmlashq_m[_n_s32](int32x4_t m1,	m1 -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32x4_t m2, int32_t add, mve_pred16_t p)	m2 -> Qn	VPST		
	add -> Rm p -> Rp	VQDMLASHT.S32 Qda,Qn,Rm		
int8x16_t [arm_]vqrdmlashq[_n_s8](int8x16_t m1,	m1 -> Qda	VQRDMLASH.S8 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t m2, int8_t add)	m2 -> Qn			
intleve to any broadwheat1676-416-9 . 1	add -> Rm	VODDMI ACII CI CO D. D.	Ode >1	MVE
int16x8_t [arm_]vqrdmlashq[_n_s16](int16x8_t m1, int16x8_t m2, int16_t add)	m1 -> Qda m2 -> Qn	VQRDMLASH.S16 Qda,Qn,Rm	Qda -> result	MVE
	add -> Rm			

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [_arm_]vqrdmlashq[_n_s32](int32x4_t m1, int32x4_t m2, int32_t add)	m1 -> Qda m2 -> Qn	VQRDMLASH.S32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [_arm_]vqrdmlashq_m[_n_s8](int8x16_t m1, int8x16_t m2, int8_t add, mve_pred16_t p)	add -> Rm m1 -> Qda m2 -> On	VMSR P0,Rp VPST	Qda -> result	MVE
	add -> Rm p -> Rp	VQRDMLASHT.S8 Qda,Qn,Rm		
int16x8_t [_arm_]vqrdmlashq_m[_n_s16](int16x8_t m1, int16x8_t m2, int16_t add, mve_pred16_t p)	m1 -> Qda m2 -> Qn add -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLASHT.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vqrdmlashq_m[_n_s32](int32x4_t m1, int32x4_t m2, int32_t add, mve_pred16_t p)	m1 -> Qda m2 -> Qn add -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLASHT.S32 Qda,Qn,Rm	Qda -> result	MVE
$int8x16_t \ [_arm_]vqdmlsdhq[_s8](int8x16_t \ inactive, \\ int8x16_t \ a, int8x16_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLSDH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqdmlsdhq[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b)	inactive -> Qd a -> Qn b -> Qm	VQDMLSDH.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4_t \ [_arm_]vqdmlsdhq[_s32](int32x4_t \ inactive, \\ int32x4_t \ a, int32x4_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLSDH.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vqdmlsdhq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	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)	p -> Rp inactive -> Qd a -> Qn b -> Qm	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)	p -> Rp 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
$\label{lem:continuous} int8x16_t \ [_arm_] vqrdmlsdhq[_s8] (int8x16_t \ inactive, \\ int8x16_t \ a, int8x16_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDH.S8 Qd,Qn,Qm	Qd -> result	MVE
$ \begin{array}{lll} int16x8_t & [_arm_]vqrdmlsdhq[_s16](int16x8_t & inactive, \\ int16x8_t & a, & int16x8_t & b) \end{array} $	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDH.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4_t \ [_arm_]vqrdmlsdhq[_s32](int32x4_t \ inactive, \\ int32x4_t \ a, int32x4_t \ b)$	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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [_arm_]vqrdmlsdhq_m[_s32](int32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VPST VQRDMLSDHT.S32 Qd,Qn,Qm		
$int8x16_t \ [_arm_]vqrdmlsdhxq[_s8](int8x16_t \ inactive, \\ int8x16_t \ a, int8x16_t \ b)$	inactive -> Qd a -> Qn b -> Om	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	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)	p -> Rp inactive -> Qd a -> Qn b -> Rm	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)	p -> Rp inactive -> Qd a -> Qn b -> Rm	VMSR P0,Rp VPST VQDMULHT.S32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vqdmulhq[_s8](int8x16_t a, int8x16_t b)	p -> Rp 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 p -> Rp	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)	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
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [arm_]vqrdmulhq_m[_n_s32](int32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VPST VQRDMULHT.S32 Qd,Qn,Rm		
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
$int64x2_t \ [_arm_]vqdmulltq_m[_s32] (int64x2_t \ inactive, \\ int32x4_t \ a, \ int32x4_t \ b, \ mve_pred16_t \ p)$	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMULLTT.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqnegq[_s8](int8x16_t a)	a -> Qm	VQNEG.S8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vqnegq[_s16](int16x8_t a)	a -> Qm	VQNEG.S16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vqnegq[_s32](int32x4_t a) int8x16_t [_arm_]vqnegq_m[_s8](int8x16_t inactive,	a -> Qm inactive -> Od	VQNEG.S32 Qd,Qm VMSR P0,Rp	Qd -> result Qd -> result	MVE/NEON MVE
int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR PO,RP VPST VQNEGT.S8 Qd,Qm	Qu -> resun	IVIVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
int32x4_t [_arm_]vqnegq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm p -> Rp	VQNEGT.S16 Qd,Qm VMSR P0,Rp VPST VQNEGT.S32 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vqsubq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VQSUB.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vqsubq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQSUB.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [arm_]vqsubq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VQSUB.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vqsubq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VQSUB.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [arm_]vqsubq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VQSUB.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [arm_]vqsubq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VQSUB.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [_arm_]vqsubq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQSUBT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vqsubq_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 VQSUBT.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vqsubq_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 VQSUBT.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vqsubq_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 VQSUBT.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [arm_]vqsubq_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 VQSUBT.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [arm_]vqsubq_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 VQSUBT.U32 Qd,Qn,Rm	Qd -> result	MVE
$int8x16_t \ [_arm_]vqsubq[_s8](int8x16_t \ a, int8x16_t \ b)$	a -> Qn b -> Qm	VQSUB.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vqsubq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQSUB.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vqsubq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQSUB.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vqsubq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VQSUB.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vqsubq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VQSUB.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vqsubq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VQSUB.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vqsubq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQSUBT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqsubq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VQSUBT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vqsubq_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 VQSUBT.S32 Qd,Qn,Qm	Qd -> result	MVE
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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]	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)	base -> Rn	VLDRB.S32 Qd,[Rn]	Qd -> result	MVE
uint8x16_t [arm_]vldrbq_u8(uint8_t const * base)	base -> Rn	VLDRB.8 Qd,[Rn]	Qd -> result	MVE
uint16x8_t [_arm_]vldrbq_u16(uint8_t const * base)	base -> Rn	VLDRB.U16 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [_arm_]vldrbq_u32(uint8_t const * base)	base -> Rn	VLDRB.U32 Qd,[Rn]	Qd -> result	MVE
int8x16_t [arm_]vldrbq_z_s8(int8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.8 Qd,[Rn]	Qd -> result	MVE
int16x8_t [_arm_]vldrbq_z_s16(int8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.S16 Qd,[Rn]	Qd -> result	MVE
int32x4_t [_arm_]vldrbq_z_s32(int8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.S32 Qd,[Rn]	Qd -> result	MVE
uint8x16_t [_arm_]vldrbq_z_u8(uint8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.8 Qd,[Rn]	Qd -> result	MVE
uint16x8_t [_arm_]vldrbq_z_u16(uint8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.U16 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [_arm_]vldrbq_z_u32(uint8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.U32 Qd,[Rn]	Qd -> result	MVE
int16x8_t [arm_]vldrhq_s16(int16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE
int32x4_t [arm_]vldrhq_s32(int16_t const * base)	base -> Rn	VLDRH.S32 Qd,[Rn]	Qd -> result	MVE
uint16x8_t [arm_]vldrhq_u16(uint16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [arm_]vldrhq_u32(uint16_t const * base)	base -> Rn	VLDRH.U32 Qd,[Rn]	Qd -> result	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, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRHT.S16 Qd,[Rn]	Qd -> result	MVE
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_f16(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
	base -> Rn base -> Rn		Qd -> result	MVE
uint32x4_t [_arm_]vldrwq_u32(uint32_t const * base) float32x4_t [_arm_]vldrwq_f32(float32_t const * base)	base -> Rn base -> Rn	VLDRW.32 Qd,[Rn] VLDRW.32 Qd,[Rn]		MVE MVE
int32x4_t [_arm_]vldrwq_132(float32_t const * base)	base -> Rn	VMSR P0,Rp	Qd -> result Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [arm_]vldrwq_z_u32(uint32_t const * base,	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	$p \rightarrow Rp$	VPST		
		VLDRWT.32 Qd,[Rn]		
float32x4_t [arm_]vldrwq_z_f32(float32_t const * base,	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VLDRWT.32 Qd,[Rn]		
int8x16_t [arm_]vld1q[_s8](int8_t const * base)	base -> Rn	VLDRW 1.52 Qd,[Rfi] 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 MVE/NEON
int32x4_t [arm]vld1q[_s32](int32_t const * base)	base -> Rn	VLDRW.32 Qd,[Rn]	Qd -> result	MVE/NEON
uint8x16_t [arm_]vld1q[_u8](uint8_t const * base)	base -> Rn	VLDRB.8 Qd,[Rn]	Qd -> result	MVE/NEON
uint16x8_t [arm_]vld1q[_u16](uint16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Od -> 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		
' . (C. O f	1 . D	VLDRBT.8 Qd,[Rn]	0.1	MATE
int16x8_t [arm_]vld1q_z[_s16](int16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
inve_pred1o_t p)	p -> Kp	VLDRHT.16 Qd,[Rn]		
int32x4_t [arm_]vld1q_z[_s32](int32_t const * base,	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
mve pred16 t p)	p -> Rp	VPST	Qu' > Tesun	11112
		VLDRWT.32 Qd,[Rn]		
uint8x16_t [arm_]vld1q_z[_u8](uint8_t const * base,	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	$p \rightarrow Rp$	VPST		
		VLDRBT.8 Qd,[Rn]		
uint16x8_t [arm_]vld1q_z[_u16](uint16_t const * base,	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
	1 D	VLDRHT.16 Qd,[Rn]	0.1 >16	MVE
uint32x4_t [_arm_]vld1q_z[_u32](uint32_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
inve_pred1o_t p)	p -> Kp	VLDRWT.32 Qd,[Rn]		
float16x8_t [_arm_]vld1q_z[_f16](float16_t const * base,	base -> Rn	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu' > Tesun	111.12
		VLDRHT.16 Qd,[Rn]		
float32x4_t [arm_]vld1q_z[_f32](float32_t const * base,	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	$p \rightarrow Rp$	VPST		
		VLDRWT.32 Qd,[Rn]		
int16x8_t [arm_]vldrhq_gather_offset[_s16](int16_t	base -> Rn	VLDRH.U16 Qd,[Rn,Qm]	Qd -> result	MVE
const * base, uint16x8_t offset)	offset -> Qm	VI DDII 622 O4 [D., O.,]	0.1 >16	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)	offset -> Qm	VEDKII.010 Qu,[Kii,Qiii]	Qu -> result	WYL
uint32x4_t [arm_]vldrhq_gather_offset[_u32](uint16_t	base -> Rn	VLDRH.U32 Qd,[Rn,Qm]	Qd -> result	MVE
const * base, uint32x4_t offset)	offset -> Qm			
float16x8_t [arm_]vldrhq_gather_offset[_f16](float16_t	base -> Rn	VLDRH.F16 Qd,[Rn,Qm]	Qd -> result	MVE
const * base, uint16x8_t offset)	offset -> Qm			
$int16x8_t \ [_arm_]vldrhq_gather_offset_z [_s16] (int16_t$	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
const * base, uint16x8_t offset, mve_pred16_t p)	offset -> Qm	VPST		
20 4 4 5 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	p -> Rp	VLDRHT.U16 Qd,[Rn,Qm]	0.1) are
int32x4_t [_arm_]vldrhq_gather_offset_z[_s32](int16_t	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
const * base, uint32x4_t offset, mve_pred16_t p)	offset -> Qm	VPST VLDRHT.S32 Qd,[Rn,Qm]		
uint16x8 t	p -> Rp base -> Rn	VMSR P0,Rp	Qd -> result	MVE
arm vldrhq gather offset z[u16](uint16 t const *	offset -> Qm	VPST	Qu => icsuit	MIVE
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 *	offset -> Qm	VPST		
base, uint32x4_t offset, mve_pred16_t p)	p -> Rp	VLDRHT.U32 Qd,[Rn,Qm]		
float16x8_t	base -> Rn	VMSR P0,Rp	Qd -> result	MVE
I ama luldula action office at f161/f1cct16 t court *	offset -> Qm	VPST		
	p -> Rp	VLDRHT.F16 Qd,[Rn,Qm]	0.1) are
base, uint16x8_t offset, mve_pred16_t p)		VLDRH.U16 Qd,[Rn,Qm,UXTW	Qd -> result	MVE
base, uint16x8_t offset, mve_pred16_t p) int16x8_t	base -> Rn	#11		1
[_arm_]vldrhq_gather_offset_z[_f16](float16_t const * base, uint16x8_t offset, mve_pred16_t p) int16x8_t [_arm_]vldrhq_gather_shifted_offset[_s16](int16_t const * base_nint16x8_t offset)	base -> Rn offset -> Qm	#1]		
base, uint16x8_t offset, mve_pred16_t p) int16x8_t [_arm_]vldrhq_gather_shifted_offset[_s16](int16_t const * base, uint16x8_t offset)	offset -> Qm		Od -> recult	MVF
base, uint16x8_t offset, mve_pred16_t p) int16x8_t [_arm_]vldrhq_gather_shifted_offset[_s16](int16_t const * base, uint16x8_t offset) int32x4_t	offset -> Qm base -> Rn	VLDRH.S32 Qd,[Rn,Qm,UXTW	Qd -> result	MVE
base, uint16x8_t offset, mve_pred16_t p) int16x8_t [_arm_]vldrhq_gather_shifted_offset[_s16](int16_t const * base, uint16x8_t offset) int32x4_t [_arm_]vldrhq_gather_shifted_offset[_s32](int16_t const	offset -> Qm		Qd -> result	MVE
base, uint16x8_t offset, mve_pred16_t p) int16x8_t [arm_]vldrhq_gather_shifted_offset[_s16](int16_t const * base, uint16x8_t offset) int32x4_t [arm_]vldrhq_gather_shifted_offset[_s32](int16_t const * base, uint32x4_t offset)	offset -> Qm base -> Rn offset -> Qm	VLDRH.S32 Qd,[Rn,Qm,UXTW #1]		
base, uint16x8_t offset, mve_pred16_t p) int16x8_t [_arm_]vldrhq_gather_shifted_offset[_s16](int16_t const * base, uint16x8_t offset) int32x4_t [_arm_]vldrhq_gather_shifted_offset[_s32](int16_t const	offset -> Qm base -> Rn	VLDRH.S32 Qd,[Rn,Qm,UXTW	Qd -> result Qd -> result	MVE MVE
base, uint16x8_t offset, mve_pred16_t p) int16x8_t [_arm_]vldrhq_gather_shifted_offset[_s16](int16_t const * base, uint16x8_t offset) int32x4_t [_arm_]vldrhq_gather_shifted_offset[_s32](int16_t const * base, uint32x4_t offset) uint16x8_t	offset -> Qm base -> Rn offset -> Qm base -> Rn	VLDRH.S32 Qd,[Rn,Qm,UXTW #1] VLDRH.U16 Qd,[Rn,Qm,UXTW		
base, uint16x8_t offset, mve_pred16_t p) int16x8_t [_arm_]vldrhq_gather_shifted_offset[_s16](int16_t const * base, uint16x8_t offset) int32x4_t [_arm_]vldrhq_gather_shifted_offset[_s32](int16_t const * base, uint32x4_t offset) uint16x8_t [_arm_]vldrhq_gather_shifted_offset[_u16](uint16_t	offset -> Qm base -> Rn offset -> Qm base -> Rn	VLDRH.S32 Qd,[Rn,Qm,UXTW #1] VLDRH.U16 Qd,[Rn,Qm,UXTW		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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 -> 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	Qd,[Rn,Qm,UXTW #1] 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) uint8x16_t [_arm_]vldrbq_gather_offset[_u8](uint8_t	base -> Rn offset -> Qm base -> Rn	VLDRB.S32 Qd,[Rn,Qm] VLDRB.U8 Qd,[Rn,Qm]	Qd -> result Qd -> result	MVE MVE
const * base, uint8x16_t offset) uint16x8 t [arm yldrbq gather offset[u16](uint8 t	offset -> Qm base -> Rn	VLDRB.U16 Qd,[Rn,Qm]	Od -> result	MVE
const * base, uint16x8_t offset) uint32x4_t [_arm_]vldrbq_gather_offset[_u32](uint8_t	offset -> Qm base -> Rn	VLDRB.U32 Qd,[Rn,Qm]	Qd -> result	MVE
const * base, uint32x4_t offset) int8x16_t [_arm_]vldrbq_gather_offset_z[_s8](int8_t const * base, uint8x16_t offset, mve_pred16_t p)	offset -> Qm 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	VLDRBT.Uo Qu,[RII,QIII] 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) int32x4_t [_arm_]vldrwq_gather_offset_z[_s32](int32_t	base -> Rn offset -> Qm base -> Rn	VLDRW.U32 Qd,[Rn,Qm] VMSR P0,Rp	Qd -> result Qd -> result	MVE MVE
const * base, uint32x4_t offset, mve_pred16_t p)	offset -> Qm p -> Rp	VPST VLDRWT.U32 Qd,[Rn,Qm]		
uint32x4_t [_arm_]vldrwq_gather_offset_z[_u32](uint32_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Rn,Qm]	Qd -> result	MVE
float32x4_t [arm_]vldrwq_gather_offset_z[_f32](float32_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Rn,Qm]	Qd -> result	MVE
int32x4_t [_arm_]vldrwq_gather_shifted_offset[_s32](int32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRW.U32 Qd,[Rn,Qm,UXTW #2]	Qd -> result	MVE
uint32x4_t [_armlvldrwq_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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [_arm_]vldrwq_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) int64x2_t [arm_]vldrdq_gather_offset_z[_s64](int64_t	base -> Rn offset -> Qm base -> Rn	VLDRD.U64 Qd,[Rn,Qm] VMSR P0,Rp	Qd -> result Qd -> result	MVE MVE
const * base, uint64x2_t offset, mve_pred16_t p)	offset -> Qm p -> Rp	VPST VLDRDT.U64 Qd,[Rn,Qm]		
uint64x2_t [_arm_]vldrdq_gather_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]	Qd -> result	MVE
int64x2_t [arm_]vldrdq_gather_shifted_offset[_s64](int64_t const * base, uint64x2_t offset)	base -> Rn offset -> Qm	VLDRD.U64 Qd,[Rn,Qm,UXTW #3]	Qd -> result	MVE
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
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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_]vldrdq_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] -> Qd2	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] -> Od4	VST40.16 {Qd - Qd4},[Rn] VST41.16 {Qd - Qd4},[Rn] VST42.16 {Qd - Qd4},[Rn] VST43.16 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_u32](uint32_t * addr, uint32x4x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.32 {Qd - Qd4},[Rn] VST41.32 {Qd - Qd4},[Rn] VST42.32 {Qd - Qd4},[Rn] VST43.32 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_f16](float16_t * addr, float16x8x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.16 {Qd - Qd4},[Rn] VST41.16 {Qd - Qd4},[Rn] VST42.16 {Qd - Qd4},[Rn] VST43.16 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_f32](float32_t * addr, float32x4x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.32 {Qd - Qd4},[Rn] VST41.32 {Qd - Qd4},[Rn] VST42.32 {Qd - Qd4},[Rn] VST43.32 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vstrbq[_s8](int8_t * base, int8x16_t value)	base -> Rn value -> Qd	VSTRB.8 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrbq[_s16](int8_t * base, int16x8_t value)	base -> Rn value -> Qd	VSTRB.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrbq[_s32](int8_t * base, int32x4_t value)	base -> Rn value -> Qd	VSTRB.32 Qd,[Rn]	void -> result	MVE

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

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void [_arm_]vstrhq_scatter_offset[_u16](uint16_t * base, uint16x8_t offset, uint16x8_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset[_u32](uint16_t * base, uint32x4_t offset, uint32x4_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset[_f16](float16_t * base, uint16x8_t offset, float16x8_t value)	base -> Rn offset -> Qm value -> Qd	VSTRH.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset_p[_s16](int16_t * base, uint16x8_t offset, int16x8_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset_p[_s32](int16_t * base, uint32x4_t offset, int32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset_p[_u16](uint16_t * base, uint16x8_t offset, uint16x8_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset_p[_u32](uint16_t * base, uint32x4_t offset, uint32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_offset_p[_f16](float16_t * base, uint16x8_t offset, float16x8_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrhq_scatter_shifted_offset[_s16](int16_t * base, uint16x8_t offset, int16x8_t value)	base -> Rn offset -> Qm value -> Od	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 #2]	void -> result	MVE
void [_arm_]vstrwq_scatter_shifted_offset_p[_f32](float32_t * base, uint32x4_t offset, float32x4_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm,UXTW #2]	void -> result	MVE
void [_arm_]vstrdq_scatter_base[_s64](uint64x2_t addr, const int offset, int64x2_t value)	addr -> Qn offset in +/- 8*[0127] value -> Qd	VSTRD.U64 Qd,[Qn,#offset]	void -> result	MVE
void [arm_]vstrdq_scatter_base[_u64](uint64x2_t addr, const int offset, uint64x2_t value)	addr -> Qn offset in +/- 8*[0127] value -> Qd	VSTRD.U64 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrdq_scatter_base_p[_s64](uint64x2_t addr, const int offset, int64x2_t value, mve_pred16_t p)	addr -> Qn offset in +/- 8*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrdq_scatter_base_p[_u64](uint64x2_t addr, const int offset, uint64x2_t value, mve_pred16_t p)	addr -> Qn offset in +/- 8*[0127] value -> Qd	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)	p -> Rp *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	VADDLVAT.S32		
uint64_t [arm_]vaddlvaq_p[_u32](uint64_t a,	p -> Rp a ->	RdaLo,RdaHi,Qm VMSR P0,Rp	[RdaHi,RdaLo]	MVE
uint32x4_t b, mve_pred16_t p)	[RdaHi,RdaLo]	VMSK PO,KP VPST	-> result	IVI V E
umiszka-t b, mve_predro_t p)	b -> Qm	VADDLVAT.U32	> resurt	
	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	MIVE
mve_predio_t p)	Prap	VADDLVT.S32	, result	
		RdaLo,RdaHi,Qm		
uint64_t [arm_]vaddlvq_p[_u32](uint32x4_t a,	a -> Qm	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
mve_pred16_t p)	p -> Rp	VPST	-> result	
		VADDLVT.U32		
	70.1	RdaLo,RdaHi,Qm	D) are
int32_t [arm_]vaddvaq[_s8](int32_t a, int8x16_t b)	a -> Rda b -> Qm	VADDVA.S8 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvaq[_s16](int32_t a, int16x8_t b)	a -> Rda b -> Qm	VADDVA.S16 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vaddvaq[_s32](int32_t a, int32x4_t b)	a -> Rda b -> Qm	VADDVA.S32 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vaddvaq[_u8](uint32_t a, uint8x16_t b)	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.Om	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	VPST		
:	p -> Rp	VADDVAT.S8 Rda,Qm	D.1- >14	MVE
int32_t [arm_]vaddvaq_p[_s16](int32_t a, int16x8_t b,	a -> Rda b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
mve_pred16_t p)	p -> QIII	VADDVAT.S16 Rda,Qm		
int32_t [arm_]vaddvaq_p[_s32](int32_t a, int32x4_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm	VPST	Ttut > Tootil	, 2
	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		
vint22 t [p -> Rp	VADDVAT.U16 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
uint32_t [arm_]vaddvaq_p[_u32](uint32_t a, uint32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm	VMSK PO,KP VPST	Kua -> resuit	IVI V E
b, mve_prearo_t p)	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	Rda -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VADDVT.S8 Rda,Qm		
int32_t [arm_]vaddvq_p[_s16](int16x8_t a,	a -> Om	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VADDVT.S16 Rda,Om		
int32_t [arm_]vaddvq_p[_s32](int32x4_t a,	a -> Qm	VMSR P0,Rp	Rda -> result	MVE
mt32_t [atm_]vaudvq_p[_832](int32x4_t a, mve_pred16_t p)	p -> Rp	VPST	rau -> result	1,1,1
	ļ	VADDVT.S32 Rda,Qm		<u> </u>
uint32_t [arm_]vaddvq_p[_u8](uint8x16_t a,	a -> Qm	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
	1	VADDVT.U8 Rda,Qm		<u> </u>
		VIMED DO P	D.1-	
uint32_t [arm_]vaddvq_p[_u16](uint16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Rda -> result	MVE

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	Rda -> result	MVE
int32_t [arm_]vmladavaq[_s8](int32_t add, int8x16_t m1, int8x16_t m2)	add -> Rda m1 -> Qn	VADDVT.U32 Rda,Qm VMLADAVA.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaq[_s16](int32_t add, int16x8_t m1, int16x8_t m2)	m2 -> Qm add -> Rda m1 -> Qn	VMLADAVA.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaq[_s32](int32_t add, int32x4_t m1, int32x4_t m2)	m2 -> Qm add -> Rda m1 -> Qn m2 -> Qm	VMLADAVA.S32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq[_u8](uint32_t add, uint8x16_t m1, uint8x16_t m2)	add -> Rda m1 -> Qn	VMLADAVA.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq[_u16](uint32_t add, uint16x8_t m1, uint16x8_t m2)	m2 -> Qm add -> Rda m1 -> Qn m2 -> Qm	VMLADAVA.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq[_u32](uint32_t add, uint32x4_t m1, uint32x4_t m2)	add -> Rda m1 -> Qn m2 -> Qm	VMLADAVA.U32 Rda,Qn,Qm	Rda -> result	MVE
$\label{eq:continuous} \begin{array}{ll} int32_t \ [_arm_]vmladavaq_p[_s8](int32_t \ add, int8x16_t \\ m1, int8x16_t \ m2, \ mve_pred16_t \ p) \end{array}$	add -> Rda m1 -> Qn m2 -> Qm	VMSR P0,Rp VPST VMLADAVAT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaq_p[_s16](int32_t add, int16x8_t m1, int16x8_t m2, mve_pred16_t p)	p -> Rp add -> Rda m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaq_p[_s32](int32_t add, int32x4_t m1, int32x4_t m2, mve_pred16_t p)	add -> Rda m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAT.S32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq_p[_u8](uint32_t add, uint8x16_t m1, uint8x16_t m2, mve_pred16_t p)	add -> Rda m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAT.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq_p[_u16](uint32_t add, uint16x8_t m1, uint16x8_t m2, mve_pred16_t p)	add -> Rda m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAT.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavaq_p[_u32](uint32_t add, uint32x4_t m1, uint32x4_t m2, mve_pred16_t p)	add -> Rda m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAT.U32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq[_s8](int8x16_t m1, int8x16_t m2)	m1 -> Qn m2 -> Qm	VMLADAV.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [_arm_]vmladavq[_s16](int16x8_t m1, int16x8_t m2)	m1 -> Qn m2 -> Qm	VMLADAV.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq[_s32](int32x4_t m1, int32x4_t m2)	m1 -> Qn m2 -> Qm	VMLADAV US Priz On One	Rda -> result	MVE
uint32_t [_arm_]vmladavq[_u8](uint8x16_t m1, uint8x16_t m2)	m1 -> Qn m2 -> Qm	VMLADAV.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavq[_u16](uint16x8_t m1, uint16x8_t m2)	m1 -> Qn m2 -> Qm	VMLADAV.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavq[_u32](uint32x4_t m1, uint32x4_t m2)	m1 -> Qn m2 -> Qm	VMLADAV.U32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq_p[_s8](int8x16_t m1, int8x16_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq_p[_s16](int16x8_t m1, int16x8_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq_p[_s32](int32x4_t m1, int32x4_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVT.S32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavq_p[_u8](uint8x16_t m1, uint8x16_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVT.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavq_p[_u16](uint16x8_t m1, uint16x8_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVT.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [_arm_]vmladavq_p[_u32](uint32x4_t m1, uint32x4_t m2, mve_pred16_t p)	m1 -> Qn m2 -> 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 add, int8x16_t m1, int8x16_t m2)	add -> Rda m1 -> Qn m2 -> Qm	VMLADAVAX.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaxq[_s16](int32_t add, int16x8_t m1, int16x8_t m2)	add -> Rda m1 -> Qn m2 -> Om	VMLADAVAX.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaxq[_s32](int32_t add, int32x4_t m1, int32x4_t m2)	add -> Rda m1 -> Qn	VMLADAVAX.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaxq_p[_s8](int32_t add, int8x16_t m1, int8x16_t m2, mve_pred16_t p)	m2 -> Qm add -> Rda m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAXT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavaxq_p[_s16](int32_t add, int16x8_t m1, int16x8_t m2, mve_pred16_t p)	add -> Rda m1 -> Qn m2 -> Qm	VMSR P0,Rp VPST VMLADAVAXT.S16	Rda -> result	MVE
int32_t [arm_]vmladavaxq_p[_s32](int32_t add, int32x4_t m1, int32x4_t m2, mve_pred16_t p)	p -> Rp add -> Rda m1 -> Qn m2 -> Qm p -> Rp	Rda,Qn,Qm VMSR P0,Rp VPST VMLADAVAXT.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavxq[_s8](int8x16_t m1, int8x16_t m2)	m1 -> Qn m2 -> Qm	VMLADAVX.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [_arm_]vmladavxq[_s16](int16x8_t m1, int16x8_t m2)	m1 -> Qn m2 -> Qm	VMLADAVX.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavxq[_s32](int32x4_t m1, int32x4_t m2) int32_t [arm_]vmladavxq_p[_s8](int8x16_t m1,	m1 -> Qn m2 -> Qm m1 -> On	VMLADAVX.S32 Rda,Qn,Qm VMSR P0,Rp	Rda -> result Rda -> result	MVE MVE
int8x16_t m2, mve_pred16_t p)	m2 -> Qm p -> Rp	VPST VMLADAVXT.S8 Rda,Qn,Qm	Rua -> icsuit	WVL
int32_t [arm_]vmladavxq_p[_s16](int16x8_t m1, int16x8_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVXT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavxq_p[_s32](int32x4_t m1, int32x4_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
int64_t [arm_]vmlaldavaq[_s16](int64_t add, int16x8_t m1, int16x8_t m2)	p -> Rp add -> [RdaHi,RdaLo] m1 -> Qn m2 -> Qm	VMLADAVXT.S32 Rda,Qn,Qm VMLALDAVA.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavaq[_s32](int64_t add, int32x4_t m1, int32x4_t m2)	add -> [RdaHi,RdaLo] m1 -> Qn m2 -> Qm	VMLALDAVA.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vmlaldavaq[_u16](uint64_t add, uint16x8_t m1, uint16x8_t m2)	add -> [RdaHi,RdaLo] m1 -> Qn m2 -> Qm	VMLALDAVA.U16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vmlaldavaq[_u32](uint64_t add, uint32x4_t m1, uint32x4_t m2)	add -> [RdaHi,RdaLo] m1 -> Qn m2 -> Qm	VMLALDAVA.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavaq_p[_s16](int64_t add, int16x8_t m1, int16x8_t m2, mve_pred16_t p)	add -> [RdaHi,RdaLo] m1 -> Qn m2 -> 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 add, int32x4_t m1, int32x4_t m2, mve_pred16_t p)	add -> [RdaHi,RdaLo] m1 -> Qn m2 -> 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 add, uint16x8_t m1, uint16x8_t m2, mve_pred16_t p)	add -> [RdaHi,RdaLo] m1 -> Qn m2 -> 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 add, uint32x4_t m1, uint32x4_t m2, mve_pred16_t p)	add -> [RdaHi,RdaLo] m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAT.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavq[_s16](int16x8_t m1, int16x8_t m2)	m1 -> Qn m2 -> Qm	VMLALDAV.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavq[_s32](int32x4_t m1, int32x4_t m2)	m1 -> Qn m2 -> 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 m1,	m1 -> Qn	VMLALDAV.U16	[RdaHi,RdaLo]	MVE
uint16x8_t m2) uint64_t [_arm_]vmlaldavq[_u32](uint32x4_t m1,	m2 -> Qm m1 -> Qn	RdaLo,RdaHi,Qn,Qm VMLALDAV.U32	-> result [RdaHi,RdaLo]	MVE
uint32x4_t m2) int64_t [_arm_lvmlaldavq_p[_s16](int16x8_t m1,	m2 -> Qm m1 -> Qn	RdaLo,RdaHi,Qn,Qm VMSR P0,Rp	-> result [RdaHi,RdaLo]	MVE
int16x8_t m2, mve_pred16_t p)	m2 -> Qm	VPST	-> result	WIVE
	p -> Rp	VMLALDAVT.S16 RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlaldavq_p[_s32](int32x4_t m1,	m1 -> Qn	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
int32x4_t m2, mve_pred16_t p)	m2 -> Qm	VPST VMLALDAVT.S32	-> result	
	p -> Rp	RdaLo,RdaHi,Qn,Qm		
uint64_t [_arm_]vmlaldavq_p[_u16](uint16x8_t m1,	m1 -> Qn	VMSR P0,Rp VPST	[RdaHi,RdaLo]	MVE
uint16x8_t m2, mve_pred16_t p)	m2 -> Qm p -> Rp	VMLALDAVT.U16	-> result	
1.51.5		RdaLo,RdaHi,Qn,Qm	CD 1 XX D 1 X 3	Nam
uint64_t [arm_]vmlaldavq_p[_u32](uint32x4_t m1, uint32x4_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Qm	VMSR P0,Rp VPST	[RdaHi,RdaLo] -> result	MVE
	p -> Rp	VMLALDAVT.U32		
int64_t [arm_]vmlaldavaxq[_s16](int64_t add, int16x8_t	add ->	RdaLo,RdaHi,Qn,Qm VMLALDAVAX.S16	[RdaHi,RdaLo]	MVE
m1, int16x8_t m2)	[RdaHi,RdaLo]	RdaLo,RdaHi,Qn,Qm	-> result	WIVE
	m1 -> Qn m2 -> Qm			
int64_t [arm_]vmlaldavaxq[_s32](int64_t add, int32x4_t	add ->	VMLALDAVAX.S32	[RdaHi,RdaLo]	MVE
m1, int32x4_t m2)	[RdaHi,RdaLo]	RdaLo,RdaHi,Qn,Qm	-> result	
	m1 -> Qn m2 -> Qm			
int64_t [arm_]vmlaldavaxq_p[_s16](int64_t add,	add ->	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
int16x8_t m1, int16x8_t m2, mve_pred16_t p)	[RdaHi,RdaLo] m1 -> On	VPST VMLALDAVAXT.S16	-> result	
	m2 -> Qm	RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlaldavaxq_p[_s32](int64_t add,	p -> Rp add ->	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
int32x4_t m1, int32x4_t m2, mve_pred16_t p)	[RdaHi,RdaLo]	VPST	-> result	MVE
	m1 -> Qn	VMLALDAVAXT.S32		
	m2 -> Qm p -> Rp	RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlaldavxq[_s16](int16x8_t m1,	m1 -> Qn	VMLALDAVX.S16	[RdaHi,RdaLo]	MVE
int16x8_t m2) int64_t [_arm_lvmlaldavxq[_s32](int32x4_t m1,	m2 -> Qm m1 -> On	RdaLo,RdaHi,Qn,Qm VMLALDAVX.S32	-> result [RdaHi,RdaLo]	MVE
int32x4_t m2)	m2 -> Qm	RdaLo,RdaHi,Qn,Qm	-> result	
int64_t [arm_]vmlaldavxq_p[_s16](int16x8_t m1, int16x8_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Qm	VMSR P0,Rp VPST	[RdaHi,RdaLo] -> result	MVE
mitono_t m2, mve_predro_t p/	p -> Rp	VMLALDAVXT.S16	Fiedure	
int64_t [arm_]vmlaldavxq_p[_s32](int32x4_t m1,	m1 -> Qn	RdaLo,RdaHi,Qn,Qm VMSR P0,Rp	[RdaHi,RdaLo]	MVE
int32x4_t m2, mve_pred16_t p)	m2 -> Qm	VPST	-> result	WIVE
	p -> Rp	VMLALDAVXT.S32 RdaLo,RdaHi,Qn,Qm		
int8x16_t [arm_]vmlaq[_n_s8](int8x16_t add, int8x16_t	add -> Qda	VMLA.S8 Qda,Qn,Rm	Qda -> result	MVE
m1, int8_t m2)	m1 -> Qn			
int16x8_t [arm_]vmlaq[_n_s16](int16x8_t add,	m2 -> Rm add -> Qda	VMLA.S16 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t m1, int16_t m2)	m1 -> Qn			
int32x4_t [arm_]vmlaq[_n_s32](int32x4_t add,	m2 -> Rm add -> Qda	VMLA.S32 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t m1, int32_t m2)	m1 -> Qn	, 17121 11552 Quai, Qui, 1111	Qua y resun	
uint8x16 t [arm]vmlaq[n u8](uint8x16 t add,	m2 -> Rm add -> Qda	VMLA.U8 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t m1, uint8_t m2)	m1 -> Qn	VIVIEZ I. CO Qua, Qui, Idii	Qua > resurt	14142
uint16x8_t [arm_]vmlaq[_n_u16](uint16x8_t add,	m2 -> Rm add -> Qda	VMLA.U16 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t m1, uint16_t m2)	m1 -> Qn	V WILA. O TO Qua,QII,KIII	Qua -> Iesuit	WIVE
pint22v4 + [arm lymlag[n v221/vint22v4 + a.i.]	m2 -> Rm	VMLA.U32 Oda,On,Rm	Ode > moult	MVE
uint32x4_t [arm_]vmlaq[_n_u32](uint32x4_t add, uint32x4_t m1, uint32_t m2)	add -> Qda m1 -> Qn	v wila.032 Qda,Qn,Km	Qda -> result	MVE
	m2 -> Rm	VA (CD DO D	01	NOTE
int8x16_t [_arm_]vmlaq_m[_n_s8](int8x16_t add, int8x16_t m1, int8_t m2, mve_pred16_t p)	add -> Qda m1 -> Qn	VMSR P0,Rp VPST	Qda -> result	MVE
mtox10_t iii1, iiito_t iii2, iiive_pied10_t p)	m2 -> Rm	VMLAT.S8 Qda,Qn,Rm		
	In \ Pn	i e	ı	1
int16x8 t[arm lymlag m[n s161/int16x8 t add	p -> Rp add -> Oda	VMSR P0.Rp	Oda -> result	MVE
int16x8_t [arm_]vmlaq_m[_n_s16](int16x8_t add, int16x8_t m1, int16_t m2, mve_pred16_t p)	add -> Qda m1 -> Qn m2 -> Rm	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 add,	add -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32x4_t m1, int32_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Rm	VPST VMLAT.S32 Qda,Qn,Rm		
	p -> Rp	, 1122111.002 Quii,Qii,11111		
uint8x16_t [arm_]vmlaq_m[_n_u8](uint8x16_t add,	add -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint8x16_t m1, uint8_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Rm	VPST VMLAT.U8 Qda,Qn,Rm		
	p -> Rp	VIVILAT.00 Qua,Qii,Kiii		
uint16x8_t [arm_]vmlaq_m[_n_u16](uint16x8_t add,	add -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint16x8_t m1, uint16_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Rm	VPST VMLAT.U16 Qda,Qn,Rm		
	p -> Rp	VMLAT.010 Qda,Qii,Kiii		
uint32x4_t [arm_]vmlaq_m[_n_u32](uint32x4_t add,	add -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint32x4_t m1, uint32_t m2, mve_pred16_t p)	m1 -> Qn	VPST		
	m2 -> Rm p -> Rp	VMLAT.U32 Qda,Qn,Rm		
int8x16_t [arm_]vmlasq[_n_s8](int8x16_t m1,	m1 -> Qda	VMLAS.S8 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t m2, int8_t add)	m2 -> Qn			
int16x8_t [arm_]vmlasq[_n_s16](int16x8_t m1,	add -> Rm m1 -> Qda	VMLAS.S16 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t m2, int16_t add)	m2 -> Qua	VWLAS.510 Qua,Qii,Kiii	Qua -> resuit	MVE
	add -> Rm			
int32x4_t [_arm_]vmlasq[_n_s32](int32x4_t m1,	m1 -> Qda	VMLAS.S32 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t m2, int32_t add)	m2 -> Qn add -> Rm			
uint8x16_t [arm_]vmlasq[_n_u8](uint8x16_t m1,	m1 -> Qda	VMLAS.U8 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t m2, uint8_t add)	m2 -> Qn			
uint16x8_t [arm_]vmlasq[_n_u16](uint16x8_t m1,	add -> Rm m1 -> Qda	VMLAS.U16 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t m2, uint16_t add)	m2 -> Qua	VMLAS.010 Qda,Qii,Kiii	Qua -> resuit	MIVE
	add -> Rm			
uint32x4_t [arm_]vmlasq[_n_u32](uint32x4_t m1,	m1 -> Qda	VMLAS.U32 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t m2, uint32_t add)	m2 -> Qn add -> Rm			
int8x16_t [arm_]vmlasq_m[_n_s8](int8x16_t m1,	m1 -> Qda	VMSR P0,Rp	Qda -> result	MVE
int8x16_t m2, int8_t add, mve_pred16_t p)	m2 -> Qn	VPST		
	add -> Rm p -> Rp	VMLAST.S8 Qda,Qn,Rm		
int16x8_t [arm_]vmlasq_m[_n_s16](int16x8_t m1,	m1 -> Qda	VMSR P0,Rp	Qda -> result	MVE
int16x8_t m2, int16_t add, mve_pred16_t p)	m2 -> Qn	VPST		
	add -> Rm p -> Rp	VMLAST.S16 Qda,Qn,Rm		
int32x4_t [arm_]vmlasq_m[_n_s32](int32x4_t m1,	m1 -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32x4_t m2, int32_t add, mve_pred16_t p)	m2 -> Qn	VPST		
	add -> Rm	VMLAST.S32 Qda,Qn,Rm		
uint8x16_t [arm_]vmlasq_m[_n_u8](uint8x16_t m1,	p -> Rp m1 -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint8x16_t m2, uint8_t add, mve_pred16_t p)	m2 -> Qn	VPST	Qua y resun	1,1,12
	add -> Rm	VMLAST.U8 Qda,Qn,Rm		
uint16x8_t [arm_]vmlasq_m[_n_u16](uint16x8_t m1,	p -> Rp m1 -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint16x8_t m2, uint16_t add, mve_pred16_t p)	m2 -> Qua	VMSK FO,KP VPST	Zau -> resuit	,
	add -> Rm	VMLAST.U16 Qda,Qn,Rm		
uint32x4 t [arm]vmlasq m[n u32](uint32x4 t m1,	p -> Rp m1 -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint32x4_t [arm_]vimasq_m[_n_u32](uint32x4_t in1, uint32x4_t m2, uint32_t add, mve_pred16_t p)	m2 -> Qua	VPST	Qua -> resuit	MIVE
- · · · - · - · · - · · · · · ·	add -> Rm	VMLAST.U32 Qda,Qn,Rm		
int32_t [arm_]vmlsdavaq[_s8](int32_t a, int8x16_t b,	p -> Rp a -> Rda	VMLSDAVA.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [armjvmisdavaq[_s8](int32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn	V MLSDA V A.SO KUR,QII,QIII	Nua -> 108uit	IVI V E
	c -> Qm			
int32_t [arm_]vmlsdavaq[_s16](int32_t a, int16x8_t b,	a -> Rda	VMLSDAVA.S16 Rda,Qn,Qm	Rda -> result	MVE
int16x8_t c)	b -> Qn c -> Qm			
int32_t [arm_]vmlsdavaq[_s32](int32_t a, int32x4_t b,	a -> Rda	VMLSDAVA.S32 Rda,Qn,Qm	Rda -> result	MVE
int32x4_t c)	b -> Qn			
int32_t [arm_]vmlsdavaq_p[_s8](int32_t a, int8x16_t b,	c -> Qm a -> Rda	VMSR P0,Rp	Rda -> result	MVE
int8x16_t c, mve_pred16_t p)	b -> Qn	VMSR FO,RP VPST	rea -/ resuit	171 7 15
	c -> Qm	VMLSDAVAT.S8 Rda,Qn,Qm		
int22 t [arm lymledayag n[a161/int22 t a int16+0 t	p -> Rp a -> Rda	VMSR P0,Rp	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	VMSR PO,RP VPST	Kua -> Iesuit	IVI V E
	c -> Qm	VMLSDAVAT.S16 Rda,Qn,Qm		
	p -> Rp		1	

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32_t [arm_]vmlsdavaq_p[_s32](int32_t a, int32x4_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, int32x4_t c, mve_pred16_t p)	b -> Qn c -> Qm	VPST VMLSDAVAT.S32 Rda,Qn,Qm		
int32_t [arm_]vmlsdavq[_s8](int8x16_t a, int8x16_t b)	p -> Rp a -> Qn	VMLSDAV.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavq[_s16](int16x8_t a, int16x8_t b)	b -> Qm a -> Qn	VMLSDAV.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavq[_s32](int32x4_t a, int32x4_t b)	b -> Qm a -> Qn	VMLSDAV.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavq_p[_s8](int8x16_t a, int8x16_t	b -> Qm a -> On	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMLSDAVT.S8 Rda,Qn,Qm	redu > resure	WYE
int32_t [_arm_]vmlsdavq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavq_p[_s32](int32x4_t a, int32x4_t	a -> Qn	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMLSDAVT.S32 Rda,Qn,Qm		
int32_t [arm_]vmlsdavaxq[_s8](int32_t a, int8x16_t b,	a -> Rda	VMLSDAVAX.S8 Rda,Qn,Qm	Rda -> result	MVE
int8x16_t c)	b -> Qn c -> Qm			
int32_t [_arm_]vmlsdavaxq[_s16](int32_t a, int16x8_t b, int16x8_t c)	a -> Rda b -> Qn c -> Qm	VMLSDAVAX.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavaxq[_s32](int32_t a, int32x4_t b,	a -> Rda	VMLSDAVAX.S32 Rda,Qn,Qm	Rda -> result	MVE
int32x4_t c)	b -> Qn c -> Qm	1112557117111.532 Rdu, Qii, Qiii	redu > resure	WYE
int32_t [arm_]vmlsdavaxq_p[_s8](int32_t a, int8x16_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, int8x16_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Qm	VMLSDAVAXT.S8 Rda,Qn,Qm		
int32_t [arm_]vmlsdavaxq_p[_s16](int32_t a, int16x8_t	p -> Rp a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, int16x8_t c, mve_pred16_t p)	b -> Qn	VPST	Rua -> resurt	WIVE
, – , – 1,	c -> Qm	VMLSDAVAXT.S16		
	p -> Rp	Rda,Qn,Qm		
int32_t [arm_]vmlsdavaxq_p[_s32](int32_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> Rda b -> On	VMSR P0,Rp VPST	Rda -> result	MVE
b, mi32x4_t c, mve_pred1o_t p)	c -> Qm	VMLSDAVAXT.S32		
	p -> Rp	Rda,Qn,Qm		
int32_t [arm_]vmlsdavxq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMLSDAVX.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavxq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMLSDAVX.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMLSDAVX.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavxq_p[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
	p -> Rp	VMLSDAVXT.S8 Rda,Qn,Qm	D.I Is	Mari
int32_t [_arm_]vmlsdavxq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
int32_t [arm_]vmlsdavxq_p[_s32](int32x4_t a,	p -> Rp a -> Qn	VMLSDAVXT.S16 Rda,Qn,Qm VMSR P0,Rp	Rda -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VMLSDAVXT.S32 Rda,Qn,Qm	FD 1 *** *) are
int64_t [_arm_]vmlsldavaq[_s16](int64_t a, int16x8_t b, int16x8_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLSLDAVA.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlsldavaq[_s32](int64_t a, int32x4_t b,	a ->	VMLSLDAVA.S32	[RdaHi,RdaLo]	MVE
int32x4_t c)	[RdaHi,RdaLo] b -> Qn	RdaLo,RdaHi,Qn,Qm	-> result	
int64_t [arm_]vmlsldavaq_p[_s16](int64_t a, int16x8_t	c -> Qm a ->	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
b, int16x8_t c, mve_pred16_t p)	[RdaHi,RdaLo]	VPST	-> result	1
	b -> Qn	VMLSLDAVAT.S16		
	c -> Qm	RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlsldavaq_p[_s32](int64_t a, int32x4_t	p -> Rp a ->	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
b, int32x4_t c, mve_pred16_t p)	[RdaHi,RdaLo]	VPST	-> result	1
-	b -> Qn	VMLSLDAVAT.S32		
	c -> Qm	RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlsldavq[_s16](int16x8_t a, int16x8_t	p -> Rp a -> Qn b > Om	VMLSLDAV.S16	[RdaHi,RdaLo]	MVE
b) int64_t [arm_]vmlsldavq[_s32](int32x4_t a, int32x4_t	b -> Qm a -> Qn	RdaLo,RdaHi,Qn,Qm VMLSLDAV.S32	-> result [RdaHi,RdaLo]	MVE
b)	b -> Qm	RdaLo,RdaHi,Qn,Qm	-> result	1

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	VPST	-> result	
	p -> Rp	VMLSLDAVT.S16		
int64_t [arm_]vmlsldavq_p[_s32](int32x4_t a, int32x4_t	a -> Qn	RdaLo,RdaHi,Qn,Qm VMSR P0,Rp	[RdaHi,RdaLo]	MVE
b, mve_pred16_t p)	b -> Qm	VPST	-> result	M V E
	p -> Rp	VMLSLDAVT.S32		
		RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlsldavaxq[_s16](int64_t a, int16x8_t b,	a ->	VMLSLDAVAX.S16	[RdaHi,RdaLo]	MVE
int16x8_t c)	[RdaHi,RdaLo] b -> Qn	RdaLo,RdaHi,Qn,Qm	-> result	
	c -> Qm			
int64_t [arm_]vmlsldavaxq[_s32](int64_t a, int32x4_t b,	a ->	VMLSLDAVAX.S32	[RdaHi,RdaLo]	MVE
int32x4_t c)	[RdaHi,RdaLo]	RdaLo,RdaHi,Qn,Qm	-> result	
	b -> Qn c -> Qm			
int64_t [arm_]vmlsldavaxq_p[_s16](int64_t a, int16x8_t	a->	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
b, int16x8_t c, mve_pred16_t p)	[RdaHi,RdaLo]	VPST	-> result	M V E
	b -> Qn	VMLSLDAVAXT.S16		
	c -> Qm	RdaLo,RdaHi,Qn,Qm		
int64 t [arm]vmlsldavaxq p[s32](int64 t a, int32x4 t	p -> Rp	VMCD DO D.	[DdoII; DdoI o]	MVE
into4_t [arm_]vmisidavaxq_p[_s52](into4_t a, int52x4_t b, int32x4_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo]	VMSR P0,Rp VPST	[RdaHi,RdaLo] -> result	IVI V E
-,	b -> Qn	VMLSLDAVAXT.S32	, result	
	c -> Qm	RdaLo,RdaHi,Qn,Qm		
	p -> Rp			
int64_t [arm_]vmlsldavxq[_s16](int16x8_t a, int16x8_t	a -> Qn	VMLSLDAVX.S16	[RdaHi,RdaLo] -> result	MVE
b) int64_t [arm_]vmlsldavxq[_s32](int32x4_t a, int32x4_t	b -> Qm a -> On	RdaLo,RdaHi,Qn,Qm VMLSLDAVX.S32	-> result [RdaHi,RdaLo]	MVE
b)	b -> Qm	RdaLo,RdaHi,Qn,Qm	-> result	WYE
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	VPST	-> result	
	p -> Rp	VMLSLDAVXT.S16		
int64_t [arm_]vmlsldavxq_p[_s32](int32x4_t a,	a -> On	RdaLo,RdaHi,Qn,Qm VMSR P0,Rp	[RdaHi,RdaLo]	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VMSK FO,KP VPST	-> result	IVIVE
	p -> Rp	VMLSLDAVXT.S32		
		RdaLo,RdaHi,Qn,Qm		
int8x16_t [arm_]vhaddq[_n_s8](int8x16_t a, int8_t b)	a -> Qn	VHADD.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vhaddq[_n_s16](int16x8_t a, int16_t b)	b -> Rm a -> On	VHADD.S16 Qd,Qn,Rm	Od -> result	MVE
	b -> Rm	7 . 7 .	`	
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	a -> Qn	VHADD.U8 Qd,Qn,Rm	Qd -> result	MVE
b) uint16x8 t [arm]vhaddq[n u16](uint16x8 t a,	b -> Rm a -> Qn	VHADD.U16 Qd,Qn,Rm	Od -> result	MVE
uint16_t b)	b->Rm	2 . 2 .	`	
uint32x4_t [arm_]vhaddq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VHADD.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vhaddq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VHADD.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vhaddq[_s16](int16x8_t a, int16x8_t b)	b -> Qm a -> Qn	VHADD.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
introxo_t [arm_]vnauuq[_sro](introxo_t a, introxo_t b)	b -> Qm	VIIADD.310 Qu,Qii,Qiii	Qu -> resuit	MIVE/NEON
int32x4_t [arm_]vhaddq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Om	VHADD.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vhaddq[_u8](uint8x16_t a, uint8x16_t	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	Od -> result	MVE/NEON
uint16x8_t b)	b -> Qm	2 . 2 . 2	`	
uint32x4_t [arm_]vhaddq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VHADD.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vhaddq_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	VHADDT.S8 Qd,Qn,Rm		
	p -> Rp	VMCD DO D	01 2 1	MVE
int16x8_t [arm_]vhaddq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
incrozo_t a, incro_t o, inve_pieuro_t p)	b -> Rm	VHADDT.S16 Qd,Qn,Rm		
	p -> Rp	2		
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		
	the N Dana	VHADDT.S32 Qd,Qn,Rm	1	•

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	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)	p -> Rp 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	VMSR P0,Rp VPST VHADDT.S8 Qd,Qn,Qm	Qd -> result	MVE
$int16x8_t \ [_arm_]vhaddq_m[_s16](int16x8_t \ inactive, \\ int16x8_t \ a, \ int16x8_t \ b, \ mve_pred16_t \ p)$	p -> Rp inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VHADDT.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4_t \ [_arm_]vhaddq_m [_s32] (int32x4_t \ inactive, \\ int32x4_t \ a, \ int32x4_t \ b, \ mve_pred16_t \ p)$	p -> Rp inactive -> Qd a -> Qn b -> Qm	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)	p -> Rp 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
$ \begin{array}{l} int16x8_t \ [_arm_]vhaddq_x[_n_s16](int16x8_t \ a, int16_t \\ b, mve_pred16_t \ p) \end{array} $	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	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)	p -> Rp a -> Qn b -> Rm p -> Rp	VHADDT.S32 Qd,Qn,Rin 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
$ \begin{array}{lll} int16x8_t & [_arm_]vhaddq_x[_s16](int16x8_t \ a, int16x8_t \\ b, mve_pred16_t \ p) \end{array} $	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.S16 Qd,Qn,Qm	Qd -> result	MVE
$\begin{array}{l} int32x4_t \; [_arm_]vhaddq_x[_s32](int32x4_t \; a, int32x4_t \\ b, mve_pred16_t \; p) \end{array}$	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, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHCADDT.S8 Qd,Qn,Qm,#90	Qd -> result	MVE
int16x8_t [arm_]vhcaddq_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 VHCADDT.S16 Qd,Qn,Qm,#90	Qd -> result	MVE
int32x4_t [arm_]vhcaddq_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 VHCADDT.S32 Qd,Qn,Qm,#90	Qd -> result	MVE
int8x16_t [arm_]vhcaddq_rot90_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHCADDT.S8 Qd,Qn,Qm,#90	Qd -> result	MVE
int16x8_t [arm_]vhcaddq_rot90_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHCADDT.S16 Qd,Qn,Qm,#90	Qd -> result	MVE
int32x4_t [_arm_]vhcaddq_rot90_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHCADDT.S32 Qd,Qn,Qm,#90	Qd -> result	MVE
int8x16_t [arm_]vhcaddq_rot270[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VHCADD.S8 Qd,Qn,Qm,#270	Qd -> result	MVE
int16x8_t [_arm_]vhcaddq_rot270[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VHCADD.S16 Qd,Qn,Qm,#270	Qd -> result	MVE
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, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VHCADDT.S8 Qd,Qn,Qm,#270	Qd -> result	MVE
int16x8_t [arm_]vhcaddq_rot270_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 VHCADDT.S16 Qd,Qn,Qm,#270	Qd -> result	MVE
int32x4_t [arm_]vhcaddq_rot270_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 VHCADDT.S32 Qd,Qn,Qm,#270	Qd -> result	MVE
int8x16_t [arm_]vhcaddq_rot270_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	p -> Rp a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHCADDT.S8 Qd,Qn,Qm,#270	Qd -> result	MVE
int16x8_t [arm_]vhcaddq_rot270_x[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHCADDT.S16 Qd,Qn,Qm,#270	Qd -> result	MVE
int32x4_t [arm_]vhcaddq_rot270_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHCADDT.S32 Qd,Qn,Qm,#270	Qd -> result	MVE
int8x16_t [arm_]vhsubq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VHSUB.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [_arm_]vhsubq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VHSUB.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vhsubq[_n_s32](int32x4_t a, int32_t b)	a -> Qn	VHSUB.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vhsubq[_n_u8](uint8x16_t a, uint8_t b)	b -> Rm a -> Qn b -> Rm	VHSUB.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vhsubq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VHSUB.U16 Qd,Qn,Rm	Qd -> result	MVE
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 b -> Om	VHSUB.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vhsubq[_s16](int16x8_t a, int16x8_t b)	b -> Qm a -> Qn b -> Om	VHSUB.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vhsubq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VHSUB.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vhsubq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VHSUB.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vhsubq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VHSUB.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vhsubq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VHSUB.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vhsubq_m[_n_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VHSUBT.S8 Qd,Qn,Rm		
	p -> Rp	7 . 7 .		
int16x8_t [arm_]vhsubq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
miroxo_t u, miro_t o, miro_predio_t p)	b -> Rm	VHSUBT.S16 Qd,Qn,Rm		
st32v4 + [arm lyheuha m[n e32]/int32v4 + inactiva	p -> Rp inactive -> Od	VMSR P0,Rp	Od -> result	MVE
nt32x4_t [arm_]vhsubq_m[_n_s32](int32x4_t inactive, nt32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn	VMSK FO,KP VPST	Qu -> resuit	IVI V E
	b -> Rm	VHSUBT.S32 Qd,Qn,Rm		
uint8x16_t [arm_]vhsubq_m[_n_u8](uint8x16_t	p -> Rp inactive -> Od	VMSR P0,Rp	Qd -> result	MVE
inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm p -> Rp	VHSUBT.U8 Qd,Qn,Rm		
uint16x8_t [arm_]vhsubq_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 p -> Rp	VHSUBT.U16 Qd,Qn,Rm		
uint32x4_t [arm_]vhsubq_m[_n_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VHSUBT.U32 Qd,Qn,Rm		
	p -> Rn	VIISOBT.032 Qu,Qii,Kiii		
int8x16_t [_arm_]vhsubq_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 VHSUBT.S8 Od,On,Om		
	p -> Rp			
int16x8_t [_arm_]vhsubq_m[_s16](int16x8_t inactive,	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	b -> Qm	VHSUBT.S16 Qd,Qn,Qm		
	p -> Rp			
int32x4_t [arm_]vhsubq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
mi32x4_t a, mi32x4_t b, mve_pred10_t p)	b -> Qm	VHSUBT.S32 Qd,Qn,Qm		
uint8x16_t [arm_]vhsubq_m[_u8](uint8x16_t inactive,	p -> Rp inactive -> Od	VMSR P0,Rp	Od -> result	MVE
uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn	VMSK FO,KP VPST	Qu -> resuit	MVE
	b -> Qm	VHSUBT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vhsubq_m[_u16](uint16x8_t inactive,	p -> Rp 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	VHSUBT.U16 Qd,Qn,Qm		
uint32x4_t [arm_]vhsubq_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	VPST		
	b -> Qm p -> Rp	VHSUBT.U32 Qd,Qn,Qm		
int8x16_t [arm_]vhsubq_x[_n_s8](int8x16_t a, int8_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Rm p -> Rp	VPST VHSUBT.S8 Qd,Qn,Rm		
int16x8_t [arm_]vhsubq_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]vhsubq x[n s32](int32x4 t a, int32 t	p -> Rp a -> Qn	VHSUBT.S16 Qd,Qn,Rm VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST	(
uint8x16_t [arm_]vhsubq_x[_n_u8](uint8x16_t a,	p -> Rp a -> Qn	VHSUBT.S32 Qd,Qn,Rm VMSR P0,Rp	Qd -> result	MVE
uintox10_t [arm_]viisuoq_x[_n_uo](uintox10_t a, uint8_t b, mve_pred16_t p)	b -> Rm	VMSK FO,KP VPST	Qu -> Tesuit	IVI V E
1.150.15	p -> Rp	VHSUBT.U8 Qd,Qn,Rm	01	NAME.
uint16x8_t [arm_]vhsubq_x[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Qd -> result	MVE
_ · _ _ _ _ .	p -> Rp	VHSUBT.U16 Qd,Qn,Rm		
uint32x4_t [_arm_]vhsubq_x[_n_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VHSUBT.U32 Qd,Qn,Rm		
int8x16_t [arm_]vhsubq_x[_s8](int8x16_t a, int8x16_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VHSUBT.S8 Qd,Qn,Qm		
int16x8_t [arm_]vhsubq_x[_s16](int16x8_t a, int16x8_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
int32x4_t [arm_]vhsubq_x[_s32](int32x4_t a, int32x4_t	p -> Rp a -> Qn	VHSUBT.S16 Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
int32x4_t [arm_]vnsubq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qu -> resuit	IVI V E
	p -> Rp	VHSUBT.S32 Qd,Qn,Qm		1

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [arm_]vhsubq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VHSUBT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vhsubq_x[_u16](uint16x8_t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VHSUBT.U16 Qd,Qn,Qm		
uint32x4_t [arm_]vhsubq_x[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VHSUBT.U32 Qd,Qn,Qm		
nt8x16_t [arm_]vrhaddq[_s8](int8x16_t a, int8x16_t b)	a -> Qn	VRHADD.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
	b -> Qm			
nt16x8_t [arm_]vrhaddq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VRHADD.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
nt32x4_t [arm_]vrhaddq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VRHADD.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
nint8x16_t [arm_]vrhaddq[_u8](uint8x16_t a, nint8x16_t b)	a -> Qn b -> Qm	VRHADD.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vrhaddq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VRHADD.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vrhaddq[_u32](uint32x4_t a,	a -> Qn b -> Qm	VRHADD.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t b) int8x16_t [arm_]vrhaddq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VRHADDT.S8 Qd,Qn,Qm		
int16x8_t [arm_]vrhaddq_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	Qu -> result	WIVE
The second secon	b -> Qm	VRHADDT.S16 Qd,Qn,Qm		
	p -> Rp			
nt32x4_t [arm_]vrhaddq_m[_s32](int32x4_t inactive, nt32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
nt32x4_t a, int32x4_t b, nive_pred10_t p)	b -> Qm	VRHADDT.S32 Qd,Qn,Qm		
	p -> Rp			
int8x16_t [arm_]vrhaddq_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
tint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VRHADDT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vrhaddq_m[_u16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
nactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VRHADDT.U16 Qd,Qn,Qm		
uint32x4_t [arm_]vrhaddq_m[_u32](uint32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
nactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn	VMSK FO,KP VPST	Qu -> resuit	WIVE
maeuve, amesza igt a, amesza igt e, mvegpieuregt p/	b -> Qm	VRHADDT.U32 Qd,Qn,Qm		
	p -> Rp			
nt8x16_t [arm_]vrhaddq_x[_s8](int8x16_t a, int8x16_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
o, mve_pred16_t p)	b -> Qm p -> Rp	VPST VRHADDT.S8 Qd,Qn,Qm		
nt16x8_t [arm_]vrhaddq_x[_s16](int16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
nt16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Qu's resuit	
	p -> Rp	VRHADDT.S16 Qd,Qn,Qm		
nt32x4_t [arm_]vrhaddq_x[_s32](int32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
nt32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VRHADDT.S32 Qd,Qn,Qm		
uint8x16_t [arm_]vrhaddq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
iint8x16_t b, mve_pred16_t p)	b -> Qm	VPST	Q	1
	p -> Rp	VRHADDT.U8 Qd,Qn,Qm		
tint16x8_t [arm_]vrhaddq_x[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
iint16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VRHADDT.U16 Qd,Qn,Qm		
uint32x4_t [arm_]vrhaddq_x[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
nint32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Q	1
	p -> Rp	VRHADDT.U32 Qd,Qn,Qm		
loat16x8_t [_arm_]vfmaq[_n_f16](float16x8_t add,	add -> Qda	VFMA.F16 Qda,Qn,Rm	Qda -> result	MVE/NEON
loat16x8_t m1, float16_t m2)	m1 -> Qn m2 -> Rm			1
loat32x4_t [arm_]vfmaq[_n_f32](float32x4_t add,	add -> Qda	VFMA.F32 Qda,Qn,Rm	Qda -> result	MVE/NEON
loat32x4_t m1, float32_t m2)	m1 -> Qn			1
	m2 -> Rm			1
loat16x8_t [_arm_]vfmaq_m[_n_f16](float16x8_t add,	add -> Qda	VMSR P0,Rp	Qda -> result	MVE
loat16x8_t m1, float16_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Rm	VPST VFMAT.F16 Qda,Qn,Rm		1
	p -> Rp	. 1 . 1 . 10 Qua,Qii,Niii		1
loat32x4_t [arm_]vfmaq_m[_n_f32](float32x4_t add,	add -> Qda	VMSR P0,Rp	Qda -> result	MVE
10at32x4_t [arm_]vrmaq_m_n_132](110at32x4_t aud,				
loat32x4_t m1, float32_t m2, mve_pred16_t p)	m1 -> Qn m2 -> Rm	VPST VFMAT.F32 Qda,Qn,Rm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [_arm_]vfmaq[_f16](float16x8_t add, float16x8_t m1, float16x8_t m2)	add -> Qda m1 -> Qn m2 -> Qm	VFMA.F16 Qda,Qn,Qm	Qda -> result	MVE/NEON
float32x4_t [_arm_]vfmaq[_f32](float32x4_t add, float32x4_t m1, float32x4_t m2)	add -> Qda m1 -> Qn m2 -> Qm	VFMA.F32 Qda,Qn,Qm	Qda -> result	MVE/NEON
float16x8_t [_arm_]vfmaq_m[_f16](float16x8_t add, float16x8_t m1, float16x8_t m2, mve_pred16_t p)	add -> Qda m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VFMAT.F16 Qda,Qn,Qm	Qda -> result	MVE
float32x4_t [_arm_]vfmaq_m[_f32](float32x4_t add, float32x4_t m1, float32x4_t m2, mve_pred16_t p)	add -> Qda m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VFMAT.F32 Qda,Qn,Qm	Qda -> result	MVE
float16x8_t [_arm_]vfmasq[_n_f16](float16x8_t m1, float16x8_t m2, float16_t add)	m1 -> Qda m2 -> Qn add -> Rm	VFMAS.F16 Qda,Qn,Rm	Qda -> result	MVE
float32x4_t [_arm_]vfmasq[_n_f32](float32x4_t m1, float32x4_t m2, float32_t add)	m1 -> Qda m2 -> Qn add -> Rm	VFMAS.F32 Qda,Qn,Rm	Qda -> result	MVE
float16x8_t [_arm_]vfmasq_m[_n_f16](float16x8_t m1, float16x8_t m2, float16_t add, mve_pred16_t p)	m1 -> Qda m2 -> Qn add -> Rm p -> Rp	VMSR P0,Rp VPST VFMAST.F16 Qda,Qn,Rm	Qda -> result	MVE
float32x4_t [_arm_]vfmasq_m[_n_f32](float32x4_t m1, float32x4_t m2, float32_t add, mve_pred16_t p)	m1 -> Qda m2 -> Qn add -> Rm p -> Rp	VMSR P0,Rp VPST VFMAST.F32 Qda,Qn,Rm	Qda -> result	MVE
float16x8_t [_arm_]vfmsq[_f16](float16x8_t add, float16x8_t m1, float16x8_t m2)	add -> Qda m1 -> Qn m2 -> Qm	VFMS.F16 Qda,Qn,Qm	Qda -> result	MVE/NEON
float32x4_t [_arm_]vfmsq[_f32](float32x4_t add, float32x4_t m1, float32x4_t m2)	add -> Qda m1 -> Qn m2 -> Qm	VFMS.F32 Qda,Qn,Qm	Qda -> result	MVE/NEON
float16x8_t [_arm_]vfmsq_m[_f16](float16x8_t add, float16x8_t m1, float16x8_t m2, mve_pred16_t p)	add -> Qda m1 -> Qn m2 -> Qm p -> Rp	VMSR P0,Rp VPST VFMST.F16 Qda,Qn,Qm	Qda -> result	MVE
float32x4_t [_arm_]yfmsq_m[_f32](float32x4_t add, float32x4_t m1, float32x4_t m2, mve_pred16_t p)	add -> Qda m1 -> Qn m2 -> 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 c -> Qm	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)	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]	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 -> Om	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	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VRMULHT.S8 Qd,Qn,Qm		
int16x8_t [arm_]vrmulhq_x[_s16](int16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VRMULHT.S16 Qd,Qn,Qm		
int32x4_t [arm_]vrmulhq_x[_s32](int32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VRMULHT.S32 Qd,Qn,Qm		
uint8x16_t [arm_]vrmulhq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VRMULHT.U8 Qd,Qn,Qm		
uint16x8_t [arm_]vrmulhq_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	VRMULHT.U16 Qd,Qn,Qm		
uint32x4_t [_arm_]vrmulhq_x[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
1.150.15	p -> Rp	VRMULHT.U32 Qd,Qn,Qm	0.1	MENEON
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 -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCVTAT.S16.F16 Qd,Qm		
int32x4_t [arm_]vcvtaq_m[_s32_f32](int32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCVTAT.S32.F32 Qd,Qm		
uint16x8_t [arm_]vcvtaq_m[_u16_f16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCVTAT.U16.F16 Qd,Qm		
uint32x4_t [arm_]vcvtaq_m[_u32_f32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCVTAT.U32.F32 Qd,Qm		
int16x8_t [arm_]vcvtaq_x_s16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
:	0	VCVTAT.S16.F16 Qd,Qm	0.1) (T T
int32x4_t [arm_]vcvtaq_x_s32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
	0	VCVTAT.S32.F32 Qd,Qm	0.1	MVE
uint16x8_t [_arm_]vcvtaq_x_u16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
uint32x4 t [arm]vcvtaq x u32 f32(float32x4 t a,	0	VCVTAT.U16.F16 Qd,Qm	0.1 >14	MVE
	a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mve_pred16_t p)	p -> Rp			
int16x8_t [arm_]vcvtnq_s16_f16(float16x8_t a)	a -> Qm	VCVTAT.U32.F32 Qd,Qm VCVTN.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4 t [arm]vcvtnq_s10_110(float10x6_t a)				MVE/NEON
	a -> Qm	VCVTN.S32.F32 Qd,Qm	Qd -> result	_
uint16x8_t [_arm_]vcvtnq_u16_f16(float16x8_t a)	a -> Qm	VCVTN.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vcvtnq_u32_f32(float32x4_t a)	a -> Qm	VCVTN.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vcvtnq_m[_s16_f16](int16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
:::22-4 + F	p -> Rp	VCVTNT.S16.F16 Qd,Qm	01 , 1	MVE
int32x4_t [_arm_]vcvtnq_m[_s32_f32](int32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST VCVTNT.S32.F32 Qd,Qm		
uint16x8 t [arm]vcvtnq m[u16 f16](uint16x8 t	p -> Rp	VCVTN1.S32.F32 Qu,QIII VMSR P0,Rp	Od -> result	MVE
	inactive -> Qd	, 1	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VCVTNT.U16.F16 Qd,Qm		
uint32x4 t [arm]vcvtnq m[u32 f32](uint32x4 t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VMSK FO,KP VPST	Qu -> resuit	IVI V L
mactive, moat32x4_t a, mve_preuro_t p)	p -> Rp	VCVTNT.U32.F32 Qd,Qm		
int16x8_t [_arm_]vcvtnq_x_s16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mtroxe_t [arm_jvcvtnq_x_sro_fro(froatroxe_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR PO,RP VPST	Qu -> resuit	141 4 17
р.овто_с р/	P - 10p	VCVTNT.S16.F16 Qd,Qm		
int32x4 t [arm]vcvtnq x s32 f32(float32x4 t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mts2x4_t [atm_]vevtiiq_x_s32_132(110at32x4_t a, mve_pred16_t p)	p -> Rp	VMSK FO,KP VPST	Zu -> Icsuit	171 7 1
p.ca.ro_t p/	P - 10p	VCVTNT.S32.F32 Qd,Qm		
uint16x8_t [arm_]vcvtnq_x_u16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	a -> QIII p -> Rp	VMSR PO,RP VPST	Qu -> resuit	141 4 17
mvc_pred10_t p)	h -> Kh	VCVTNT.U16.F16 Qd,Qm		
uint32x4_t [_arm_]vcvtnq_x_u32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	a -> Qm p -> Rp	VMSR PO,RP VPST	Qu -> resuit	141 A TO
mrc_prod10_t p)	P -> KP	VCVTNT.U32.F32 Qd,Qm		
int16x8_t [arm_]vcvtpq_s16_f16(float16x8_t a)	a -> Qm	VCVTP.S16.F16 Qd,Qm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vcvtpq_u16_f16(float16x8_t a)	a -> Qm	VCVTP.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vcvtpq_u32_f32(float32x4_t a)	a -> Qm	VCVTP.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vcvtpq_m[_s16_f16](int16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCVTPT.S16.F16 Qd,Qm		
int32x4_t [arm_]vcvtpq_m[_s32_f32](int32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCVTPT.S32.F32 Qd,Qm		
uint16x8_t [_arm_]vcvtpq_m[_u16_f16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
: (20 4 (F 3 3 4 (F 20 620)/ : (20 4 (p -> Rp	VCVTPT.U16.F16 Qd,Qm	0.1 1:	MATE
uint32x4_t [_arm_]vcvtpq_m[_u32_f32](uint32x4_t	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	p -> Qm	VCVTPT.U32.F32 Qd,Qm		
int16x8_t [arm_]vcvtpq_x_s16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VMSK FO,KP VPST	Qu -> resuit	IVI V IL
mvc_pred1o_t p)	p -> Kp	VCVTPT.S16.F16 Qd,Qm		
int32x4_t [arm_]vcvtpq_x_s32_f32(float32x4_t a,	a -> Om	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu -> resuit	WIVE
mvc_pred10_t p)	p -> Kp	VCVTPT.S32.F32 Qd,Qm		
uint16x8_t [arm_]vcvtpq_x_u16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu > result	M v E
mve_predio_t p)	p > Kp	VCVTPT.U16.F16 Qd,Qm		
uint32x4_t [arm_]vcvtpq_x_u32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu > resurt	14142
-F	r · · · · · · · · · ·	VCVTPT.U32.F32 Qd,Qm		
int16x8_t [arm_]vcvtmq_s16_f16(float16x8_t a)	a -> Om	VCVTM.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4 t [arm]vevtmq_sr2_fr32(float32x4 t a)	a -> Om	VCVTM.S32.F32 Qd,Qm	Od -> result	MVE/NEON
uint16x8_t [arm_]vcvtmq_u16_f16(float16x8_t a)	a -> Qm	VCVTM.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vcvtmq_u32_f32(float32x4_t a)	a -> Qm	VCVTM.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vevtmq_m[_s16_f16](int16x8_t	inactive -> Od	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST	Qu -> resurt	WIVE
maeuve, noacrono_t a, mve_prearo_t p)	p -> Rp	VCVTMT.S16.F16 Qd,Qm		
int32x4_t [arm_]vcvtmq_m[_s32_f32](int32x4_t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Om	VPST	Qu' y resuit	
maeave, noaeszavijt a, mvejprearojt p	p -> Rp	VCVTMT.S32.F32 Qd,Qm		
uint16x8_t [_arm_]vcvtmq_m[_u16_f16](uint16x8_t	inactive -> Od	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST	Q	
	p -> Rp	VCVTMT.U16.F16 Qd,Qm		
uint32x4_t [arm_]vcvtmq_m[_u32_f32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Om	VPST		
	p -> Rp	VCVTMT.U32.F32 Qd,Qm		
int16x8_t [arm_]vcvtmq_x_s16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VCVTMT.S16.F16 Qd,Qm		
int32x4_t [arm_]vcvtmq_x_s32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VCVTMT.S32.F32 Qd,Qm		
uint16x8_t [arm_]vcvtmq_x_u16_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VCVTMT.U16.F16 Qd,Qm		
uint32x4_t [arm_]vcvtmq_x_u32_f32(float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VCVTMT.U32.F32 Qd,Qm		
float16x8_t [arm_]vcvtbq_f16_f32(float16x8_t a,	a -> Qd	VCVTB.F16.F32 Qd,Qm	Qd -> result	MVE
float32x4_t b)	b -> Qm			
float32x4_t [arm_]vcvtbq_f32_f16(float16x8_t a)	a -> Qm	VCVTB.F32.F16 Qd,Qm	Qd -> result	MVE
float16x8_t [arm_]vcvtbq_m_f16_f32(float16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		1
	p -> Rp	VCVTBT.F16.F32 Qd,Qm		1
float32x4_t [_arm_]vcvtbq_m_f32_f16(float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
fl-120-4 t 1 d - 220 215/2 115 0	p -> Rp	VCVTBT.F32.F16 Qd,Qm	04 : 1:	MVE
float32x4_t [arm_]vcvtbq_x_f32_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	1	1
floot16v9 t	0.501	VCVTT F16 F22 Od Om	04 5 10	MVE
float16x8_t [arm_]vcvttq_f16_f32(float16x8_t a,	a -> Qd	VCVTT.F16.F32 Qd,Qm	Qd -> result	MVE
float32x4_t b)	b -> Qm	VCVET PART CALL	01.	MATE
float32x4_t [_arm_]vcvttq_f32_f16(float16x8_t a)	a -> Qm	VCVTT.F32.F16 Qd,Qm	Qd -> result	MVE
float16x8_t [_arm_]vcvttq_m_f16_f32(float16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST	1	
g (20 4) F 7	p -> Rp	VCVTTT.F16.F32 Qd,Qm	0, ,	2472
float32x4_t [_arm_]vcvttq_m_f32_f16(float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		1
	p -> Rp	VCVTTT.F32.F16 Qd,Qm		Ī

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [arm_]vcvttq_x_f32_f16(float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
	_	VCVTTT.F32.F16 Qd,Qm		
float16x8_t [arm_]vcvtq[_f16_s16](int16x8_t a)	a -> Qm	VCVT.F16.S16 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [_arm_]vcvtq[_f16_u16](uint16x8_t a)	a -> Qm	VCVT.F16.U16 Qd,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]vcvtq[_f32_s32](int32x4_t a)	a -> Qm	VCVT.F32.S32 Qd,Qm	Qd -> 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, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, mitroxo_t a, mve_pred1o_t p)	p -> Rp	VCVTT.F16.S16 Qd,Qm		
float16x8_t [arm_]vcvtq_m[_f16_u16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, uint16x8_t a, mve_pred16_t p)	a -> Qm	VPST	(
	p -> Rp	VCVTT.F16.U16 Qd,Qm		
$float32x4_t \ [_arm_]vcvtq_m[_f32_s32](float32x4_t$	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VCVTT.F32.S32 Qd,Qm		
float32x4_t [_arm_]vcvtq_m[_f32_u32](float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
float16x8_t [arm_]vcvtq_x[_f16_u16](uint16x8_t a,	p -> Rp a -> Qm	VCVTT.F32.U32 Qd,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VPST	Qu -> resuit	IVI V IL
mve_pred10_t p)	p -> Kp	VCVTT.F16.U16 Od,Om		
float16x8_t [arm_]vcvtq_x[_f16_s16](int16x8_t a,	a -> Om	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	(
-1 - 1 <i>/</i>		VCVTT.F16.S16 Qd,Qm		
float32x4_t [arm_]vcvtq_x[_f32_s32](int32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VCVTT.F32.S32 Qd,Qm		
float32x4_t [arm_]vcvtq_x[_f32_u32](uint32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
float16-0 tf and land of f16 a1616-t16-0 to and	- > 0	VCVTT.F32.U32 Qd,Qm	0.1 >16	MATERIEON
float16x8_t [_arm_]vcvtq_n[_f16_s16](int16x8_t a, const int imm6)	a -> Qm 1 <= imm6 <=	VCVT.F16.S16 Qd,Qm,imm6	Qd -> result	MVE/NEON
int inimo)	16			
float16x8_t [arm_]vcvtq_n[_f16_u16](uint16x8_t a,	a -> Om	VCVT.F16.U16 Qd,Qm,imm6	Od -> result	MVE/NEON
const int imm6)	1 <= imm6 <=	7 C 7 TH 1010 TO QU,QIII,IIIIII	Qu'y resuit	11112112011
,	16			
float32x4_t [arm_]vcvtq_n[_f32_s32](int32x4_t a, const	a -> Qm	VCVT.F32.S32 Qd,Qm,imm6	Qd -> result	MVE/NEON
int imm6)	1 <= imm6 <=			
	32			
float32x4_t [arm_]vcvtq_n[_f32_u32](uint32x4_t a,	a -> Qm	VCVT.F32.U32 Qd,Qm,imm6	Qd -> result	MVE/NEON
const int imm6)	1 <= imm6 <= 32			
float16x8_t [arm_]vcvtq_m_n[_f16_s16](float16x8_t	inactive -> Od	VMSR P0,Rp	Od -> result	MVE
inactive, int16x8_t a, const int imm6, mve_pred16_t p)	a -> Om	VMSK FO,KP VPST	Qu -> resuit	IVI V IL
materies, merono_t as, const me minos, mve_preuro_t p	1 <= imm6 <=	VCVTT.F16.S16 Qd,Qm,imm6		
	16			
	p -> Rp			
float16x8_t [arm_]vcvtq_m_n[_f16_u16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint16x8_t a, const int imm6, mve_pred16_t p)	a -> Qm	VPST		
	1 <= imm6 <=	VCVTT.F16.U16 Qd,Qm,imm6		
	16			
floot22v4 t [omm]vovto m n [f22 o22]/floot22v4 t	p -> Rp inactive -> Qd	VMCD DO Do	Od > magnife	MVE
float32x4_t [_arm_]vcvtq_m_n[_f32_s32](float32x4_t inactive, int32x4_t a, const int imm6, mve_pred16_t p)	a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, mt32x4_t a, const int mino, mve_pred10_t p)	1 <= imm6 <=	VCVTT.F32.S32 Qd,Qm,imm6		
	32	VC V 1 1.1 32.1332 Qu,Qini,iniinio		
	p -> Rp			
float32x4_t [arm_]vcvtq_m_n[_f32_u32](float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32x4_t a, const int imm6, mve_pred16_t p)	a -> Qm	VPST	1	
	1 <= imm6 <=	VCVTT.F32.U32 Qd,Qm,imm6		
	32			
0.460.6	p -> Rp	VII (GD DO D) (T)
float16x8_t [_arm_]vcvtq_x_n[_f16_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
const int imm6, mve_pred16_t p)	1 <= imm6 <=	VPST		
	16	VCVTT.F16.S16 Qd,Qm,imm6		
float16x8_t [arm_]vcvtq_x_n[_f16_u16](uint16x8_t a,	p -> Rp a -> Om	VMSR P0,Rp	Qd -> result	MVE
const int imm6, mve_pred16_t p)	1 <= imm6 <=	VMSK PO,RP VPST	Qu -> Iesuit	IVI V IS
,,,,,,,,,	16	VCVTT.F16.U16 Qd,Qm,imm6		
	p -> Rp	2,2,3,,,,,,,,		
float32x4_t [arm_]vcvtq_x_n[_f32_s32](int32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
const int imm6, mve_pred16_t p)	1 <= imm6 <=	VPST	1	
	32	VCVTT.F32.S32 Qd,Qm,imm6	1	
	p -> Rp	1	1	1

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [_arm_]vcvtq_x_n[_f32_u32](uint32x4_t a, const int imm6, mve_pred16_t p)	a -> Qm 1 <= imm6 <= 32	VMSR P0,Rp VPST VCVTT.F32.U32 Qd,Qm,imm6	Qd -> result	MVE
1.15 0.15	p -> Rp	VOVE GLO DI COLO	0.1	MENEON
int16x8_t [_arm_]vcvtq_s16_f16(float16x8_t a) int32x4 t [_arm_]vcvtq_s32_f32(float32x4_t a)	a -> Qm a -> Om	VCVT.S16.F16 Qd,Qm VCVT.S32.F32 Qd,Qm	Qd -> result Od -> result	MVE/NEON MVE/NEON
uint16x8_t [arm_]vcvtq_u16_f16(float16x8_t a)	a -> Qm a -> Qm	VCVT.U16.F16 Qd,Qm	Qd -> result	MVE/NEON MVE/NEON
uint32x4_t [arm_]vcvtq_u32_f32(float32x4_t a)	a -> Qm	VCVT.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vcvtq_m[_s16_f16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
1 (20 A (F)) F (20 MO)((20 A ())	p -> Rp	VCVTT.S16.F16 Qd,Qm	01	MUE
int32x4_t [arm_]vcvtq_m[_s32_f32](int32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.S32.F32 Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vcvtq_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_]vcvtq_m[_u32_f32](uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm	VCVTT.U16.F16 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t [arm_]vcvtq_x_s16_f16(float16x8_t a,	p -> Rp a -> Qm	VCVTT.U32.F32 Qd,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VMSK FO,KP VPST VCVTT.S16.F16 Qd,Qm	Qu -> resuit	MIVE
int32x4_t [arm_]vcvtq_x_s32_f32(float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
	0	VCVTT.S32.F32 Qd,Qm VMSR P0,Rp	Od -> result	MVE
uint16x8_t [arm_]vcvtq_x_u16_f16(float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSK PO,KP VPST	Qu -> result	MVE
m.ve_prearot p/	p × Ap	VCVTT.U16.F16 Qd,Qm		
uint32x4_t [_arm_]vcvtq_x_u32_f32(float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t [arm_]vcvtq_n_s16_f16(float16x8_t a, const	a -> Om	VCVTT.U32.F32 Qd,Qm VCVT.S16.F16 Qd,Qm,imm6	Od -> result	MVE/NEON
int imm6)	1 <= imm6 <= 16	VC V 1.S16.F16 Qu,Qiii,iiiiiilo	Qu -> resuit	WIVE/NEON
int32x4_t [_arm_]vcvtq_n_s32_f32(float32x4_t a, const int imm6)	a -> Qm 1 <= imm6 <= 32	VCVT.S32.F32 Qd,Qm,imm6	Qd -> result	MVE/NEON
uint16x8_t [arm_]vcvtq_n_u16_f16(float16x8_t a, const int imm6)	a -> Qm 1 <= imm6 <= 16	VCVT.U16.F16 Qd,Qm,imm6	Qd -> result	MVE/NEON
uint32x4_t [arm_]vcvtq_n_u32_f32(float32x4_t a, const int imm6)	a -> Qm 1 <= imm6 <= 32	VCVT.U32.F32 Qd,Qm,imm6	Qd -> result	MVE/NEON
int16x8_t [_arm_]vcvtq_m_n[_s16_f16](int16x8_t inactive, float16x8_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm6 <= 16	VMSR P0,Rp VPST VCVTT.S16.F16 Qd,Qm,imm6	Qd -> result	MVE
int32x4_t [arm_]vcvtq_m_n[_s32_f32](int32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, float32x4_t a, const int imm6, mve_pred16_t p)	a -> Qm 1 <= imm6 <= 32 p -> Rp	VPST VCVTT.S32.F32 Qd,Qm,imm6	Qu -> result	NIVE
uint16x8_t [_arm_]vcvtq_m_n[_u16_f16](uint16x8_t inactive, float16x8_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm6 <= 16	VMSR P0,Rp VPST VCVTT.U16.F16 Qd,Qm,imm6	Qd -> result	MVE
uint32x4_t [_arm_]vcvtq_m_n[_u32_f32](uint32x4_t inactive, float32x4_t a, const int imm6, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm 1 <= imm6 <= 32 p -> Rp	VMSR P0,Rp VPST VCVTT.U32.F32 Qd,Qm,imm6	Qd -> result	MVE
int16x8_t [arm_]vcvtq_x_n_s16_f16(float16x8_t a, const int imm6, mve_pred16_t p)	a -> Qm 1 <= imm6 <= 16 p -> Rp	VMSR P0,Rp VPST VCVTT.S16.F16 Qd,Qm,imm6	Qd -> result	MVE
int32x4_t [_arm_]vcvtq_x_n_s32_f32(float32x4_t a, const int imm6, mve_pred16_t p)	a -> Qm 1 <= imm6 <= 32 p -> Rp	VMSR P0,Rp VPST VCVTT.S32.F32 Qd,Qm,imm6	Qd -> result	MVE
uint16x8_t [_arm_]vcvtq_x_n_u16_f16(float16x8_t a, const int imm6, mve_pred16_t p)	a -> Qm 1 <= imm6 <=	VMSR P0,Rp VPST	Qd -> result	MVE

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 p -> Rp	VPST VCVTT.U32.F32 Qd,Qm,imm6		
float16x8_t [arm_]vrndq[_f16](float16x8_t a)	a -> Qm	VRINTZ.F16 Qd,Qm	Od -> 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 -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
fl-122-4 + f l t f f221/fl-122-4 + iti	p -> Rp	VRINTZT.F16 Qd,Qm	Od -> result	MVE
float32x4_t [arm_]vrndq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTZT.F32 Qd,Qm	Qd -> result	MVE
float16x8_t [arm_]vrndq_x[_f16](float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
float32x4_t [arm_]vrndq_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VRINTZT.F16 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
		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, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTNT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [arm_]vrndnq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST VRINTNT.F32 Qd,Qm	Qd -> result	MVE
float16x8_t [arm_]vrndnq_x[_f16](float16x8_t a, mve_pred16_t p)	p -> Rp a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
float32x4_t [arm_]vrndnq_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VRINTNT.F16 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
mvc_pred10_t p)	p -> Kp	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, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTMT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [arm_]vrndmq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR PO,Rp VPST VRINTMT.F32 Qd,Qm	Qd -> result	MVE
float16x8_t [arm_]vrndmq_x[_f16](float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTMT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vrndmq_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
float16x8_t [arm_]vrndpq[_f16](float16x8_t a)	a -> Om	VRINTMT.F32 Qd,Qm	Qd -> result	MVE
float32x4_t [arm_]vrndpq[_f32](float32x4_t a)	a -> Qm a -> Qm	VRINTP.F16 Qd,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
float32x4_t [arm_]vrndpq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm	VRINTPT.F16 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
float16x8_t [arm_]vrndpq_x[_f16](float16x8_t a, mve_pred16_t p)	p -> Rp a -> Qm p -> Rp	VRINTPT.F32 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
float32x4_t [_arm_]vrndpq_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VRINTPT.F16 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
float16x8_t [_arm_]vrndaq[_f16](float16x8_t a)	a -> Qm	VRINTPT.F32 Qd,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, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTAT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [arm_]vrndaq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
float16x8_t [arm_]vrndaq_x[_f16](float16x8_t a, mve_pred16_t p)	p -> Rp a -> Qm p -> Rp	VRINTAT.F32 Qd,Qm VMSR P0,Rp VPST VRINTAT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [arm_]vrndaq_x[_f32](float32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTAT.F32 Qd,Qm	Qd -> result	MVE
float16x8_t [arm_]vrndxq[_f16](float16x8_t a) float32x4_t [arm_]vrndxq[_f32](float32x4_t a)	a -> Qm a -> Qm	VRINTX.F16 Qd,Qm VRINTX.F32 Qd,Qm	Qd -> result Qd -> result	MVE 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	VPST		
float32x4_t [arm_]vrndxq_m[_f32](float32x4_t	p -> Rp inactive -> Qd	VRINTXT.F16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Qu > result	III V E
	p -> Rp	VRINTXT.F32 Qd,Qm		
float16x8_t [_arm_]vrndxq_x[_f16](float16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VRINTXT.F16 Qd,Qm		
float32x4_t [arm_]vrndxq_x[_f32](float32x4_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
:	0:	VRINTXT.F32 Qd,Qm	Od -> result	MUENEON
int8x16_t [arm_]vandq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qu -> resuit	MVE/NEON
int16x8_t [arm_]vandq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vandq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vandq[_u8](uint8x16_t a, uint8x16_t	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	Qd -> result	MVE/NEON
b) uint32x4_t [arm_]vandq[_u32](uint32x4_t a, uint32x4_t	b -> Qm a -> Qn	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
b) float16x8_t [arm_]vandq[_f16](float16x8_t a,	b -> Qm a -> On	VAND Qd,Qn,Qm	Od -> result	MVE/NEON
float16x8_t b)	b -> Qm	VAND Qu,Qii,Qiii	Qu -> resuit	WIVE/NEON
float32x4_t [arm_]vandq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vandq_m[_s8](int8x16_t inactive,	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	VANDT Qd,Qn,Qm		
int16x8_t [arm_]vandq_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	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	Qu > 105un	111,2
	b -> Qm	VANDT Qd,Qn,Qm		
uint8x16_t [arm_]vandq_m[_u8](uint8x16_t inactive,	p -> Rp inactive -> Od	VMSR P0,Rp	Od -> result	MVE
uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn	VPST	Qu -> resurt	WVL
	b -> Qm	VANDT Qd,Qn,Qm		
vint16v0 t [omm lyondo m [v16](vint16v0 t inoctive	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
uint16x8_t [arm_]vandq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qu	VMSK PO,KP VPST	Qu -> resuit	NIVE
	b -> Qm	VANDT Qd,Qn,Qm		
	p -> Rp	VII (GD DO D	0.1) am
uint32x4_t [arm_]vandq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
umt52x4_t a, umt52x4_t b, mvc_preu10_t p)	b -> Qm	VANDT Qd,Qn,Qm		
	p -> Rp			
float16x8_t [_arm_]vandq_m[_f16](float16x8_t inactive,	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Om	VANDT Od,On,Om		
	p -> Rp	2,72,72		
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 -> Om	VPST VANDT Qd,Qn,Qm		
	p -> Rp	VANDT Qu,Qii,Qiii		
int8x16_t [arm_]vandq_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_]vandq_x[_s16](int16x8_t a, int16x8_t	p -> Rp a -> Qn	VANDT Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST	22 . 100011	. =
int22v4 + [arm]vanda v[a221/int22v4 + a int22-4 +	p -> Rp	VANDT Qd,Qn,Qm	Od -> result	MVE
int32x4_t [arm_]vandq_x[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qu -> resuit	MVE
-,	p -> Rp	VANDT Qd,Qn,Qm		
uint8x16_t [arm_]vandq_x[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST VANDT Qd,Qn,Qm		
uint16x8_t [arm_]vandq_x[_u16](uint16x8_t a,	p -> Rp a -> Qn	VANDI Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST		1 =
	p -> Rp	VANDT Qd,Qn,Qm		

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [arm_]vbicq_x[_f16](float16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
float32x4_t [arm_]vbicq_x[_f32](float32x4_t a,	p -> Rp a -> On	VBICT Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
float32x4_t tatmvoicq_x[_132](noat32x4_t a, float32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Qu -> result	WIVE
	p -> Rp	VBICT Qd,Qn,Qm		
int16x8_t [arm_]vbicq[_n_s16](int16x8_t a, const	a -> Qda	VBIC.I16 Qda,#imm	Qda -> result	MVE
int16_t imm)	imm in			
	AdvSIMDExpa			
int22v4 t [amm lyhica[n c221/int22v4 t a const	ndImm	VBIC.I32 Oda,#imm	Oda -> result	MVE
int32x4_t [_arm_]vbicq[_n_s32](int32x4_t a, const int32_t imm)	a -> Qda imm in	VBIC.132 Qda,#IIIIII	Qua -> resuit	MIVE
mis2_t mini)	AdvSIMDExpa			
	ndImm			
uint16x8_t [arm_]vbicq[_n_u16](uint16x8_t a, const	a -> Qda	VBIC.I16 Qda,#imm	Qda -> result	MVE
uint16_t imm)	imm in			
	AdvSIMDExpa			
uint32x4_t [arm_]vbicq[_n_u32](uint32x4_t a, const	ndImm a -> Qda	VBIC.I32 Qda,#imm	Qda -> result	MVE
uint32x4_t [atm_]voicq[_n_u32](uint32x4_t a, const uint32_t imm)	imm in	VBIC.132 Qua,#IIIIII	Qua -> resuit	MIVE
unito 2_t mini)	AdvSIMDExpa			
	ndImm			
int16x8_t [arm_]vbicq_m_n[_s16](int16x8_t a, const	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int16_t imm, mve_pred16_t p)	imm in	VPST		
	AdvSIMDExpa	VBICT.I16 Qda,#imm		
	ndImm			
int32x4 t [arm]vbicq m n[s32](int32x4 t a, const	p -> Rp a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32x4_t [atm_]voicq_m_n[_s32](int32x4_t a, const int32_t imm, mve_pred16_t p)	imm in	VMSK FO,KP VPST	Qua -> resuit	WIVE
misz_t mini, mvc_preuro_t p)	AdvSIMDExpa	VBICT.I32 Qda,#imm		
	ndImm	and the control of th		
	p -> Rp			
uint16x8_t [arm_]vbicq_m_n[_u16](uint16x8_t a, const	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint16_t imm, mve_pred16_t p)	imm in	VPST		
	AdvSIMDExpa ndImm	VBICT.I16 Qda,#imm		
	p -> Rp			
uint32x4_t [arm_]vbicq_m_n[_u32](uint32x4_t a, const	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint32_t imm, mve_pred16_t p)	imm in	VPST		
	AdvSIMDExpa	VBICT.I32 Qda,#imm		
	ndImm			
int8x16_t [arm_]vbrsrq[_n_s8](int8x16_t a, int32_t b)	p -> Rp a -> Qn	VBRSR.8 Qd,Qn,Rm	Od -> result	MVE
mtox10_t [um]voisiq[_n_so](intox10_t u, into2_t o)	b -> Rm	V BRBR. 0 Qu,Qu,Run	Qu > result	I WE
int16x8_t [arm_]vbrsrq[_n_s16](int16x8_t a, int32_t b)	a -> Qn	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 Qd,Qn,Rm	Qd -> result	MVE
b)	b -> Rm	VBKSK.8 Qu,Qii,Kiii	Qu -> resuit	WIVE
uint16x8_t [arm_]vbrsrq[_n_u16](uint16x8_t a, int32_t	a -> Qn	VBRSR.16 Qd,Qn,Rm	Qd -> result	MVE
b)	b->Rm			
uint32x4_t [arm_]vbrsrq[_n_u32](uint32x4_t a, int32_t	a -> Qn	VBRSR.32 Qd,Qn,Rm	Qd -> result	MVE
b)	b -> Rm	VPDCD 16 010 D	0.1	MATE
float16x8_t [arm_]vbrsrq[_n_f16](float16x8_t a, int32_t b)	a -> Qn b -> Rm	VBRSR.16 Qd,Qn,Rm	Qd -> result	MVE
float32x4 t [arm]vbrsrq[n f32](float32x4 t a, int32 t	a -> On	VBRSR.32 Qd,Qn,Rm	Od -> result	MVE
b)	b -> Rm	1 211211122	Qu's result	1.17.2
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	VPST		
	b -> Rm	VBRSRT.8 Qd,Qn,Rm		
int16x8_t [arm_]vbrsrq_m[_n_s16](int16x8_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t [armjvoisiq_mi_n_stoj(mt16x8_t mactive, int16x8_t a, int32_t b, mve_pred16_t p)	a -> On	VPST	Qu -> resuit	IVI V E
mitologia, mozgi o, mrogrediogi p)	b -> Rm	VBRSRT.16 Od,On,Rm		
	p -> Rp			
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	VPST		
	b -> Rm	VBRSRT.32 Qd,Qn,Rm		
	p -> Rp	VIMOR DO P		NOTE
uint8x16_t [_arm_]vbrsrq_m[_n_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
	a -> Qn	VPST	1	1
uint8x16_t a, int32_t b, mve_pred16_t p)	b -> Rm	VBRSRT.8 Qd,Qn,Rm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vbrsrq_m[_n_u16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint16x8_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VBRSRT.16 Qd,Qn,Rm		
uint32x4_t [arm_]vbrsrq_m[_n_u32](uint32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm p -> Rp	VBRSRT.32 Qd,Qn,Rm		
float16x8_t [arm_]vbrsrq_m[_n_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VBRSRT.16 Qd,Qn,Rm		
	p -> Rm	V DKSK1.10 Qu,Qii,Kiii		
float32x4_t [arm_]vbrsrq_m[_n_f32](float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VBRSRT.32 Qd,Qn,Rm		
	p -> Rp			
int8x16_t [_arm_]vbrsrq_x[_n_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VBRSRT.8 Qd,Qn,Rm		
int16x8_t [arm_]vbrsrq_x[_n_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rn	VBRSRT.16 Qd,Qn,Rm		
int32x4_t [_arm_]vbrsrq_x[_n_s32](int32x4_t a, int32_t b, mve pred16 t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Qd -> result	MVE
, -1 -1/	p -> Rm	VBRSRT.32 Qd,Qn,Rm		
uint8x16_t [_arm_]vbrsrq_x[_n_u8](uint8x16_t a, int32_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VBRSRT.8 Qd,Qn,Rm		
uint16x8_t [arm_]vbrsrq_x[_n_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VBRSRT.16 Qd,Qn,Rm		
uint32x4_t [arm_]vbrsrq_x[_n_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
float16x8_t [arm_]vbrsrq_x[_n_f16](float16x8_t a,	p -> Rp a -> Qn	VBRSRT.32 Qd,Qn,Rm VMSR P0,Rp	Qd -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
float32x4_t [arm_]vbrsrq_x[_n_f32](float32x4_t a,	p -> Rp a -> Qn	VBRSRT.16 Qd,Qn,Rm VMSR P0,Rp	Qd -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
int8x16_t [arm_]veorq[_s8](int8x16_t a, int8x16_t b)	p -> Rp a -> Qn	VBRSRT.32 Qd,Qn,Rm VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]veorq[_s16](int16x8_t a, int16x8_t b)	b -> Qm a -> Qn	VEOR Qd,Qn,Qm	Od -> result	MVE/NEON
int32x4 t [arm veorg s32 (int32x4 t a, int32x4 t b)	b -> Qm	2 . 2 . 2	`	MVE/NEON
10 10 10 10 10 10 10 10 10 10 10 10 10 1	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	
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	a -> Qn	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
b) float16x8_t [arm_]veorq[_f16](float16x8_t a,	b -> Qm a -> Qn	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t b) float32x4_t [arm_]veorq[_f32](float32x4_t a,	b -> Qm a -> Qn	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t b)	b -> Qm	VMCD DO D.:	Od -> result	MVE
int8x16_t [_arm_]veorq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qa -> result	MVE
	b -> Qm p -> Rp	VEORT Qd,Qn,Qm		
int16x8_t [arm_]veorq_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	VEORT Qd,Qn,Qm		
int32x4_t [_arm_]veorq_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	VPST VEORT Qd,Qn,Qm		
uint8x16_t [arm_]veorq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
	a -> Qn	VPST	Qu > result	
	b -> Qm p -> Rp	VEORT Qd,Qn,Qm		
uint16x8_t [arm_]veorq_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 VEORT Qd,Qn,Qm		
	p -> Qm p -> Rp	*LOKI Qu,QII,QIII		1

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 -> On	VMSR P0,Rp VPST	Qd -> result	MVE
-	b -> Qm p -> Rp	VEORT Qd,Qn,Qm		
float16x8_t [arm_]veorq_m[_f16](float16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VEORT Qd,Qn,Qm		
float32x4_t [arm_]veorq_m[_f32](float32x4_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VEORT Qd,Qn,Qm		
	p -> Rp	2 : 2 : 2		
int8x16_t [arm_]veorq_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_]veorq_x[_s16](int16x8_t a, int16x8_t	p -> Rp a -> Qn	VEORT Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST	Qu > result	W.Y.E
int32x4_t [arm_]veorq_x[_s32](int32x4_t a, int32x4_t	p -> Rp a -> Qn	VEORT Qd,Qn,Qm VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VEORT Qd,Qn,Qm		
uint8x16_t [_arm_]veorq_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	VEORT Qd,Qn,Qm		
uint16x8_t [arm_]veorq_x[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VEORT Qd,Qn,Qm		
uint32x4_t [arm_]veorq_x[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
float16x8_t [arm_]veorq_x[_f16](float16x8_t a,	p -> Rp a -> On	VEORT Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Qu => resuit	MVL
float32x4_t [arm_]veorq_x[_f32](float32x4_t a,	p -> Rp a -> On	VEORT Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VEORT Qd,Qn,Qm		
int16x8_t [_arm_]vmovlbq[_s8](int8x16_t a)	a -> Qm	VMOVLB.S8 Qd,Qm	Qd -> result	MVE
int32x4_t [arm_]vmovlbq[_s16](int16x8_t a) uint16x8_t [arm_]vmovlbq[_u8](uint8x16_t a)	a -> Qm a -> Qm	VMOVLB.S16 Qd,Qm VMOVLB.U8 Qd,Qm	Qd -> result Qd -> result	MVE MVE
uint32x4_t [_arm_]vmovlbq[_u16](uint16x8_t a)	a -> Qm	VMOVLB.U16 Qd,Qm	Qd -> result	MVE
$int16x8_t \ [_arm_] vmovlbq_m [_s8] (int16x8_t \ inactive, \\ int8x16_t \ a, \ mve_pred16_t \ p)$	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t [arm_]vmovlbq_m[_s16](int32x4_t inactive,	p -> Rp inactive -> Qd	VMOVLBT.S8 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VMOVLBT.S16 Qd,Qm		
uint16x8_t [_arm_]vmovlbq_m[_u8](uint16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VMOVLBT.U8 Qd,Qm		
uint32x4_t [arm_]vmovlbq_m[_u16](uint32x4_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VMOVLBT.U16 Qd,Qm		
int16x8_t [_arm_]vmovlbq_x[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t [_arm_lvmovlbq_x[_s16](int16x8_t a,	a -> Qm	VMOVLBT.S8 Qd,Qm VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	$p \rightarrow Rp$	VPST	Qu' y Tesun	11172
uint16x8_t [arm_]vmovlbq_x[_u8](uint8x16_t a,	a -> Qm	VMOVLBT.S16 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VMOVLBT.U8 Qd,Qm		
uint32x4_t [_arm_]vmovlbq_x[_u16](uint16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VMOVLBT.U16 Qd,Qm		
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) uint16x8_t [arm_]vmovltq[_u8](uint8x16_t a)	a -> Qm a -> Qm	VMOVLT.S16 Qd,Qm VMOVLT.U8 Qd,Qm	Qd -> result Qd -> result	MVE 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	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp inactive -> Qd	VMOVLTT.S8 Qd,Qm	Od > manula	MVE
int32x4_t [_arm_]vmovltq_m[_s16](int32x4_t inactive, int16x8_t a, mve_pred16_t p)	a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VMOVLTT.S16 Qd,Qm		

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 -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
umorro_c a, mve_preuro_c p)	p -> Rp	VMOVLTT.U8 Qd,Qm		
uint32x4_t [arm_]vmovltq_m[_u16](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
' 16 0 . F 1 1 1 F 070' 10 16 1	p -> Rp	VMOVLTT.U16 Qd,Qm	0.1 1:	MATE
int16x8_t [arm_]vmovltq_x[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
nive_pred10_t p)	p -> Kp	VMOVLTT.S8 Qd,Qm		
int32x4_t [arm_]vmovltq_x[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
		VMOVLTT.S16 Qd,Qm		
uint16x8_t [arm_]vmovltq_x[_u8](uint8x16_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
uint32x4_t [arm_]vmovltq_x[_u16](uint16x8_t a,	a -> Qm	VMOVLTT.U8 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST	Qu => result	MVE
_F	r · ··r	VMOVLTT.U16 Qd,Qm		
int8x16_t [arm_]vmovnbq[_s16](int8x16_t a, int16x8_t	a -> Qd	VMOVNB.I16 Qd,Qm	Qd -> result	MVE
b)	b -> Qm			
int16x8_t [arm_]vmovnbq[_s32](int16x8_t a, int32x4_t	a -> Qd	VMOVNB.I32 Qd,Qm	Qd -> result	MVE
b)	b -> Qm	VALOVADA NICO LO	0.1 1:	MATE
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,	a -> Qd	VMOVNB.I32 Qd,Qm	Od -> result	MVE
uint32x4_t b)	b -> Om	7110 711 <u>D.152</u> Qu,Qiii	Qu > result	M V E
int8x16_t [arm_]vmovnbq_m[_s16](int8x16_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VMOVNBT.I16 Qd,Qm		
int16x8_t [arm_]vmovnbq_m[_s32](int16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VPST VMOVNBT.I32 Qd,Qm		
uint8x16_t [arm_]vmovnbq_m[_u16](uint8x16_t a,	p -> Rp a -> Qd	VMSR P0,Rp	Od -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Qu => result	MVE
	p -> Rp	VMOVNBT.I16 Qd,Qm		
uint16x8_t [arm_]vmovnbq_m[_u32](uint16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VMOVNBT.I32 Qd,Qm		
int8x16_t [arm_]vmovntq[_s16](int8x16_t a, int16x8_t	a -> Qd	VMOVNT.I16 Qd,Qm	Qd -> result	MVE
b) int16x8_t [arm_]vmovntq[_s32](int16x8_t a, int32x4_t	b -> Qm a -> Od	VMOVNT.I32 Qd,Qm	Qd -> result	MVE
b)	b -> Qm	VWO VIVI.132 Qu,QIII	Qu -> result	WIVE
uint8x16_t [arm_]vmovntq[_u16](uint8x16_t a,	a -> Qd	VMOVNT.I16 Qd,Qm	Qd -> result	MVE
uint16x8_t b)	b -> Qm			
uint16x8_t [arm_]vmovntq[_u32](uint16x8_t a,	a -> Qd	VMOVNT.I32 Qd,Qm	Qd -> result	MVE
uint32x4_t b)	b -> Qm	VA (GD DO D	0.1) a m
int8x16_t [arm_]vmovntq_m[_s16](int8x16_t a,	a -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VMOVNTT.I16 Qd,Qm		
int16x8 t [arm]vmovntq m[s32](int16x8 t a,	a -> Qd	VMSR P0,Rp	Od -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Q	
	p -> Rp	VMOVNTT.I32 Qd,Qm		
uint8x16_t [arm_]vmovntq_m[_u16](uint8x16_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
uint16x8_t [arm_]vmovntq_m[_u32](uint16x8_t a,	p -> Rp	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	VMSR P0,Rp VPST	Qu -> resuit	IVI V E
umoza i to, mre prodrott p	p -> Rp	VMOVNTT.I32 Qd,Qm		
int8x16_t [arm_]vmvnq[_s8](int8x16_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vmvnq[_s16](int16x8_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vmvnq[_s32](int32x4_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vmvnq[_u8](uint8x16_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vmvnq[_u16](uint16x8_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vmvnq[_u32](uint32x4_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vmvnq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
miox10_t a, mvc_picu10_t p)	p -> Rp	VMVNT Qd,Qm		
int16x8_t [arm_]vmvnq_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	VMVNT Qd,Qm		
int32x4_t [arm_]vmvnq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
uint8x16_t [arm_]vmvnq_m[_u8](uint8x16_t inactive,	p -> Rp inactive -> Od	VMVNT Qd,Qm	Od > #20114	MVE
HILLIAND I AND LYMVIA ME HARMINTAX IN LINACTIVE	mactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, mve_pred16_t p)	a -> Qm	VPST		

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	VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t [_arm_]vmvnq_m[_u32](uint32x4_t inactive, uint32x4_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm p -> Rp	VMVNT Qd,Qm VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vmvnq_x[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMVNT Qu,Qm VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vmvnq_x[_s16](int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vmvnq_x[_s32](int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vmvnq_x[_u8](uint8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmvnq_x[_u16](uint16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmvnq_x[_u32](uint32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vmvnq_n_s16(const int16_t imm)	imm in AdvSIMDExpa ndImm	VMVN.I16 Qd,#imm	Qd -> result	MVE
int32x4_t [arm_]vmvnq_n_s32(const int32_t imm)	imm in AdvSIMDExpa ndImm	VMVN.I32 Qd,#imm	Qd -> result	MVE
uint16x8_t [arm_]vmvnq_n_u16(const uint16_t imm)	imm in AdvSIMDExpa ndImm	VMVN.I16 Qd,#imm	Qd -> result	MVE
uint32x4_t [arm_]vmvnq_n_u32(const uint32_t imm)	imm in AdvSIMDExpa ndImm	VMVN.I32 Qd,#imm	Qd -> result	MVE
int16x8_t [_arm_]vmvnq_m[_n_s16](int16x8_t inactive, const int16_t imm, mve_pred16_t p)	inactive -> Qd imm in AdvSIMDExpa ndImm	VMSR P0,Rp VPST VMVNT.116 Qd,#imm	Qd -> result	MVE
int32x4_t [_arm_]vmvnq_m[_n_s32](int32x4_t inactive, const int32_t imm, mve_pred16_t p)	p -> Rp inactive -> Qd imm in AdvSIMDExpa ndImm	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)	p -> Rp inactive -> Qd imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I16 Qd,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vmvnq_m[_n_u32](uint32x4_t inactive, const uint32_t imm, mve_pred16_t p)	inactive -> Qd imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I32 Qd,#imm	Qd -> result	MVE
int16x8_t [_arm_]vmvnq_x_n_s16(const int16_t imm, mve_pred16_t p)	imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I16 Qd,#imm	Qd -> result	MVE
int32x4_t [_arm_]vmvnq_x_n_s32(const int32_t imm, mve_pred16_t p)	imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I32 Qd,#imm	Qd -> result	MVE
uint16x8_t [arm_]vmvnq_x_n_u16(const uint16_t imm, mve_pred16_t p)	imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I16 Qd,#imm	Qd -> result	MVE
uint32x4_t [arm_]vmvnq_x_n_u32(const uint32_t imm, mve_pred16_t p)	imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VMVNT.I32 Qd,#imm	Qd -> result	MVE
mve_pred16_t [arm_]vpnot(mve_pred16_t a)	a -> Rp	VMSR P0,Rp VPNOT VMRS Rt,P0	Rt -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vpselq[_s8](int8x16_t a, int8x16_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Qm	VPSEL Qd,Qn,Qm		
int16x8_t [arm_]vpselq[_s16](int16x8_t a, int16x8_t b,	p -> Rp a -> On	VMSR P0.Rp	Od -> result	MVE
mrtoxo_t [aiii_]vpsciq[_s10](iii10xo_t a, iii10xo_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPSEL Qd,Qn,Qm	Qu -> resuit	WIVE
int32x4_t [arm_]vpselq[_s32](int32x4_t a, int32x4_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Qm p -> Rp	VPSEL Qd,Qn,Qm		
int64x2_t [arm_]vpselq[_s64](int64x2_t a, int64x2_t b,	a -> Qn	VMSR P0,Rp	Od -> result	MVE
mve_pred16_t p)	b -> Qm p -> Rp	VPSEL Qd,Qn,Qm		
uint8x16_t [arm_]vpselq[_u8](uint8x16_t a, uint8x16_t	a -> Qn	VMSR P0,Rp	Od -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPSEL Qd,Qn,Qm		
uint16x8_t [arm_]vpselq[_u16](uint16x8_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPSEL Qd,Qn,Qm	Qu' y 105uit	
uint32x4_t [arm_]vpselq[_u32](uint32x4_t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPSEL Qd,Qn,Qm	Qu'y Iosan	11172
uint64x2_t [arm_]vpselq[_u64](uint64x2_t a,	a -> Qn	VMSR P0,Rp	Od -> result	MVE
uint64x2_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPSEL Qd,Qn,Qm	2 7 105411	
float16x8_t [arm_]vpselq[_f16](float16x8_t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPSEL Qd,Qn,Qm	Qu' y 105uit	11172
float32x4_t [arm_]vpselq[_f32](float32x4_t a,	a -> Qn	VMSR P0,Rp	Od -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPSEL Qd,Qn,Qm	Qu > resun	141,12
	p -> Rp			
float16x8_t [arm_]vornq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [arm_]vornq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vornq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vornq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vornq[_s32](int32x4_t a, int32x4_t b)	a -> Qn	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vornq[_u8](uint8x16_t a, uint8x16_t	b -> Qm a -> Qn	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
b) uint16x8_t [arm_]vornq[_u16](uint16x8_t a, uint16x8_t	b -> Qm a -> Qn	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
b) uint32x4 t [arm]vornq[u32](uint32x4 t a, uint32x4 t	b -> Qm	VORN 04 0 · O ··	0.1 >16	MUENICON
uint32x4_t [arm_jvornq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vornq_m[_f16](float16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VORNT Qd,Qn,Qm		
float32x4_t [arm_]vornq_m[_f32](float32x4_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn	VPST	Q	1
	b -> Qm p -> Rp	VORNT Qd,Qn,Qm		
int8x16_t [arm_]vornq_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	VORNT Qd,Qn,Qm		
int16x8_t [arm_]vornq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Om	VPST		
	p -> Qm	VORNT Qd,Qn,Qm		
int32x4_t [arm_]vornq_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	VPST VORNT Qd,Qn,Qm		
	p -> Rp	1 . 1 . 1		
int8x16_t [arm_]vornq_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 VORNT Qd,Qn,Qm		
	p -> Rp	1 . 1 . 1		
uint16x8_t [_arm_]vornq_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 VORNT Qd,Qn,Qm		
	p -> Rp			

Intrinsic	Argument Preparation	Instruction	Result	Supported Architecture
uint32x4_t [arm_]vornq_m[_u32](uint32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VORNT Qd,Qn,Qm		
float16x8_t [arm_]vornq_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	Qu -> result	WYE
	p -> Rp	VORNT Qd,Qn,Qm		
float32x4_t [arm_]vornq_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	VORNT Qd,Qn,Qm		
int8x16_t [arm_]vornq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mive_pred10_t p)	p -> Rp	VORNT Qd,Qn,Qm		
int16x8_t [arm_]vornq_x[_s16](int16x8_t a, int16x8_t	a -> On	VMSR P0,Rp	Od -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		·
	p -> Rp	VORNT Qd,Qn,Qm		
nt32x4_t [arm_]vornq_x[_s32](int32x4_t a, int32x4_t	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
o, mve_pred16_t p)	b -> Qm	VPST		
sintly 16 t f name by name of v01/vintly 16 to	p -> Rp	VORNT Qd,Qn,Qm VMSR P0,Rp	Od -> result	MVE
hint8x16_t [arm_]vornq_x[_u8](uint8x16_t a, hint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSK PO,KP VPST	Qu -> resuit	IVI V E
minox10_t b, inve_pred10_t p)	p -> Rp	VORNT Od,On,Om		
uint16x8 t [arm]vornq x[u16](uint16x8 t a,	a -> On	VMSR P0,Rp	Od -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VORNT Qd,Qn,Qm		
nint32x4_t [arm_]vornq_x[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
1 (10.0)	p -> Rp	VORNT Qd,Qn,Qm	0.1	MATE
float16x8_t [arm_]vorrq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE
Toat32x4 t [arm]vorrq[f32](float32x4 t a,	a -> Qn	VORR Od,On,Om	Od -> result	MVE
float32x4_t [ami_jvoriq[_i32](noat32x4_t a,	b -> Qm	VOICE Qu,Qii,Qiii	Qu -> result	WYL
nt8x16_t [arm_]vorrq[_s8](int8x16_t a, int8x16_t b)	a -> On	VORR Qd,Qn,Qm	Od -> result	MVE/NEON
	b -> Qm	7,7,7	`	
nt16x8_t [arm_]vorrq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
	b -> Qm			
int32x4_t [arm_]vorrq[_s32](int32x4_t a, int32x4_t b)	a -> Qn	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
: .0.16 .F	b -> Qm	WORD OLO O	0.1	MENEON
uint8x16_t [arm_]vorrq[_u8](uint8x16_t a, uint8x16_t	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vorrq[_u16](uint16x8_t a, uint16x8_t	a -> Qn	VORR Qd,Qn,Qm	Od -> result	MVE/NEON
	b -> Qm	, 9141 44,41,411	Qu' > Tesair	
uint32x4_t [arm_]vorrq[_u32](uint32x4_t a, uint32x4_t	a -> Qn	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
b)	b -> Qm			
float16x8_t [arm_]vorrq_m[_f16](float16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
loat16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm p -> Rp	VORRT Qd,Qn,Qm		
loat32x4_t [arm_]vorrq_m[_f32](float32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
loat32x4_t iarm_jvortq_mi_132j(noat32x4_t mactive, loat32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn	VIVISIC 1 O, KP	Qu -> resuit	WIVE
ionio 2n i_t u, nomo 2n i_t o, mre_prouto_t p)	b -> Qm	VORRT Qd,Qn,Qm		
	p -> Rp	C-7 C-7 C		
nt8x16_t [arm_]vorrq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
nt8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VORRT Qd,Qn,Qm		
nt16v0 + f arm lyarra mf a161/int16v0 + insati	p -> Rp inactive -> Od	VMCD DO D-	Od > manule	MVE
nt16x8_t [arm_]vorrq_m[_s16](int16x8_t inactive, nt16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
mrono_t a, introvo_t o, invo_picuro_t p)	b -> Qm	VORRT Od,On,Om		
	p -> Rp	. (.,, (, (
nt32x4_t [arm_]vorrq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
nt32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VORRT Qd,Qn,Qm		
int8v16 t [arm]vores m[v9](vint9v16 timestime	p -> Rp inactive -> Qd	VMCD DO Do	04 > ====16	MVE
tint8x16_t [arm_]vorrq_m[_u8](uint8x16_t inactive, tint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qm	VORRT Qd,Qn,Qm		
	p -> Rp	C-7 C-7 C		
uint16x8_t [_arm_]vorrq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
	a -> Qn	VPST		
	b -> Qm	VORRT Qd,Qn,Qm		
	p -> Rp	VMCD DO D	01.	MY
tint32x4_t [_arm_]vorrq_m[_u32](uint32x4_t inactive,	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPS1 VORRT Qd,Qn,Qm		
		TORKI OU.UII.UIII	1	

	Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
Description				Qd -> result	MVE
More	float16x8_t b, mve_pred16_t p)	~			
	float32x4 t [arm]vorrq x[f32](float32x4 t a,			Od -> result	MVE
initials_i_l_mm_hormn_s_i_s8imsx16_i_s initisx16_i_s in so on p > kg vortice	float32x4_t b, mve_pred16_t p)				
p > R VORRT QLQn,Qm				Qd -> result	MVE
mid 168.8_1_mm_ vorq_A_sl_sl6 mid 68_1 a, ind 168_2, b a > 0	mve_pred16_t p)				
mvc_perd(c_i_p)	int16x8 t [arm]vorrq x[s16](int16x8 t a, int16x8 t b,			Od -> result	MVE
mid234_11_mm_lyorq_N_s32[mid224_1 a, mid224_1 b]	mve_pred16_t p)				
b > Qm					
Description				Qd -> result	MVE
	mve_pred16_t p)				
wind16x8_t[_arm_ vorrq_n_nf_t]16 (wint16x8_t a, wind16x8_t t, arm_ vorrq_n_nf_t]16 (wint16x8_t t, arm_)vorrq_n_nf_t]16 (wint16x8_t t, ar	uint8x16 t [arm]vorrg x[u8](uint8x16 t a			Od -> result	MVF.
Description				Qu > result	111 1 1
UPST		-	VORRT Qd,Qn,Qm		
Dec VORRT Qd,Qn,Qm				Qd -> result	MVE
und124_4_L_amm_lyornq_x_u322[uind32x4_t a, ub. p. > 0m VMSR PORp Qd > result MVE int1688_t_L_amm_lyornq_n_s32[ind32x4_t a, const int6_t imm) a > 0m VORR_116 Qda_#imm Qda > result MVE int1688_t_L_amm_lyornq_n_s32[ind32x4_t a, const int6_t imm) a > 0d VORR_116 Qda_#imm Qda > result MVE uint16x8_t_L_amm_lyornq_n_u16[idint16x8_t a, const unt16_t imm) a > 0d VORR_132 Qda_#imm Qda > result MVE uint12x4_t_L_amm_lyornq_n_u16[idint16x8_t a, const unt16_t imm) a > 0da VORR_132 Qda_#imm Qda > result MVE uint2x4_t_L_amm_lyornq_m_n[s16](int16x8_t a, const unt16x8_t_L_amm_lyornq_m_n[s16](int16x8_t a, const unt16x8_t_L_amm_lyornq_m_n[s16](int16x8_t a, const unt16x8_t_L_amm_lyornq_m_n[s24](int16x8_t a, const unt16_t_imm, nwe_pred16_t_p) WSSR_PORP Qda > result MVE uint16x8_t_L_amm_lyornq_m_n[s24](int16x8_t a, const unt16_t_imm, nwe_pred16_t_p) a > 0da WSSR_PORP Qda > result MVE uint16x8_t_L_amm_lyornq_m_n[s24](int16x8_t a, const unt16_t_imm, nwe_pred16_t_p) a > 0da WSR_PORP Qda > result MVE uint22x4_t_L_amm_lyornq_m_n[s24](int16x8_t a, const unt16_t_imm, nwe_pred16_t_p) a > 0da WSR_PORP Qda > result MVE uint16x8_t_L_amm_	uint16x8_t b, mve_pred16_t p)		1 12		
uint128.k_t_arm_lyorrq[_n_s16](int16x8_t a, const int16x_t_imm) b > 0m	vint22v4 t [Od > magnit	MVE
P > Rp				Qu -> resuit	MVE
	umeszki-t o, mvc_prouto_t py		1 12		
AdvSIMDExpa ndmm	int16x8_t [arm_]vorrq[_n_s16](int16x8_t a, const			Qda -> result	MVE
ndlmm	int16_t imm)		-		
a					
Imm in AdvSIMDExpa ndimm Part Part	:		VODB 122 O.1- #:	0.4- >14	MVE
AdvSIMDExpa			VORR.132 Qda,#imm	Qda -> result	MVE
mit16x8_t[_arm_ vorrq_n_u16](uint16x8_t a, const uint16_t imm)	int32_t inini)				
AdvSIMDExpa	uint16x8_t [arm_]vorrq[_n_u16](uint16x8_t a, const	a -> Qda	VORR.I16 Qda,#imm	Qda -> result	MVE
Intliance	uint16_t imm)				
a					
imm in AdvSIMDExpa AdvSI	uint32v4 t [arm]vorra[n u32](uint32v4 t a const		VOPP 132 Oda #imm	Oda > regult	MVE
AdvSIMDExpa		`	VOICIC.132 Qua,#IIIIII	Qua -> resurt	WYE
Init		AdvSIMDExpa			
imm in					
AdvSIMDExpa ndlmm p > Rp				Qda -> result	MVE
int32x4_t[_arm_]vorrq_m_n[_s32](int16x8_t a, const int32_t imm, mve_pred16_t p)	int16_t imm, mve_pred16_t p)				
Description			VORK1:110 Qua;#IIIIII		
imm in AdvSIMDExpa ndlmm p > Rp					
AdvSIMDExpa ndlmm	int32x4_t [arm_]vorrq_m_n[_s32](int32x4_t a, const	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
Intition Intition	int32_t imm, mve_pred16_t p)				
D -> Rp			VORRT.I32 Qda,#imm		
uint16x8_t [_arm_]vorrq_mn[_u16](uint16x8_t a, const uint16_t imm, mve_pred16_t p) a > Qda imm in AdvSIMDExpa ndImm p > Rp VPST VORRT.116 Qda,#imm Qda > result MVE uint32x4_t [_arm_]vorrq_mn[_u32](uint32x4_t a, const uint32_t imm, mve_pred16_t p) a > Qda imm in AdvSIMDExpa ndImm p > Rp VMSR P0,Rp Qda > result MVE uint32x1_imm, mve_pred16_t p) a > Qda imm in AdvSIMDExpa ndImm p > Rp VPST VORRT.132 Qda,#imm MVE uint8x16_t [_arm_]vqmovnbq[_s16](int8x16_t a, int16x8_t t] a > Qd VQMOVNB.S16 Qd,Qm Qd > result MVE uint8x16_t [_arm_]vqmovnbq[_s32](int16x8_t a, int16x8_t b) a > Qd VQMOVNB.S32 Qd,Qm Qd > result MVE uint16x8_t b) b > Qm VQMOVNB.U16 Qd,Qm Qd > result MVE uint16x8_t b) a > Qd VQMOVNB.U16 Qd,Qm Qd > result MVE uint16x8_t [_arm_]vqmovnbq_m[_s16](int8x16_t a, int8x16_t a, int8x16_t [_arm_]vqmovnbq_m[_s16](int8x16_t a, a > Qd VQMOVNBT.S16 Qd,Qm Qd > result MVE uint16x8_t [_arm_]vqmovnbq_m[_s32](int16x8_t a, a > Qd VQMOVNBT.S32 Qd,Qm Qd > result MVE uint16x8_t [_arm_]vqmovnbq_m[_s32](int16x8_t a, a > Qd VQMOVNBT.S32 Qd,Qm Qd > result MVE uint16x8_t [_arm_]vqmovnbq_m[_s32](int16x8_t a, a > Qd VQMOVNBT.S32 Qd,Qm Qd > result <					
uint16_t imm, mve_pred16_t p) imm in AdvSIMDExpa ndImm p -> Rp VPST VORRT.116 Qda,#imm WVE uint32x4_t tarmvorrq_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 Qda -> result WVE int8x16_t [_armvormovnbq[_s16](int8x16_t a, int32x4_t b) a -> Qd VQMOVNB.S16 Qd,Qm Qd -> result MVE int16x8_t [_armvormovnbq[_s32](int16x8_t a, int32x4_t b) b -> Qm VQMOVNB.S32 Qd,Qm Qd -> result MVE uint16x8_t [_armvormovnbq[_u16](uint8x16_t a, int32x4_t b) b -> Qm VQMOVNB.U16 Qd,Qm Qd -> result MVE uint16x8_t [_armvormovnbq[_u32](uint16x8_t a, int32x4_t b) b -> Qm VQMOVNB.U32 Qd,Qm Qd -> result MVE uint16x8_t [_armvormovnbqu32](uint16x8_t a, int32x4_t b) b -> Qm VQMOVNB.U32 Qd,Qm Qd -> result MVE int8x16_t [_armvormovnbq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp Qd -> result MVE int16x8_t [_armvormovnbq_m[_s32](int16x8_t a, int16x8_t a, int16x8_t b, mve_pred16_t p) b -> Qm VPST uint8x16_t [_armvormovnbq_m[_s32](int16x8_t a, int16x8_t b, mve_pred16_t p) b -> Qm VPST uint8x16_t [_armvormovnbq_m[_s16](uint8x16_t a, int16x8_t b, mve_pred16_t p) b -> Qm VPST	uint16x8 t [arm]vorra m n[u16](uint16x8 t a. const		VMSR P0.Rp	Oda -> result	MVE
Indimm P -> Rp			*****	Q	
D -> Rp			VORRT.I16 Qda,#imm		
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 Qda -> result MVE int8x16_t [_arm_]vqmovnbq[_s16](int8x16_t a, int16x8_t b) a -> Qd b -> Qd VQMOVNB.S16 Qd,Qm Qd -> result MVE int16x8_t b) b -> Qm VQMOVNB.S32 Qd,Qm Qd -> result MVE int16x8_t b) b -> Qm VQMOVNB.S32 Qd,Qm Qd -> result MVE uint8x16_t farm_]vqmovnbq[_u16](uint8x16_t a, uint16x8_t b) a -> Qd VQMOVNB.U16 Qd,Qm Qd -> result MVE uint16x8_t tarm_]vqmovnbq_[u32](uint16x8_t a, uint2x4_t b) a -> Qd VQMOVNB.U32 Qd,Qm Qd -> result MVE uint8x16_t farm_]vqmovnbq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp Qd -> result MVE uint8x16_t [_arm_]vqmovnbq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp Qd -> result MVE uint8x16_t [_arm_]vqmovnbq_m[_s16](uint8x16_t a, int32x4_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp Qd -> result MVE uint8x16_t [_arm_]vqmovnbq_m[_s16](uint8x16_t a, int32x4_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp Qd -> result MVE					
uint32_t imm, mve_pred16_t p) imm in AdvSIMDExpa ndImm p > Rp VPST VORRT.I32 Qda,#imm VPST VORRT.I32 Qda,#imm int8x16_t [_arm_]vqmovnbq[_s16](int8x16_t a, int16x8_t b) a > Qd VQMOVNB.S16 Qd,Qm Qd > result MVE int16x8_t [_arm_]vqmovnbq[_s32](int16x8_t a, int32x4_t b) a > Qd VQMOVNB.S32 Qd,Qm Qd > result MVE uint8x16_t [_arm_]vqmovnbq[_u16](uint8x16_t a, int16x8_t b) a > Qd VQMOVNB.U16 Qd,Qm Qd >> result MVE uint16x8_t b) b > Qm VQMOVNB.U32 Qd,Qm Qd -> result MVE uint16x8_t b) b > Qm VQMOVNB.U32 Qd,Qm Qd -> result MVE uint32x4_t b) b > Qm VPST VPST uint8x16_t [_arm_]vqmovnbq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p) b > Qm VPST Qd -> result MVE vMSR P0,Rp VQMOVNBT.S16 Qd,Qm VMSR P0,Rp Qd -> result MVE uint8x16_t [_arm_]vqmovnbq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp Qd -> result MVE uint8x16_t [_arm_]vqmovnbq_m[_s16](uint8x16_t a, int32x4_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp Qd -> result MVE	nint32v4 t [arm]vorra m n[n22](nint22v4 t a const		VMSR P0 Pn	Oda > rocult	MVF
AdvSIMDExpa ndImm p -> Rp				Qua -> resuit	IVI V E
p -> Rp	amez_t mm, m·e_predio_t p)				
int8x16_t [_arm_]vqmovnbq[_s16](int8x16_t a, int16x8_t b) a -> Qd b -> Qd VQMOVNB.S16 Qd,Qm Qd -> result MVE int16x8_t b) a -> Qd b -> Qd VQMOVNB.S32 Qd,Qm Qd -> result MVE int32x4_t b) b -> Qm VQMOVNB.S32 Qd,Qm Qd -> result MVE uint8x16_t [_arm_]vqmovnbq[_u16](uint8x16_t a, uint16x8_t b) a -> Qd b -> Qd VQMOVNB.U16 Qd,Qm Qd -> result MVE uint16x8_t b b -> Qm VQMOVNB.U32 Qd,Qm Qd -> result MVE uint2x4_t b) b -> Qm VMSR P0,Rp Qd -> result MVE int8x16_t [_arm_]vqmovnbq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p) b -> Qm VMSR P0,Rp Qd -> result MVE int16x8_t [_arm_]vqmovnbq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp VMSR P0,Rp Qd -> result MVE uint8x16_t [_arm_]vqmovnbq_m[_u16](uint8x16_t a, uint8x16_t a, b, owe_pred16_t p) a -> Qd VMSR P0,Rp VMSR P0,Rp Qd -> result MVE		ndImm	- /		
int16x8_t b			YOU OVER THE STATE OF THE STATE) are
int16x8_t [_arm_]vqmovnbq[_s32](int16x8_t a, int32x4_t b) a -> Qd b -> Qd VQMOVNB.S32 Qd,Qm Qd -> result MVE uint8x16_t [_arm_]vqmovnbq[_u16](uint8x16_t a, uint16x8_t b) a -> Qd b -> Qd VQMOVNB.U16 Qd,Qm Qd -> result MVE uint16x8_t [_arm_]vqmovnbq[_u32](uint16x8_t a, uint32x4_t b) b -> Qm VQMOVNB.U32 Qd,Qm Qd -> result MVE uint32x4_t b) b -> Qm VMSR P0,Rp Qd -> result MVE uint8x16_t [_arm_]vqmovnbq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp Qd -> result MVE uint32x4_t b, mve_pred16_t p) b -> Qm VPST VQMOVNBT.S16 Qd,Qm VPST uint8x16_t [_arm_]vqmovnbq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p) b -> Qm VPST VQMOVNBT.S32 Qd,Qm uint8x16_t [_arm_]vqmovnbq_m[_u16](uint8x16_t a, int16x8_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp Qd -> result MVE		~	VQMOVNB.S16 Qd,Qm	Qd -> result	MVE
b -> Qm			VOMOVNR \$32 Od Om	Od -> result	MVF
uint8x16_t [_arm_]vqmovnbq[_u16](uint8x16_t a, a -> Qd VQMOVNB.U16 Qd,Qm Qd -> result MVE uint16x8_t b) b -> Qm VQMOVNB.U32 Qd,Qm Qd -> result MVE uint16x8_t t [_arm_]vqmovnbq[_u32](uint16x8_t a, a -> Qd VQMOVNB.U32 Qd,Qm Qd -> result MVE uint32x4_t b) b -> Qm VMSR P0,Rp Qd -> result MVE int16x8_t b, mve_pred16_t p) b -> Qm VPST VQMOVNBT.S16 Qd,Qm int16x8_t [_arm_]vqmovnbq_m[_s32](int16x8_t a, a -> Qd VMSR P0,Rp Qd -> result MVE int32x4_t b, mve_pred16_t p) b -> Qm VPST VQMOVNBT.S32 Qd,Qm uint8x16_t [_arm_]vqmovnbq_m[_u16](uint8x16_t a, a -> Qd VMSR P0,Rp Qd -> result MVE uint16x8_t b, mve_pred16_t p) b -> Qm VMSR P0,Rp Qd -> result MVE		`	, 2,110 (11b.552 Qu,Qiii	Qu -> resuit	141 4 12
uint16x8_t b) b -> Qm uint16x8_t tarmlvqmovnbq[_u32](uint16x8_t a, uint32x4_t b) a -> Qd b -> Qm VQMOVNB.U32 Qd,Qm Qd -> result MVE int8x16_t [_armlvqmovnbq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p) a -> Qd b -> Qm VMSR P0,Rp Qd -> result MVE int16x8_t [_armlvqmovnbq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p) a -> Qd b -> Qm VMSR P0,Rp Qd -> result MVE uint8x16_t [_armlvqmovnbq_m[_u16](uint8x16_t a, uint16x8_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp Qd -> result MVE uint8x16_t [_armlvqmovnbq_m[_u16](uint8x16_t a, uint16x8_t b, mve_pred16_t p) a -> Qd VMSR P0,Rp Qd -> result MVE			VQMOVNB.U16 Qd,Qm	Qd -> result	MVE
uint32x4_t b) b -> Qm int8x16_t [_arm_]vqmovnbq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p) a -> Qd					
int8x16_t [_arm_]vqmovnbq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p)			VQMOVNB.U32 Qd,Qm	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)			VMCD DO D-	04 > =====14	MVE
p -> Rp VQMOVNBT.S16 Qd,Qm		`		Qu -> result	MVE
$ \begin{array}{llllllllllllllllllllllllllllllllllll$	introde_t o, into_prodio_t p)	-			
p -> Rp VQMOVNBT.S32 Qd,Qm uint8x16_t [_arm_]vqmovnbq_m[_u16](uint8x16_t a, a -> Qd VMSR P0,Rp Qd -> result MVE uint16x8_t b, mve_pred16_t p) b -> Qm VPST MVE	int16x8_t [arm_]vqmovnbq_m[_s32](int16x8_t a,			Qd -> result	MVE
$ \begin{array}{llllllllllllllllllllllllllllllllllll$		b -> Qm	VPST		
$ uint16x8_t \ b, mve_pred16_t \ p) \qquad \qquad b \rightarrow Qm \qquad VPST $					1.07.00
- / -I - I/				Qd -> result	MVE
I n -> Kn	umitoxo_t v, mve_predio_t p)	b -> Qm p -> Rp	VQMOVNBT.U16 Qd,Qm		

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 -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
umis2x1_t s, inve_pred16_t p)	p -> Rp	VQMOVNBT.U32 Qd,Qm		
int8x16_t [arm_]vqmovntq[_s16](int8x16_t a, int16x8_t b)	a -> Qd b -> Qm	VQMOVNT.S16 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vqmovntq[_s32](int16x8_t a, int32x4_t b)	a -> Qd b -> Qm	VQMOVNT.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [arm_]vqmovntq[_u16](uint8x16_t a,	a -> Qd	VQMOVNT.U16 Qd,Qm	Qd -> result	MVE
uint16x8_t b) uint16x8_t [_arm_]vqmovntq[_u32](uint16x8_t a,	b -> Qm a -> Qd	VQMOVNT.U32 Qd,Qm	Qd -> result	MVE
uint32x4_t b) int8x16_t [arm_]vqmovntq_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 VQMOVNTT.S16 Qd,Qm		
int16x8_t [_arm_]vqmovntq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
_ _ _ _ _	p -> Rp	VQMOVNTT.S32 Qd,Qm	0.1) OF
uint8x16_t [arm_]vqmovntq_m[_u16](uint8x16_t a, uint16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
: 16 0 · F	p -> Rp	VQMOVNTT.U16 Qd,Qm	0.1	MATE
uint16x8_t [_arm_]vqmovntq_m[_u32](uint16x8_t a, uint32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
	p -> Rp	VQMOVNTT.U32 Qd,Qm VOMOVUNB.S16 Od.Om	0.1 >16	MVE
uint8x16_t [arm_]vqmovunbq[_s16](uint8x16_t a, int16x8_t b)	a -> Qd b -> Qm		Qd -> result	MVE
uint16x8_t [arm_]vqmovunbq[_s32](uint16x8_t a, int32x4_t b)	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	VMSR P0,Rp VPST	Qd -> result	MVE
_ ,	p -> Rp	VQMOVUNBT.S16 Qd,Qm		
uint16x8_t [_arm_]vqmovunbq_m[_s32](uint16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
- 1	p -> Rp	VQMOVUNBT.S32 Qd,Qm		
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,	a -> Od	VMSR P0,Rp	Od -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VQMOVUNTT.S16 Qd,Qm		
uint16x8_t [arm_]vqmovuntq_m[_s32](uint16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VQMOVUNTT.S32 Qd,Qm		
int8x16_t [arm_]vqrshlq[_n_s8](int8x16_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [arm_]vqrshlq[_n_s16](int16x8_t a, int32_t b)	a -> Qda	VQRSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vqrshlq[_n_s32](int32x4_t a, int32_t b)	b -> Rm a -> Qda	VQRSHL.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [arm_]vqrshlq[_n_u8](uint8x16_t a, int32_t	b -> Rm a -> Qda	VQRSHL.U8 Qda,Rm	Qda -> result	MVE
b) uint16x8_t [arm_]vqrshlq[_n_u16](uint16x8_t a, int32_t	b -> Rm a -> Qda	VQRSHL.U16 Qda,Rm	Qda -> result	MVE
b) uint32x4_t [_arm_]vqrshlq[_n_u32](uint32x4_t a, int32_t	b -> Rm a -> Qda	VQRSHL.U32 Qda,Rm	Qda -> result	MVE
b)	b->Rm			
int8x16_t [arm_]vqrshlq_m_n[_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
	p -> Rp	VQRSHLT.S8 Qda,Rm		
int16x8_t [arm_]vqrshlq_m_n[_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
	p -> Rp	VQRSHLT.S16 Qda,Rm		
int32x4_t [_arm_]vqrshlq_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 -> Rp	VQRSHLT.S32 Qda,Rm		No.
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
	p -> Rp	VQRSHLT.U8 Qda,Rm) am
uint16x8_t [arm_]vqrshlq_m_n[_u16](uint16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
	p -> Rp	VQRSHLT.U16 Qda,Rm	<u> </u>	3.67.50
uint32x4_t [arm_]vqrshlq_m_n[_u32](uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
1.0 16.1	p -> Rp	VQRSHLT.U32 Qda,Rm	01 :	Marian
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 -> Qn	VQRSHL.U32 Qd,Qm,Qn	Qd -> result	MVE/NEON
int8x16_t [_arm_]vqrshlq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.S8 Qd,Qm,Qn	Qd -> result	MVE
int16x8_t [_arm_]vqrshlq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.S16 Qd,Qm,Qn	Qd -> result	MVE
int32x4_t [_arm_]vqrshlq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.S32 Qd,Qm,Qn	Qd -> result	MVE
uint8x16_t [_arm_]vqrshlq_m[_u8](uint8x16_t inactive, uint8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.U8 Qd,Qm,Qn	Qd -> result	MVE
uint16x8_t [_arm_]vqrshlq_m[_u16](uint16x8_t inactive, uint16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.U16 Qd,Qm,Qn	Qd -> result	MVE
uint32x4_t [_arm_]vqrshlq_m[_u32](uint32x4_t inactive, uint32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VQRSHLT.U32 Qd,Qm,Qn	Qd -> result	MVE
int8x16_t [arm_]vqrshrnbq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQRSHRNB.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vqrshrnbq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQRSHRNB.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqrshrnbq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQRSHRNB.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqrshrnbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQRSHRNB.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vqrshrnbq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQRSHRNBT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vqrshrnbq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQRSHRNBT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqrshrnbq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQRSHRNBT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vqrshrnbq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQRSHRNBT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vqrshrntq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQRSHRNT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vqrshrntq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQRSHRNT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [arm_]vqrshrntq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQRSHRNT.U16 Qd,Qm,#imm	Qd -> result	MVE

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

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [_arm_]vqshlq_m_r[_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
o, mve_pred1o_t p)	p -> Rn	VOSHLT.S16 Oda,Rm		
int32x4_t [arm_]vqshlq_m_r[_s32](int32x4_t a, int32_t	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VQSHLT.S32 Qda,Rm		
uint8x16_t [_arm_]vqshlq_m_r[_u8](uint8x16_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST 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	Q	1
	p -> Rp	VQSHLT.U16 Qda,Rm		
uint32x4_t [arm_]vqshlq_m_r[_u32](uint32x4_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
uint8x16_t [arm_]vqshluq[_n_s8](int8x16_t a, const int	p -> Rp a -> Qm	VQSHLT.U32 Qda,Rm VQSHLU.S8 Qd,Qm,#imm	Qd -> result	MVE
imm)	$0 \le \text{imm} \le 7$	VQSHLU.S8 Qd,Qiii,#iiiiiii	Qu -> resuit	MIVE
uint16x8_t [arm_]vqshluq[_n_s16](int16x8_t a, const	a -> Om	VQSHLU.S16 Qd,Qm,#imm	Qd -> result	MVE
int imm)	0 <= imm <=			
	15			
uint32x4_t [arm_]vqshluq[_n_s32](int32x4_t a, const	a -> Qm	VQSHLU.S32 Qd,Qm,#imm	Qd -> result	MVE
int imm)	0 <= imm <=			
vint0v16 t [ome lyochlyo m[n o0](vint0v16 t	31 inactive -> Od	VMCD DO Do	Qd -> result	MVE
uint8x16_t [arm_]vqshluq_m[_n_s8](uint8x16_t inactive, int8x16_t a, const int imm, mve_pred16_t p)	a -> Om	VMSR P0,Rp VPST	Qu -> result	MVE
mactive, mitox10_t a, const mt mini, mve_pred10_t p)	$0 \le imm \le 7$	VOSHLUT.S8 Od,Om,#imm		
	p -> Rp	. (2000000000000000000000000000000000000		
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 -> Qm	VPST		
	0 <= imm <=	VQSHLUT.S16 Qd,Qm,#imm		
	15			
uint32x4_t [arm_]vqshluq_m[_n_s32](uint32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, int32x4_t a, const int imm, mve_pred16_t p)	a -> Om	VMSK PO,RP VPST	Qu -> resuit	NIVE
mactive, mt32x4_t a, const mt mini, mve_pred10_t p)	0 <= imm <=	VQSHLUT.S32 Qd,Qm,#imm		
	31			
	p -> Rp			
int8x16_t [arm_]vqshrnbq[_n_s16](int8x16_t a,	a -> Qd	VQSHRNB.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t b, const int imm)	b -> Qm			
int16x8_t [arm_]vqshrnbq[_n_s32](int16x8_t a,	1 <= imm <= 8 a -> Qd	VQSHRNB.S32 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t b, const int imm)	a -> Qu b -> Om	VQSHKNB.532 Qd,Qlii,#Illilli	Qu -> resuit	MIVE
into 2x = t o, const int inimi	1 <= imm <=			
	16			
uint8x16_t [arm_]vqshrnbq[_n_u16](uint8x16_t a,	a -> Qd	VQSHRNB.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t b, const int imm)	b -> Qm			
: 16.0 . 1 1 1 1 201/: 16.0 .	1 <= imm <= 8	MOCHENIA MISS OF O III.	0.1 1:	MATE
uint16x8_t [arm_]vqshrnbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm	VQSHRNB.U32 Qd,Qm,#imm	Qd -> result	MVE
unit32x4_t b, const int inimi)	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		
1.15.0.15.0.1	p -> Rp	VII (OD DO D	0.1	\ am
int16x8_t [_arm_]vqshrnbq_m[_n_s32](int16x8_t a,	a -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t b, const int imm, mve_pred16_t p)	b -> Qm 1 <= imm <=	VOSHRNBT.S32 Od,Om,#imm		
	16	V QBITTA VB 1.832 Qu,Qini,#Mini		
	p -> Rp		<u> </u>	<u></u>
uint8x16_t [arm_]vqshrnbq_m[_n_u16](uint8x16_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, const int imm, mve_pred16_t p)	b -> Qm	VPST		
	1 <= imm <= 8	VQSHRNBT.U16 Qd,Qm,#imm		1
uint16x8 t[arm]vqshrnbq m[n u32](uint16x8 ta,	p -> Rp a -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t [arm_jvqsnrnbq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm	VMSK PO,KP VPST	Qu -> resuit	IVIVE
	1 <= imm <=	VQSHRNBT.U32 Qd,Qm,#imm		
	16			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			1
	1 <= imm <= 8	MOCHENIT COO O 1 O ""	Qd -> result	MVE
int16m0 t []	0 > 0.1			
int16x8_t [arm_]vqshrntq[_n_s32](int16x8_t a,	a -> Qd b -> Om	VQSHRNT.S32 Qd,Qm,#imm	Qu -> resuit	IVI V E
int16x8_t [arm_]vqshrntq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <=	VQSHRN1.S32 Qd,Qm,#imm	Qu -> resuit	WIVE

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[_x8](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	Qd -> result	MVE/NEON
uint8x16_t [arm_]vrev32q[_u8](uint8x16_t a)	a -> Qm	VREV32.8 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vrev32q[_u16](uint16x8_t a)	a -> Qm	VREV32.16 Qd,Qm	Qd -> result	MVE/NEON

VREV32T.16 Qd.Qm	Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mistrice_i_mm_jvev22q_ml_sisjointioss_imacrive_i_mistrice_i_mist	int8x16 t[arm]vrev32q m[s8](int8x16 t inactive.	inactive -> Od	VMSR P0.Rp	Od -> result	MVE
				Qu' > 105air	,
ministrict min					
	int16x8 t [arm]vrev32q m[s16](int16x8 t inactive.			Od -> result	MVE
p>Rp VRFV371.16 QdQm MVE MVE				Ç	
minds		-			
	uint8x16 t [arm lyrey32g m[u8](uint8x16 t inactive.			Od -> result	MVE
p > Rp				Qu's resun	
initionset amlvers/32q_ml_sfo[(loutloss_1 a)					
nactive, mintlos 8, 1 _amm_lvrev3q_ml_fl6j(float16s8, 1 as_0m very 3q_ml_fl6j(float16s8, 1 as_0m very 4q_ml_fl6j(float16s8,	uint16x8 t [arm]vrev32q m[u16](uint16x8 t			Od -> result	MVE
Double St. am vrev32q_ml_f16(float16x8_t) a > 0m	_ 1	-			
	macu ve, umeronoge u, mve_preuroge p				
a=\times\text{minimax} a=\times\text{minimax} a=\times\text{minimax} a=\times\text{minimax} a=\times\text{minimax} a=\times\text{minimax} a>\times\text{minimax} a>\timesminim	float16x8 t [arm]vrev32a m[f16](float16x8 t			Od -> result	MVE
D - PE				Qu > resurt	11112
misk16_1_mm_ vrev32q_xs _s8 (imisk16_1 a, p > 0m vrev8q_me_pred16_t.p)	mactive, moatroxo_t a, mve_predio_t p)				
mids8_tamlyrev32q_xs16[dint16x8_t a, a	int0v16 + [orm]vrov22a v[c0](int0v16 + a			Od > rocult	MVE
MID MID				Qu -> resuit	IVI V E
MVE	mve_pred1o_t p)	p -> Kp			
P > Rp	:-+16-9 + F	0		0.1 >16	MVE
		-		Qd -> result	MVE
MASE PORP Qd > result MVE Port Qd > result MVE Port Po	mve_pred16_t p)	p -> Rp			
P					
VREV32T.16 Qd.Qm	uint8x16_t [arm_]vrev32q_x[_u8](uint8x16_t a,	a -> Qm		Qd -> result	MVE
	mve_pred16_t p)	p -> Rp	VPST		
Deat168.8_1			VREV32T.8 Qd,Qm		
Deat168.8_1	uint16x8 t [arm]vrev32q x[u16](uint16x8 t a,	a -> Om		Od -> result	MVE
New York New York					
		1			
p > Rp	float16v8 t [arm]vrev32a v[f16](float16v8 t a	a -> Om		Od -> result	MVE
NRS16_t_t_arm_ vrev64q_ssl(int8x16_t_a)				Qu => resuit	IVI V L
m8x16 t [_arm_ vrev64q[_s8](imt8x16_t a) a > Qm VREV64.8 Qd.Qm Qd > result MVENEON m16s8_t [_arm_ vrev64q[_s16](int16x8_t a) a > Qm VREV64.32 Qd.Qm Qd > result MVENEON m16s8_t [_arm_ vrev64q[_s16](int16x8_t a) a > Qm VREV64.32 Qd.Qm Qd > result MVENEON m16x1_t [_arm_ vrev64q[_s16](int16x8_t a) a > Qm VREV64.32 Qd.Qm Qd > result MVENEON m16x1_t [_arm_ vrev64q[_s16](int16x8_t a) a > Qm VREV64.32 Qd.Qm Qd > result MVENEON m16x1_t [_arm_ vrev64q[_s16](int16x8_t a) a > Qm VREV64.32 Qd.Qm Qd > result MVENEON m16x1_t [_arm_ vrev64q_m[_s16](int16x8_t a) a > Qm VREV64.32 Qd.Qm Qd > result MVENEON m16x1_t [_arm_ vrev64q_m[_s8](int16x8_t a) a > Qm VREV64.32 Qd.Qm Qd > result MVENEON m16x8_t [_arm_ vrev64q_m[_s8](int16x8_t i) a > Qm VREV64.32 Qd.Qm Qd > result MVENEON m16x8_t [_arm_ vrev64q_m[_s8](int16x8_t i) a > Qm VREV64.32 Qd.Qm Qd > result MVE m16x8_t [_arm_ vrev64q_m[_s8](int16x8_t i) a > Qm VRSR PQRp Qd > result<	mve_pred1o_t p)	p -> Kp			
MTEARS arm vrev64q s1g (int16x8,t a) a > Qm VREV64.16 Qd,Qm Qd > result MVENEON	' +0 16 +F 1	. 0		0.1 1:	MUENIEON
m1324_t_1		1	` ' `		
a > Qm		a -> Qm	VREV64.32 Qd,Qm		MVE/NEON
a > Qm	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 -> Om	VREV64.32 Qd,Qm	Od -> result	MVE/NEON
mt8x16_t[_arm_]vrev64q_m[_s8](int8x16_t inactive, nt8x16_t a, nwe_pred16_t p) inactive > Qd a > Qm VPST VPST VPST VPST VPST VPST VPST VPST					
A					_
P > Rp				Qu -> resuit	IVI V E
Interversed_a_m[_s16](int16x8_t inactive, nt16x8_t a, mve_pred16_t p)	intox10_t a, nive_pred10_t p)	-			
	: 16 0 · F			01	MATE
p -> Rp VREV64T.16 Qd,Qm				Qd -> result	MVE
mt32x4_t [_arm_]vrev64q_m[_s32](int32x4_t inactive, nt32x4_t a, mve_pred16_t p)	int16x8_t a, mve_pred16_t p)				
a -> Qm					
p - Rp				Qd -> result	MVE
inactive > Qd v v v v v v v v v	int32x4_t a, mve_pred16_t p)				
a -> Qm			VREV64T.32 Qd,Qm		
p -> Rp	uint8x16_t [arm_]vrev64q_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive Qd VMSR P0,Rp Qd Possult MVE	uint8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
inactive Qd VMSR P0,Rp Qd Possult MVE		p -> Rp	VREV64T.8 Qd,Qm		
a -> Qm	uint16x8_t [arm_]vrev64q_m[_u16](uint16x8_t		VMSR P0,Rp	Qd -> result	MVE
p -> Rp VREV64T.16 Qd,Qm	inactive, uint16x8_t a, mve_pred16_t p)				
Initial Init		-			
a -> Qm	uint32x4 t [arm]vrey64g m[u32](uint32x4 t			Od -> result	MVE
P -> Rp				Za > Icsuit	,
Doat16x8_t [_arm_]vrev64q_m[_f16](float16x8_t nactive -> Qd a -> Qm vPST vREV64T.16 Qd,Qm vPST vREV64T.16 Qd,Qm vPST vREV64T.16 Qd,Qm vPST vREV64T.16 Qd,Qm vPST vREV64T.8 Qd,Qm vPST vREV64T.8 Qd,Qm vPST vREV64T.16 Qd,Qm vPST vREV64T.8 Qd,Qm vPST vREV64T.8 Qd,Qm vPST vREV64T.8 Qd,Qm vPST vREV64T.16 Qd,Qm vPST vREV64T.8 Qd,Qm vPST vREV64T.8 Qd,Qm vPST vREV64T.8 Qd,Qm vPST vREV64T.8 Qd,Qm vPST vREV64T.16 Qd,Qm vPST vPST vPST vPST vPST vPST vPST vPST vPST vREV64T.32 Qd,Qm vPST vREV64T.32 Qd,Qm vPST vREV64T.32 Qd,Qm vPST vREV64T.32 Qd,Qm vPST vRSV64T.32 Qd,Qm vPST vRSV64T.32 Qd,Qm vPST vRSV64T.32 Qd,Qm vPST vPST vPST vRSV64T.32 Qd,Qm vPST vPST vRSV64T.92 Qd -> result vPST vRSV64T.92 Qd -> result vPST vRSV64T.92 Qd,Qm vPST vRSV64T.92 Qd -> result vPST vRSV64T.92 Qd -> result vPST vRSV64T.92 Qd,Qm vPST		•			
a -> Qm	float16x8 t [arm]vrev6/a m[f16]/float16x8 t			Od -> recult	MVF
p -> Rp		-		Qu -> icsuit	141 4 17
Doat32x4_t [_arm_]vrev64q_m[_f32](float32x4_t inactive -> Qd vMSR P0,Rp vPST vREV64T.32 Qd,Qm vMSR P0,Rp vPST vP	mactive, moatroxo_t a, mve_predio_t p)				
nactive, float32x4_t a, mve_pred16_t p) a -> Qm p -> Rp VPST VREV64T.32 Qd,Qm nt8x16_t [_arm_]vrev64q_x[_s8](int8x16_t a, mve_pred16_t p) a -> Qm p -> Rp VMSR P0,Rp VPST VPST VREV64T.8 Qd,Qm Qd -> result MVE nt16x8_t [_arm_]vrev64q_x[_s16](int16x8_t a, mve_pred16_t p) a -> Qm VMSR P0,Rp VPST VREV64T.16 Qd,Qm Qd -> result MVE nt16x8_t [_arm_]vrev64q_x[_s16](int16x8_t a, mve_pred16_t p) a -> Qm VMSR P0,Rp VPST VREV64T.16 Qd,Qm Qd -> result MVE nt32x4_t [_arm_]vrev64q_x[_s32](int32x4_t a, mve_pred16_t p) a -> Qm VMSR P0,Rp VPST VPST VREV64T.32 Qd,Qm Qd -> result MVE nint8x16_t [_arm_]vrev64q_x[_u8](uint8x16_t a, mve_pred16_t p) a -> Qm VMSR P0,Rp VPST VREV64T.32 Qd,Qm Qd -> result MVE	floot22v4 + f - omm - lymays64a f - 6201/61 + 22 4 -			041	MVE
p -> Rp		,		Qu -> result	WIVE
nt8x16_t [_arm_]vrev64q_x[_s8](int8x16_t a, nve_pred16_t p) a -> Qm p -> Rp VMSR P0,Rp VPST VREV64T.8 Qd,Qm Qd -> result MVE nt16x8_t [_arm_]vrev64q_x[_s16](int16x8_t a, nve_pred16_t p) a -> Qm p -> Rp VMSR P0,Rp VPST VREV64T.16 Qd,Qm Qd -> result MVE nt32x4_t [_arm_]vrev64q_x[_s32](int32x4_t a, nve_pred16_t p) a -> Qm p -> Rp VMSR P0,Rp VPST VREV64T.32 Qd,Qm Qd -> result MVE ntint8x16_t [_arm_]vrev64q_x[_u8](uint8x16_t a, nve_pred16_t p) a -> Qm VMSR P0,Rp VPST VMSR P0,Rp VPST Qd -> result MVE	macuve, moat52x4_t a, mve_pred16_t p)				
P-> Rp					1.55
VREV64T.8 Qd,Qm				Qd -> result	MVE
nt16x8_t [_arm_]vrev64q_x[_s16](int16x8_t a, nve_pred16_t p) a -> Qm p -> Rp VMSR P0,Rp VPST VREV64T.16 Qd,Qm Qd -> result MVE nt32x4_t [_arm_]vrev64q_x[_s32](int32x4_t a, nve_pred16_t p) a -> Qm VMSR P0,Rp VPST VPST VPST VREV64T.32 Qd,Qm Qd -> result MVE ntint8x16_t [_arm_]vrev64q_x[_u8](uint8x16_t a, nve_pred16_t p) a -> Qm VMSR P0,Rp VPST VREV64T.32 Qd,Qm Qd -> result MVE	mve_pred16_t p)	p -> Rp			
P - Rp					
P-> Rp	int16x8_t [arm_]vrev64q_x[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Qd -> result	MVE
VREV64T.16 Qd,Qm nt32x4_t [_arm_]vrev64q_x[_s32](int32x4_t a, a -> Qm p -> Rp VPST VREV64T.32 Qd,Qm nint8x16_t [_arm_]vrev64q_x[_u8](uint8x16_t a, a -> Qm vMSR P0,Rp VPST VREV64T.32 Qd,Qm vws_pred16_t p) VMSR P0,Rp VMSR P0,Rp VMSR P0,Rp VPST VMSR P0,Rp VPST VPST VREV64T.32 VMSR P0,Rp VPST VPST VPST VPST VPST VPST VPST VPST	mve_pred16_t p)		VPST		
nt32x4_t [_arm_]vrev64q_x[_s32](int32x4_t a, a -> Qm VMSR P0,Rp Qd -> result MVE nve_pred16_t p) vPST VPST VREV64T.32 Qd,Qm vint8x16_t [_arm_]vrev64q_x[_u8](uint8x16_t a, a -> Qm VMSR P0,Rp Qd -> result MVE nve_pred16_t p) p -> Rp VPST VPST MVE	I/				
P -> Rp	nt32x4 t [arm]vrev64a x[s32](int32x4 t a	a -> Om		Od -> result	MVE
VREV64T.32 Qd,Qm uint8x16_t [_arm_]vrev64q_x[_u8](uint8x16_t a,				24 - 100011	1 2
sint8x16_t [_arm_]vrev64q_x[_u8](uint8x16_t a, a -> Qm VMSR P0,Rp Qd -> result MVE nve_pred16_t p) p -> Rp VPST	mre_predio_t p)	P -> KP			
nve_pred16_t p)	wint0v16 t [arm]vrov64a v[v0]/vint0v16 t a	0 > 0==		Od > mooult	MVE
				Qu -> result	IVIVE
VREV64T.8 Qd,Qm	mve_pred10_t p)	p -> K p			

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vrshlq_m[_u32](uint32x4_t inactive, uint32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn	VMSR P0,Rp VPST VRSHLT.U32 Qd,Qm,Qn	Qd -> result	MVE
int8x16_t [_arm_]vrshlq_x[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	p -> Rp a -> Qm b -> Qn 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	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)	p -> Rp 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	a -> Qm 1 <= imm <= 8	VSHLLB.S8 Qd,Qm,#imm	Qd -> result	MVE
imm) 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	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)	p -> Rp 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 PO,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	VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4_t [arm_]vshlltq_x[_n_u16](uint16x8_t a, const int imm, mve_pred16_t p)	p -> Rp a -> Qm 1 <= imm <= 16	VSHLLTT.U8 Qd,Qm,#imm 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	VSHL.S8 Qd,Qm,Qn	Qd -> result	MVE/NEON
int16x8_t [arm_]vshlq[_s16](int16x8_t a, int16x8_t b)	b -> Qn a -> Qm	VSHL.S16 Qd,Qm,Qn	Qd -> result	MVE/NEON
int32x4_t [arm_]vshlq[_s32](int32x4_t a, int32x4_t b)	b -> Qn a -> Qm	VSHL.S32 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vshlq[_u8](uint8x16_t a, int8x16_t b)	b -> Qn 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	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)	p -> Rp 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 Qd,Qm,Qn	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	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)	p -> Rp inactive -> Qd a -> Qm 0 <= imm <= 15	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)	p -> Rp inactive -> Qd a -> Qm 0 <= imm <= 31	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)	p -> Rp 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	VMSR P0,Rp VPST	Qda -> result	MVE
int32x4_t [_arm_]vshlq_m_r[_s32](int32x4_t a, int32_t b, mve_pred16_t p)	p -> Rp a -> Qda b -> Rm	VSHLT.S16 Qda,Rm VMSR P0,Rp VPST VSHLT.S22 Qda, Rm	Qda -> result	MVE
uint8x16_t [arm_]vshlq_m_r[_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	p -> Rp a -> Qda b -> Rm p -> Rp	VSHLT.S32 Qda,Rm 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	VMSR P0,Rp VPST	Qda -> result	MVE
int8x16_t [arm_]vrshrnbq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	p -> Rp a -> Qd b -> Qm 1 <= imm <= 8	VSHLT.U32 Qda,Rm VRSHRNB.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vrshrnbq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VRSHRNB.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vrshrnbq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VRSHRNB.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vrshrnbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VRSHRNB.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vrshrnbq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNBT.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [_arm_]vrshrnbq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16	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.132 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vrshrntq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VRSHRNT.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vrshrntq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VRSHRNT.132 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vrshrntq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNTT.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vrshrntq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNTT.132 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 <=	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)	32 inactive -> Qd a -> Qm 1 <= imm <= 8 p -> Rp	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)	inactive -> Qd a -> Qm 1 <= imm <= 16 p -> Rp	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)	inactive -> Qd a -> Qm 1 <= imm <= 32 p -> Rp	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)	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 p -> Rp	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)	inactive -> Qd a -> Qm 1 <= imm <= 32 p -> Rp	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)	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 p -> Rp	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)	a -> Qm 1 <= imm <= 32 p -> Rp	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)	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 p -> Rp	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)	a -> Qm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VRSHRT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vshrnbq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VSHRNB.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vshrnbq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VSHRNB.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [arm_]vshrnbq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VSHRNB.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vshrnbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VSHRNB.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vshrnbq_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	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)	p -> Rp 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 p -> Rp	VMSR P0,Rp VPST VSRIT.32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vsriq_m[_n_u8](uint8x16_t a, uint8x16_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSRIT.8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vsriq_m[_n_u16](uint16x8_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSRIT.16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vsriq_m[_n_u32](uint32x4_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSRIT.32 Qd,Qm,#imm	Qd -> result	MVE
float16_t [arm_]vgetq_lane[_f16](float16x8_t a, const int idx)	$a \rightarrow Qn$ $0 \le idx \le 7$	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 -> Qn 0 <= idx <= 1	VMOV Rt1,Rt2,D(2*n+idx)	[Rt1,Rt2] -> result	MVE/NEON
uint8_t [arm_]vgetq_lane[_u8](uint8x16_t a, const int idx)	a -> Qn 0 <= idx <= 15	VMOV.U8 Rt,Qn[idx]	Rt -> result	MVE/NEON
uint16_t [arm_]vgetq_lane[_u16](uint16x8_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 7$	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 -> Rt b -> 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) int32x4_t [arm_]vuninitializedq_s32(void)			Qd -> result Qd -> result	MVE MVE
int64x2_t [arm_]vuninitializedq_s64(void)			Qd -> result	MVE
uint8x16_t [arm_]vuninitializedq_u8(void)			Qd -> result	MVE
uint16x8_t [arm_]vuninitializedq_u16(void)			Qd -> result	MVE
uint32x4_t [arm_]vuninitializedq_u32(void)			Qd -> result	MVE
uint64x2_t [_arm_]vuninitializedq_u64(void)			Qd -> result	MVE
float16x8_t [_arm_]vuninitializedq_f16(void) float32x4_t [_arm_]vuninitializedq_f32(void)			Qd -> result Qd -> result	MVE MVE
int8x16_t [_arm_]vuninitializedq(int8x16_t t)	t -> Do Not		Qd -> result 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) int32x4_t [_arm_]vreinterpretq_s32[_s8](int8x16_t a)	a -> Qd a -> Qd	NOP NOP	Qd -> result Qd -> result	MVE/NEON MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_s8](int8x16_t a)	a -> Qd a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_s8](int8x16_t a)	a -> Od	NOP	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vreinterpretq_u16[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [arm_]vreinterpretq_s64[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [_arm_]vreinterpretq_s8[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [_arm_]vreinterpretq_s32[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [_arm_]vreinterpretq_s64[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_s16](int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [_arm_]vreinterpretq_s8[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [_arm_]vreinterpretq_s16[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vreinterpretq_u32[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [arm_]vreinterpretq_s64[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_s32](int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [_arm_]vreinterpretq_s8[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [_arm_]vreinterpretq_s16[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [_arm_]vreinterpretq_s32[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vreinterpretq_u16[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [arm_]vreinterpretq_s64[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_f32](float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [arm_]vreinterpretq_s8[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [arm_]vreinterpretq_s16[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [_arm_]vreinterpretq_s32[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [_arm_]vreinterpretq_f32[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [_arm_]vreinterpretq_s64[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [_arm_]vreinterpretq_f16[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [_arm_]vreinterpretq_s8[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [_arm_]vreinterpretq_s16[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [_arm_]vreinterpretq_s32[_u16](uint16x8_t a) float32x4_t [_arm_]vreinterpretq_f32[_u16](uint16x8_t	a -> Qd a -> Qd	NOP NOP	Qd -> result Qd -> result	MVE/NEON MVE/NEON
a)	1 " 2 "	1.02	Qu > resuit	
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	NOP	Qd -> result	MVE/NEON
a) uint64x2_t [arm_]vreinterpretq_u64[_u16](uint16x8_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)	a => Od	NOP	Od -> recult	MVE/NEON
	a -> Qd a -> Qd	NOP NOP	Qd -> result Qd -> result	MVE/NEON 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)	a -> Qd	NOP	Qd -> result	MVE/NEON
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)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [_arm_]vreinterpretq_u64[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
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)	a -> Qd	NOP	Qd -> result	MVE/NEON
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)	a -> Qd	NOP	Qd -> result	MVE/NEON
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)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_u64](uint64x2_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a) int64x2_t [arm_]vreinterpretq_s64[_u64](uint64x2_t a)	a -> Od	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_u64](uint64x2_t	a -> Qd a -> Qd	NOP	Qd -> result	MVE/NEON MVE/NEON
a)	01	NOD	0.1	MUEATECN
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	Od -> 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)	a -> Qd	NOP	Qd -> result	MVE/NEON
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 -> Qd	NOP	Qd -> result	MVE/NEON
a)	0.1	Won	0.1	A TENERAL
int64x2_t [_arm_]vreinterpretq_s64[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64_t [arm_]lsll(uint64_t value, int32_t shift)	value -> [RdaHi,RdaLo] shift -> Rm	LSLL RdaLo,RdaHi,Rm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]asrl(int64_t value, int32_t shift)	value -> [RdaHi,RdaLo] shift -> Rm	ASRL RdaLo,RdaHi,Rm	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]uqrshll(uint64_t value, int32_t shift)	value -> [RdaHi,RdaLo] shift -> Rm	UQRSHLL RdaLo,RdaHi,#64,Rm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]uqrshll_sat48(uint64_t value, int32_t shift)	value -> [RdaHi,RdaLo] shift -> Rm	UQRSHLL RdaLo,RdaHi,#48,Rm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]sqrshrl(int64_t value, int32_t shift)	value -> [RdaHi,RdaLo] shift -> Rm	SQRSHRL RdaLo,RdaHi,#64,Rm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]sqrshrl_sat48(int64_t value, int32_t shift)	value -> [RdaHi,RdaLo] shift -> Rm	SQRSHRL RdaLo,RdaHi,#48,Rm	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]uqshll(uint64_t value, const int shift)	value -> [RdaHi,RdaLo] 1 <= shift <= 32	UQSHLL RdaLo,RdaHi,#shift	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]urshrl(uint64_t value, const int shift)	value -> [RdaHi,RdaLo] 1 <= shift <= 32	URSHRL RdaLo,RdaHi,#shift	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]srshrl(int64_t value, const int shift)	value -> [RdaHi,RdaLo] 1 <= shift <= 32	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