

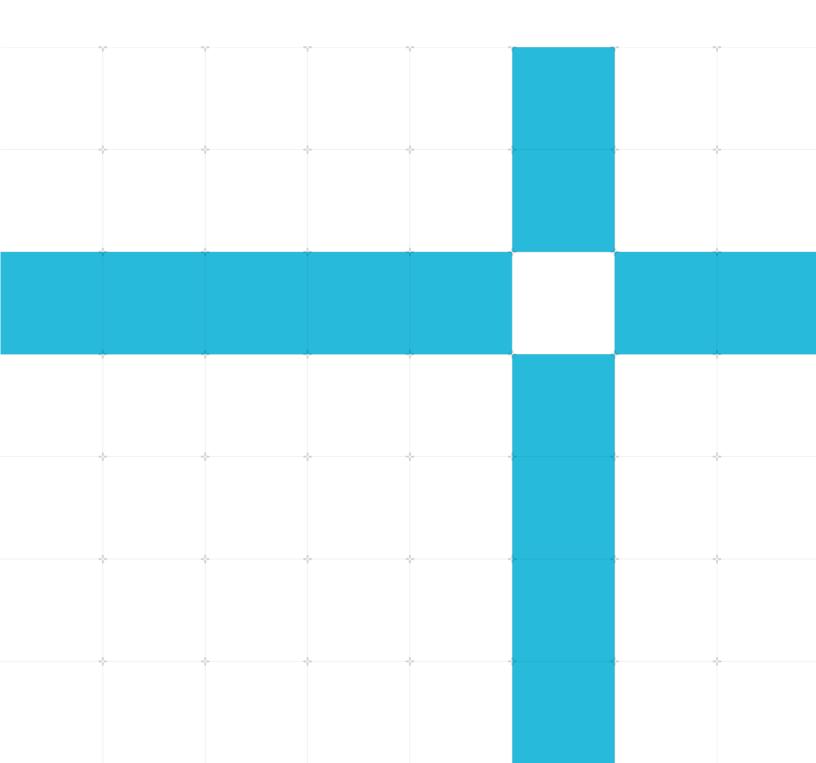
Arm MVE Intrinsics Reference for ACLE Q2 2019

Non-Confidential

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

Issue Q219-00

101809



Arm MVE Intrinsics Reference 101809

Arm MVE Intrinsics

Reference

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

Release information

Document history

Issue	Date	Confidentiality	Change
Q219-00	30 June 2019	Non-Confidential	Version ACLE Q2 2019.

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

Arm MVE Intrinsics Reference 101809

English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

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

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

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Web Address

.http://www.arm.com.

About this document

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

List of Intrinsics

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [arm_]vcreateq_f16(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
float32x4_t [arm_]vcreateq_f32(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
int8x16_t [arm_]vcreateq_s8(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
int16x8_t [arm_]vcreateq_s16(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[2],Qd[1],Rt4,Rt2	Qd -> result	MVE
int32x4_t [arm_]vcreateq_s32(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[2],Qd[1],Rt4,Rt2	Qd -> result	MVE
int64x2_t [arm_]vcreateq_s64(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint8x16_t [arm_]vcreateq_u8(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[2],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint16x8_t [arm_]vcreateq_u16(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint32x4_t [arm_]vcreateq_u32(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint64x2_t [arm_]vcreateq_u64(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint8x16_t [_arm_]vddupq[_n]_u8(uint32_t a, const int imm)	a -> Rn imm in [1,2,4,8]	VDDUP.U8 Qd,Rn,imm	Qd -> result	MVE
uint16x8_t [_arm_]vddupq[_n]_u16(uint32_t a, const int imm)	a -> Rn imm in [1,2,4,8]	VDDUP.U16 Qd,Rn,imm	Qd -> result	MVE
uint32x4_t [_arm_]vddupq[_n]_u32(uint32_t a, const int imm)	a -> Rn imm in [1,2,4,8]	VDDUP.U32 Qd,Rn,imm	Qd -> result	MVE
uint8x16_t [_arm_]vddupq[_wb]_u8(uint32_t * a, const int imm)	*a -> Rn imm in [1,2,4,8]	VDDUP.U8 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [_arm_]vddupq[_wb]_u16(uint32_t * a, const int imm)	*a -> Rn imm in [1,2,4,8]	VDDUP.U16 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]vddupq[_wb]_u32(uint32_t * a, const int imm)	*a -> Rn imm in [1,2,4,8]	VDDUP.U32 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [_arm_]vddupq_m[_n_u8](uint8x16_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn	VMSR P0,Rp VPST	Qd -> result	MVE
maetive, unito 2_t a, const int mini, inve_predito_t p)	imm in [1,2,4,8] p -> Rp	VDDUPT.U8 Qd,Rn,imm		
uint16x8_t [_arm_]vddupq_m[_n_u16](uint16x8_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn	VMSR P0,Rp VPST	Qd -> result	MVE
	imm in [1,2,4,8] p -> Rp	VDDUPT.U16 Qd,Rn,imm		
uint32x4_t [_arm_]vddupq_m[_n_u32](uint32x4_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn	VMSR P0,Rp VPST	Qd -> result	MVE
, , , , , , , , , , , , , , , , , , , ,	imm in [1,2,4,8] p -> Rp	VDDUPT.U32 Qd,Rn,imm		
uint8x16_t [_arm_]vddupq_m[_wb_u8](uint8x16_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn	VMSR P0,Rp VPST	Qd -> result Rn -> *a	MVE
,	imm in [1,2,4,8] p -> Rp	VDDUPT.U8 Qd,Rn,imm		
uint16x8_t [_arm_]vddupq_m[_wb_u16](uint16x8_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn	VMSR P0,Rp VPST	Qd -> result Rn -> *a	MVE
, , , ,	imm in [1,2,4,8] p -> Rp	VDDUPT.U16 Qd,Rn,imm		
uint32x4_t [_arm_]vddupq_m[_wb_u32](uint32x4_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn	VMSR P0,Rp VPST	Qd -> result Rn -> *a	MVE
	imm in [1,2,4,8] p -> Rp	VDDUPT.U32 Qd,Rn,imm		
uint8x16_t [_arm_]vdwdupq[_n]_u8(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm	VDWDUP.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
uint16x8_t [arm_]vdwdupq[_n]_u16(uint32_t a,	imm in [1,2,4,8] a -> Rn	VDWDUP.U16 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32_t b, const int imm)	b -> Rm imm in [1,2,4,8]	AMMINISTRA SA		Name -
uint32x4_t [_arm_]vdwdupq[_n]_u32(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm	VDWDUP.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint8x16_t [_arm_]vdwdupq[_wb]_u8(uint32_t * a,	imm in [1,2,4,8] *a -> Rn	VDWDUP.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32_t b, const int imm)	b -> Rm imm in [1,2,4,8]	ADMDIB 114 O 1 B B :	Rn -> *a	Myre
uint16x8_t [_arm_]vdwdupq[_wb]_u16(uint32_t * a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U16 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vdwdupq[_wb]_u32(uint32_t * a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U32 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [_arm_]vdwdupq_m[_n_u8](uint8x16_t inactive, uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	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)	inactive -> Qd a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32x4_t [_arm_]vdwdupq_m[_n_u32](uint32x4_t inactive, uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint8x16_t [arm_]vdwdupq_m[_wb_u8](uint8x16_t inactive, uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDWDUPT.U8 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [arm_]vdwdupq_m[_wb_u16](uint16x8_t inactive, uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]vdwdupq_m[_wb_u32](uint32x4_t inactive, uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [arm_]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
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) uint16x8_t [_arm_]vidupq[_wb]_u16(uint32_t * a, const	*a -> Rn imm in [1,2,4,8] *a -> Rn	VIDUP.U8 Qd,Rn,imm VIDUP.U16 Qd,Rn,imm	Qd -> result Rn -> *a Qd -> result	MVE MVE
int imm) uint32x4 t [_arm_ vidupq[_wb]_u32(uint32_t *a, const	imm in [1,2,4,8] *a -> Rn	VIDUP.U32 Qd,Rn,imm	Rn -> *a Qd -> result	MVE
int imm) uint8x16_t [_arm_]vidupq_m[_n_u8](uint8x16_t inactive, uint32_t a, const int imm, mve_pred16_t p)	imm in [1,2,4,8] inactive -> Qd a -> Rn imm in [1,2,4,8]	VMSR P0,Rp VPST VIDUPT.U8 Qd,Rn,imm	Rn -> *a Qd -> result	MVE
uint16x8_t [_arm_]vidupq_m[_n_u16](uint16x8_t inactive, uint32_t a, const int imm, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U16 Qd,Rn,imm	Qd -> result	MVE
uint32x4_t [_arm_]vidupq_m[_n_u32](uint32x4_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U32 Qd,Rn,imm	Qd -> result	MVE
uint8x16_t [arm_]vidupq_m[_wb_u8](uint8x16_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U8 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [_arm_]vidupq_m[_wb_u16](uint16x8_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U16 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]vidupq_m[_wb_u32](uint32x4_t inactive, uint32_t * a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U32 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [_arm_]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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]viwdupq[_n]_u32(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm imm in [1,2,4,8]	VIWDUP.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint8x16_t [_arm_]viwdupq[_wb]_u8(uint32_t * a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VIWDUP.U8 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [_arm_]viwdupq[_wb]_u16(uint32_t * a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VIWDUP.U16 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]viwdupq[_wb]_u32(uint32_t * a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VIWDUP.U32 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [_arm_]viwdupq_m[_n_u8](uint8x16_t inactive, uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
uint16x8_t [_arm_]viwdupq_m[_n_u16](uint16x8_t inactive, uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32x4_t [_arm_]viwdupq_m[_n_u32](uint32x4_t inactive, uint32_t a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint8x16_t [_arm_]viwdupq_m[_wb_u8](uint8x16_t inactive, uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U8 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [_arm_]viwdupq_m[_wb_u16](uint16x8_t inactive, uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U16 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [_arm_]viwdupq_m[_wb_u32](uint32x4_t inactive, uint32_t * a, uint32_t b, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn b -> Rm imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIWDUPT.U32 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
int8x16_t [arm_]vdupq_n_s8(int8_t a)	a -> Rt	VDUP.8 Qd,Rt	Qd -> result	MVE/NEON
int16x8_t [arm_]vdupq_n_s16(int16_t a)	a -> Rt	VDUP.16 Qd,Rt	Qd -> result	MVE/NEON
int32x4_t [arm_]vdupq_n_s32(int32_t a)	a -> Rt	VDUP.32 Qd,Rt	Qd -> result	MVE/NEON
uint8x16_t [arm_]vdupq_n_u8(uint8_t a)	a -> Rt	VDUP.8 Qd,Rt	Qd -> result	MVE/NEON
uint16x8_t [arm_]vdupq_n_u16(uint16_t a)	a -> Rt	VDUP.16 Qd,Rt	Qd -> result	MVE/NEON
uint32x4_t [arm_]vdupq_n_u32(uint32_t a)	a -> Rt	VDUP.32 Qd,Rt	Qd -> result	MVE/NEON
float16x8_t [arm_]vdupq_n_f16(float16_t a)	a -> Rt	VDUP.16 Qd,Rt	Qd -> result	MVE/NEON
float32x4_t [_arm_]vdupq_n_f32(float32_t a) int8x16_t [_arm_]vdupq_m[_n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p)	a -> Rt inactive -> Qd a -> Rt	VDUP.32 Qd,Rt VMSR P0,Rp VPST	Qd -> result Qd -> result	MVE/NEON MVE
int16x8_t [_arm_]vdupq_m[_n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Rt p -> Rp	VDUPT.8 Qd,Rt VMSR P0,Rp VPST VDUPT.16 Qd,Rt	Qd -> result	MVE
int32x4_t [_arm_]vdupq_m[_n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.32 Qd,Rt	Qd -> result	MVE
uint8x16_t [arm_]vdupq_m[_n_u8](uint8x16_t inactive, uint8_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.8 Qd,Rt	Qd -> result	MVE
uint16x8_t [arm_]vdupq_m[_n_u16](uint16x8_t inactive, uint16_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.16 Qd,Rt	Qd -> result	MVE
uint32x4_t [_arm_]vdupq_m[_n_u32](uint32x4_t inactive, uint32_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.32 Qd,Rt	Qd -> result	MVE
float16x8_t [_arm_]vdupq_m[_n_f16](float16x8_t inactive, float16_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.16 Qd,Rt	Qd -> result	MVE
float32x4_t [_arm_]vdupq_m[_n_f32](float32x4_t inactive, float32_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.32 Qd,Rt	Qd -> result	MVE
mve_pred16_t [arm_]vcmpeqq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMP.F16 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpeqq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMP.F32 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpeqq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VCMP.I8 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpeqq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VCMP.I16 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
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) mve_pred16_t [arm_]vcmpeqq[_u16](uint16x8_t a,	b -> Qm a -> Qn	VMRS Rd,P0 VCMP.I16 eq,Qn,Qm	Rd -> result	MVE
uint16x8_t b) mve_pred16_t [arm_]vcmpeqq[_u32](uint32x4_t a,	b -> Qm a -> On	VMRS Rd,P0 VCMP.I32 eq,Qn,Qm	Rd -> result	MVE
uint32x4_t b) mve_pred16_t [arm_]vcmpeqq[_n_f16](float16x8_t a,	b -> Qm a -> On	VMRS Rd,P0 VCMP.F16 eq,Qn,Rm	Rd -> result	MVE
float16_t b)	b->Rm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VCMP.F32 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VCMP.I8 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VCMP.I16 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq[_n_s32](int32x4_t a, int32 t b)	a -> Qn b -> Rm	VCMP.I32 eq,Qn,Rm VMRS Rd.P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpeqq[_n_u8](uint8x16_t a, uint8 t b)	a -> Qn	VCMP.I8 eq,Qn,Rm	Rd -> result	MVE
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) mve_pred16_t [arm_]vcmpeqq[_n_u32](uint32x4_t a,	b -> Rm a -> Qn	VMRS Rd,P0 VCMP.I32 eq,Qn,Rm	Rd -> result	MVE
uint32_t b) mve_pred16_t [arm_]vcmpeqq_m[_f16](float16x8_t a,	b -> Rm a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCMPT.F16 eq,Qn,Qm		
mve_pred16_t [arm_]vcmpeqq_m[_f32](float32x4_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Ku -> resuit	MVE
	p -> Rp	VCMPT.F32 eq,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rd -> result	MVE
	p -> Rp	VCMPT.I8 eq,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rd -> result	MVE
mitoto_to, mie_predio_tp)	p -> Rp	VCMPT.I16 eq,Qn,Qm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpeqq_m[_s32](int32x4_t a,	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCMPT.I32 eq,Qn,Qm		
mve_pred16_t [arm_]vcmpeqq_m[_u8](uint8x16_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VCMPT.I8 eq,Qn,Qm		
mve pred16 t[arm]vcmpeqq m[u16](uint16x8 ta,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST VCMPT.I16 eq,Qn,Qm	Ku -> result	WIVE
116 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	p -> Rp	VMRS Rd,P0	7.1) am
mve_pred16_t [arm_]vcmpeqq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rd -> result	MVE
	p -> Rp	VCMPT.I32 eq,Qn,Qm VMRS Rd,P0		
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
,	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 p -> Rp	VPST VCMPT.F32 eq,Qn,Rm		
mve_pred16_t [arm_]vcmpeqq_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.I8 eq,Qn,Rm		
		VMRS Rd,P0		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpeqq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I16 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR PO,Rp VPST VCMPT.I32 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq_m[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I8 eq.Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpeqq_m[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR PO,Rp VMSR PO,Rp VPST VCMPT.I16 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpeqq_m[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I32 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMP.F16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMP.F32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VCMP.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VCMP.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VCMP.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VCMP.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VCMP.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VCMP.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR Rd,P0 VMSR PO,Rp VPST VCMPT.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR PO,Rp VPST VCMPT.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VCMP.F16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VCMP.F32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VCMP.I8 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VCMP.I16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpneq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VCMP.I32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VCMP.I8 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VCMP.I16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VCMP.I32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpneq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I8 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)	$\begin{array}{l} a \rightarrow Qn \\ b \rightarrow Rm \\ p \rightarrow Rp \end{array}$	VMSR P0,Rp VPST VCMPT.I8 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpneq_m[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.I32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMP.F16 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgeq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMP.F32 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
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 VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgeq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S8 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgeq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S16 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpgeq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S32 ge,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VCMP.F16 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VCMP.F32 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VCMP.S8 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [_arm_]vcmpgeq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VCMP.S16 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VCMP.S32 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgeq_m[_n_f16](float16x8_t	a -> Qn	VMSR P0,Rp	Rd -> result	MVE
a, float16_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VCMPT.F16 ge,Qn,Rm		
mve_pred16_t [arm_]vcmpgeq_m[_n_f32](float32x4_t	a -> On	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
a, float32_t b, mve_pred16_t p) mve_pred16_t [arm_]vcmpgeq_m[_n_s8](int8x16_t a,	b -> Rm p -> Rp	VPST VCMPT.F32 ge,Qn,Rm		
		VMRS Rd,P0	D1 to	NOTE
int8_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Rd -> result	MVE
- · · · - · · · · · · · · · · · · · · ·	p -> Rp	VCMPT.S8 ge,Qn,Rm VMRS Rd,P0		
mve_pred16_t [_arm_]vcmpgeq_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 -> Rp	VCMPT.S16 ge,Qn,Rm		
mve_pred16_t [arm_]vcmpgeq_m[_n_s32](int32x4_t a,	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VCMPT.S32 ge,Qn,Rm		
mve_pred16_t [arm_]vcmpgtq[_f16](float16x8_t a,	a -> Qn	VMRS Rd,P0 VCMP.F16 gt,Qn,Qm	Rd -> result	MVE
float16x8_t b)	b->Qm	VMRS Rd,P0 VCMP.F32 gt,Qn,Qm	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VMRS Rd,P0		
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,	a -> Qn	VCMP.S32 gt,Qn,Qm	Rd -> result	MVE
int32x4_t b) mve_pred16_t [arm_]vcmpgtq_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 p -> Rp	VPST VCMPT.F16 gt,Qn,Qm		
mve_pred16_t [arm_]vcmpgtq_m[_f32](float32x4_t a,	a -> On	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST VCMPT.F32 gt,Qn,Qm	Ted 5 Testal	
	p -> Rp	VMRS Rd,P0		
mve_pred16_t [_arm_]vcmpgtq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rd -> result	MVE
	p -> Rp	VCMPT.S8 gt,Qn,Qm VMRS Rd,P0		
mve_pred16_t [_arm_]vcmpgtq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rd -> result	MVE
introxe_t b, inve_predio_t p)	p -> QIII	VCMPT.S16 gt,Qn,Qm		
mve_pred16_t [arm_]vcmpgtq_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 gt,Qn,Qm		
mve_pred16_t [arm_]vcmpgtq[_n_f16](float16x8_t a,	a -> Qn	VMRS Rd,P0 VCMP.F16 gt,Qn,Rm	Rd -> result	MVE
float16_t b)	b->Rm	VMRS Rd,P0		
mve_pred16_t [arm_]vcmpgtq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VCMP.F32 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VCMP.S8 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_n_s16](int16x8_t a, int16 t b)	a -> Qn b -> Rm	VCMP.S16 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpgtq[_n_s32](int32x4_t a,	a -> Qn	VCMP.S32 gt,Qn,Rm	Rd -> result	MVE
int32_t b) mve_pred16_t [arm_]vcmpgtq_m[_n_f16](float16x8_t	b -> Rm a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
a, float16_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VCMPT.F16 gt,Qn,Rm		
mve_pred16_t [arm_]vcmpgtq_m[_n_f32](float32x4_t	a -> Qn	VMRS Rd,P0 VMSR P0,Rp	Rd -> result	MVE
a, float32_t b, mve_pred16_t p)	b -> Rm	VPST	Ta > result	
	p -> Rp	VCMPT.F32 gt,Qn,Rm VMRS Rd,P0		
mve_pred16_t [arm_]vcmpgtq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm	VMSR P0,Rp VPST	Rd -> result	MVE
	p -> Rp	VCMPT.S8 gt,Qn,Rm VMRS Rd,P0		

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [_arm_]vcmpltq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VCMP.S32 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S8 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S16 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.S32 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VCMP.F16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VCMP.F32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VCMP.S8 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VCMP.S16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VCMP.S32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpltq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.F32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S8 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmpltq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.S32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VCMP.U8 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VCMP.U16 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VCMP.U32 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq_m[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U8 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq_m[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U16 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U32 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VCMP.U8 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VCMP.U16 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VCMP.U32 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vcmpcsq_m[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.U8 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq_m[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMRS RQ,F0 VMSR P0,Rp VPST VCMPT.U16 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmpcsq_m[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.U32 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VCMP.U8 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VCMP.U16 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VCMP.U32 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmphiq_m[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U8 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq_m[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U16 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [_arm_]vcmphiq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMPT.U32 hi,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VCMP.U8 hi,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VCMP.U16 hi,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VCMP.U32 hi,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq_m[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.U8 hi,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq_m[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.U16 hi,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [arm_]vcmphiq_m[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMPT.U32 hi,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
int8x16_t [_arm_]vminq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMIN.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vminq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMIN.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vminq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMIN.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vminq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMIN.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vminq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMIN.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vminq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMIN.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vminq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMINT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vminq_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 VMINT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vminq_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 VMINT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vminq_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 VMINT.U8 Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vminq_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 VMINT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vminq_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 VMINT.U32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [arm_]vminaq[_s8](uint8x16_t a, int8x16_t b)	a -> Qda b -> Om	VMINA.S8 Qda,Qm	Qda -> result	MVE
uint16x8_t [_arm_]vminaq[_s16](uint16x8_t a, int16x8_t b)	a -> Qda b -> Qm	VMINA.S16 Qda,Qm	Qda -> result	MVE
uint32x4_t [arm_]vminaq[_s32](uint32x4_t a, int32x4_t b)	a -> Qda b -> Qm	VMINA.S32 Qda,Qm	Qda -> result	MVE
uint8x16_t [_arm_]vminaq_m[_s8](uint8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINAT.S8 Qda,Qm	Qda -> result	MVE
uint16x8_t [_arm_]vminaq_m[_s16](uint16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINAT.S16 Qda,Qm	Qda -> result	MVE
uint32x4_t [_arm_]vminaq_m[_s32](uint32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINAT.S32 Qda,Qm	Qda -> result	MVE
int8_t [arm_]vminvq[_s8](int8_t a, int8x16_t b)	a -> Rda b -> Qm	VMINV.S8 Rda,Qm	Rda -> result	MVE
int16_t [arm_]vminvq[_s16](int16_t a, int16x8_t b)	a -> Rda b -> Qm	VMINV.S16 Rda,Qm	Rda -> result	MVE
int32_t [_arm_]vminvq[_s32](int32_t a, int32x4_t b)	a -> Rda b -> Qm	VMINV.S32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vminvq[_u8](uint8_t a, uint8x16_t b)	a -> Rda b -> Qm	VMINV.U8 Rda,Qm	Rda -> result	MVE
uint16_t [arm_]vminvq[_u16](uint16_t a, uint16x8_t b)	a -> Rda b -> Qm	VMINV.U16 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vminvq[_u32](uint32_t a, uint32x4_t b)	a -> Rda b -> Qm	VMINV.U32 Rda,Qm	Rda -> result	MVE
int8_t [arm_]vminvq_p[_s8](int8_t a, int8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINVT.S8 Rda,Qm	Rda -> result	MVE
int16_t [arm_]vminvq_p[_s16](int16_t a, int16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINVT.S16 Rda,Qm	Rda -> result	MVE
int32_t [arm_]vminvq_p[_s32](int32_t a, int32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINVT.S32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vminvq_p[_u8](uint8_t a, uint8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINVT.U8 Rda,Om	Rda -> result	MVE
uint16_t [_arm_]vminvq_p[_u16](uint16_t a, uint16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINVT.U16 Rda,Om	Rda -> result	MVE
uint32_t [_arm_]vminvq_p[_u32](uint32_t a, uint32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINVT.U32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vminavq[_s8](uint8_t a, int8x16_t b)	a -> Rda b -> Qm	VMINAV.S8 Rda,Qm	Rda -> result	MVE
uint16_t [_arm_]vminavq[_s16](uint16_t a, int16x8_t b)	a -> Rda b -> Qm	VMINAV.S16 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vminavq[_s32](uint32_t a, int32x4_t b)	a -> Rda b -> Qm	VMINAV.S32 Rda,Qm	Rda -> result	MVE
uint8_t [arm_]vminavq_p[_s8](uint8_t a, int8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINAVT.S8 Rda,Qm	Rda -> result	MVE
uint16_t [_arm_]vminavq_p[_s16](uint16_t a, int16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINAVT.S16 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vminavq_p[_s32](uint32_t a, int32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINAVT.S32 Rda,Qm	Rda -> result	MVE
float16x8_t [_arm_]vminnmq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VMINNM.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
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 b -> Qm p -> Rp	VMSR P0,Rp VPST VMINNMT.F16 Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [_arm_]vminnmq_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 VMINNMT.F32 Qd,Qn,Qm	Qd -> result	MVE
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 p -> Rp	VMSR P0,Rp VPST VMINNMAT.F16 Qda,Qm	Qda -> result	MVE
float32x4_t [_arm_]vminnmaq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qda b -> Qm p -> Rp	VMSR P0,Rp VPST VMINNMAT.F32 Qda,Qm	Qda -> result	MVE
float16_t [_arm_]vminnmvq[_f16](float16_t a, float16x8_t b)	a -> Rda b -> Qm	VMINNMV.F16 Rda,Qm	Rda -> result	MVE
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 -> Om	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	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)	p -> Rp 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 p -> Rp	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)	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
uint8x16_t [_arm_]vmaxaq[_s8](uint8x16_t a, int8x16_t b)	a -> Qda b -> Qm	VMAXA.S8 Qda,Qm	Qda -> result	MVE
uint16x8_t [_arm_]vmaxaq[_s16](uint16x8_t a, int16x8_t b)	a -> Qda b -> Qm	VMAXA.S16 Qda,Qm	Qda -> result	MVE
uint32x4_t [_arm_]vmaxaq[_s32](uint32x4_t a, int32x4_t b)	a -> Qda b -> Qm	VMAXA.S32 Qda,Qm	Qda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [arm_]vmaxaq_m[_s8](uint8x16_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
uint16x8_t [arm_]vmaxaq_m[_s16](uint16x8_t a,	p -> Rp a -> Oda	VMAXAT.S8 Qda,Qm VMSR P0,Rp	Oda -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Qua > result	111 1 12
	p -> Rp	VMAXAT.S16 Qda,Qm		
uint32x4_t [_arm_]vmaxaq_m[_s32](uint32x4_t a,	a -> Qda b -> Qm	VMSR P0,Rp VPST	Qda -> result	MVE
int32x4_t b, mve_pred16_t p)	p -> Rp	VMAXAT.S32 Qda,Qm		
int8_t [arm_]vmaxvq[_s8](int8_t a, int8x16_t b)	a -> Rda	VMAXV.S8 Rda,Qm	Rda -> result	MVE
1.100.100.1100.11	b -> Qm	ADMANAGE DE LO	7.1) am
int16_t [arm_]vmaxvq[_s16](int16_t a, int16x8_t b)	a -> Rda b -> Om	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	VMAXV.U16 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vmaxvq[_u32](uint32_t a, uint32x4_t b)	b -> Qm a -> Rda	VMAXV.U32 Rda,Qm	Rda -> result	MVE
	b -> Qm			
int8_t [arm_]vmaxvq_p[_s8](int8_t a, int8x16_t b, mve_pred16_t p)	a -> Rda b -> Om	VMSR P0,Rp VPST	Rda -> result	MVE
live_pred1o_t p)	p -> Rp	VMAXVT.S8 Rda,Qm		
int16_t [arm_]vmaxvq_p[_s16](int16_t a, int16x8_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm	VPST		
int32_t [arm_]vmaxvq_p[_s32](int32_t a, int32x4_t b,	p -> Rp a -> Rda	VMAXVT.S16 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm	VPST	Rua -> resuit	WIVE
	p -> Rp	VMAXVT.S32 Rda,Qm		
uint8_t [arm_]vmaxvq_p[_u8](uint8_t a, uint8x16_t b,	a -> Rda	VMSR P0,Rp VPST	Rda -> result	MVE
mve_pred16_t p)	b -> Qm p -> Rp	VMAXVT.U8 Rda,Qm		
uint16_t [arm_]vmaxvq_p[_u16](uint16_t a, uint16x8_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
wint22 t [www.yyz w22]/wint22 t wint22w4 t	p -> Rp	VMAXVT.U16 Rda,Qm	Ddo > monit	MVE
uint32_t [_arm_]vmaxvq_p[_u32](uint32_t a, uint32x4_t b, mve_pred16_t p)	a -> Rda b -> Om	VMSR P0,Rp VPST	Rda -> result	MVE
,	p -> Rp	VMAXVT.U32 Rda,Qm		
uint8_t [arm_]vmaxavq[_s8](uint8_t a, int8x16_t b)	a -> Rda	VMAXAV.S8 Rda,Qm	Rda -> result	MVE
uint16_t [arm_]vmaxavq[_s16](uint16_t a, int16x8_t b)	b -> Qm a -> Rda	VMAXAV.S16 Rda,Qm	Rda -> result	MVE
uint32_t [arm_]vmaxavq[_s32](uint32_t a, int32x4_t b)	b -> Qm a -> Rda	VMAYAV S22 Pdo Om	Rda -> result	MVE
umt32_t [arm_Jvmaxavq[_s32](umt32_t a, mt32x4_t b)	b -> Om	VMAXAV.S32 Rda,Qm	Rua -> resuit	MVE
uint8_t [arm_]vmaxavq_p[_s8](uint8_t a, int8x16_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	b -> Qm	VPST		
uint16_t [arm_]vmaxavq_p[_s16](uint16_t a, int16x8_t	p -> Rp a -> Rda	VMAXAVT.S8 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Om	VMSK FO,KP VPST	Kua -> resuit	IVIVE
	p -> Rp	VMAXAVT.S16 Rda,Qm		
uint32_t [_arm_]vmaxavq_p[_s32](uint32_t a, int32x4_t b, mve_pred16_t p)	a -> Rda	VMSR P0,Rp VPST	Rda -> result	MVE
b, live_pred16_t p)	b -> Qm p -> Rp	VMAXAVT.S32 Rda,Qm		
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,	a -> Qm b -> Qm	VMAXNM.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t b) float16x8 t [arm]vmaxnmq 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	() 22.02	
	b -> Qm p -> Rp	VMAXNMT.F16 Qd,Qn,Qm		
float32x4_t [arm_]vmaxnmq_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	VPST VMAXNMT F32 Od On Om		
	b -> Qm p -> Rp	VMAXNMT.F32 Qd,Qn,Qm		
float16x8_t [arm_]vmaxnmaq[_f16](float16x8_t a, float16x8_t b)	a -> Qda b -> Qm	VMAXNMA.F16 Qda,Qm	Qda -> result	MVE
float32x4_t [arm_]vmaxnmaq[_f32](float32x4_t a,	a -> Qda	VMAXNMA.F32 Qda,Qm	Qda -> result	MVE
float32x4_t b) float16x8_t [arm_]vmaxnmaq_m[_f16](float16x8_t a,	b -> Qm a -> Qda	VMSR P0,Rp	Qda -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Qui > resuit	111 1 12
float32x4_t [arm_]vmaxnmaq_m[_f32](float32x4_t a,	p -> Rp	VMAXNMAT.F16 Qda,Qm	Qda -> result	MVE
float32x4_t [armjvmaxnmaq_mi_132](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qda b -> Qm	VMSR P0,Rp VPST	Qua -> resuit	MVE
_ / r · · · · - · r /	p -> Rp	VMAXNMAT.F32 Qda,Qm	1	

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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	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)	p -> Rp inactive -> Qd a -> Qn	VMSR PO,Rp VPST	Qd -> result	MVE
int32x4_t [_arm_]vabdq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp inactive -> Qd	VABDT.S16 Qd,Qn,Qm VMSR P0,Rp VPST	Qd -> result	MVE
	a -> Qn b -> Qm p -> Rp	VABDT.S32 Qd,Qn,Qm	Odament	MOVE
uint8x16_t [_arm_]vabdq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vabdq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vabdq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vabdq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vabdq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.F32 Qd,Qn,Qm	Qd -> result	MVE
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) int32x4_t [arm_]vabsq[_s32](int32x4_t a)	a -> Qm a -> Qm	VABS.S16 Qd,Qm VABS.S32 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
float16x8_t [arm_]vabsq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.F16 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	VMSR P0,Rp VPST VABST.F32 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vabsq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S8 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vabsq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S16 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vabsq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VABST.S32 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vadciq[_s32](int32x4_t a, int32x4_t b, unsigned * carry_out)	a -> Qn b -> Qm	VADCI.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
uint32x4_t [_arm_]vadciq[_u32](uint32x4_t a, uint32x4_t b, unsigned * carry_out)	a -> Qn b -> Qm	VADCI.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
int32x4_t [_arm_]vadciq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, unsigned * carry_out, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADCIT.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
uint32x4_t [_arm_]vadciq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, unsigned * carry_out, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADCIT.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [_arm_]vaddq_m[_n_f32](float32x4_t inactive, float32x4_t a, float32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.F32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [_arm_]vaddq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vaddq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vaddq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vaddq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vaddq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [arm_]vaddq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vaddq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vaddq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vaddq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.I32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vaddq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vaddq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vaddq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.I32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vclsq[_s8](int8x16_t a)	a -> Qm	VCLS.S8 Qd,Qm	Qd -> result	MVE/NEON
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) int8x16_t [_arm_]vclsq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	a -> Qm inactive -> Qd a -> Qm p -> Rp	VCLS.S32 Qd,Qm VMSR P0,Rp VPST VCLST.S8 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE
int16x8_t [_arm_]vclsq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCLST.S16 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vclsq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p) int8x16_t [_arm_]vclzq[_s8](int8x16_t a)	inactive -> Qd a -> Qm p -> Rp a -> Qm	VMSR P0,Rp VPST VCLST.S32 Qd,Qm VCLZ.18 Qd,Qm	Qd -> result Qd -> result	MVE MVE/NEON
int16x8_t [arm_]vclzq[_s8](int8x16_t a) int16x8_t [arm_]vclzq[_s16](int16x8_t a)	a -> Qm a -> Qm	VCLZ.18 Qd,Qm VCLZ.116 Qd,Qm	Qd -> result	MVE/NEON MVE/NEON
int32x4_t [_arm_]vclzq[_s32](int32x4_t a)	a -> Qm	VCLZ.I32 Qd,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vclzq[_u8](uint8x16_t a)	a -> Qm	VCLZ.I8 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vclzq[_u16](uint16x8_t a)	a -> Qm	VCLZ.I16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_ vclzq[_u32](uint32x4_t a) int8x16_t [_arm_ vclzq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	a -> Qm inactive -> Qd a -> Qm p -> Rp	VCLZ.132 Qd,Qm VMSR P0,Rp VPST VCLZT.18 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE
int16x8_t [arm_]vclzq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCLZT.I16 Qd,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [_arm_]vclzq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
uint8x16_t [_arm_]vclzq_m[_u8](uint8x16_t inactive, uint8x16_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm	VCLZT.I32 Qd,Qm VMSR P0,Rp VPST	Qd -> result	MVE
uint16x8_t [_arm_]vclzq_m[_u16](uint16x8_t inactive, uint16x8_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm p -> Rp	VCLZT.I8 Qd,Qm VMSR P0,Rp VPST VCLZT.I16 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vclzq_m[_u32](uint32x4_t inactive, uint32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCLZT.I32 Qd,Qm	Qd -> result	MVE
float16x8_t [arm_]vnegq[_f16](float16x8_t a) float32x4_t [arm_]vnegq[_f32](float32x4_t a)	a -> Qm a -> Qm	VNEG.F16 Qd,Qm VNEG.F32 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
int8x16_t [_arm_]vnegq[_s8](int8x16_t a) int16x8_t [_arm_]vnegq[_s16](int16x8_t a) int32x4_t [_arm_]vnegq[_s32](int32x4_t a)	a -> Qm a -> Qm a -> Qm	VNEG.S8 Qd,Qm VNEG.S16 Qd,Qm VNEG.S32 Qd,Qm	Qd -> result Qd -> result Qd -> result	MVE/NEON MVE/NEON MVE/NEON
float16x8_t [arm_]vnegq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VNEGT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vnegq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VNEGT.F32 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vnegq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VNEGT.S8 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vnegq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VNEGT.S16 Qd,Qm	Qd -> result	MVE
int32x4_t [arm_]vnegq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VNEGT.S32 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vmulhq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vmulhq[_s16](int16x8_t a, int16x8_t b) int32x4_t [arm_]vmulhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm a -> Qn	VMULH.S16 Qd,Qn,Qm VMULH.S32 Qd,Qn,Qm	Qd -> result Qd -> result	MVE MVE
uint8x16_t [arm_]vmulhq[_u8](uint8x16_t a, uint8x16_t	b -> Qm a -> Qn	VMULH.U8 Qd,Qn,Qm	Qd -> result	MVE
b) uint16x8_t [arm_]vmulhq[_u16](uint16x8_t a, uint16x8_t b)	b -> Qm a -> Qn b -> Qm	VMULH.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmulhq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMULH.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vmulhq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vmulhq_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 VMULHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vmulhq_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 VMULHT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [arm_]vmulhq_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 VMULHT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmulhq_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 VMULHT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmulhq_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 VMULHT.U32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmullbq_poly[_p8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMULLB.P8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmullbq_poly[_p16](uint16x8_t a, uint16x8_t b) int16x8_t [_arm_]vmullbq_int[_s8](int8x16_t a,	a -> Qn b -> Qm a -> Qn	VMULLB.P16 Qd,Qn,Qm VMULLB.S8 Qd,Qn,Qm	Qd -> result Qd -> result	MVE MVE
int16x8_t [arm_]vmullbq_int[_s8](int8x16_t a, int8x16_t b) int32x4_t [arm_]vmullbq_int[_s16](int16x8_t a,	a -> Qn b -> Qm a -> Qn	VMULLB.S8 Qd,Qn,Qm VMULLB.S16 Qd,Qn,Qm	Qd -> result Qd -> result	MVE
int16x8_t b)	a -> QII b -> Qm	* 1/10 LLD.510 Qu,QII,QIII	Qu -> resuit	IVI V IS

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, uint8x16_t b)	b -> Qm a -> Qn b -> Qm	VMULLB.U8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmullbq_int[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMULLB.U16 Qd,Qn,Qm	Qd -> result	MVE
uint64x2_t [_arm_]vmullbq_int[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMULLB.U32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vmullbq_poly_m[_p8](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VPST VMULLBT.P8 Qd,Qn,Qm		
uint32x4_t [_arm_]vmullbq_poly_m[_p16](uint32x4_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VMULLBT.P16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vmullbq_int_m[_s8](int16x8_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VMULLBT.S8 Qd,Qn,Qm	Qu' > resuit	MYE
int32x4_t [arm_]vmullbq_int_m[_s16](int32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VPST VMULLBT.S16 Qd,Qn,Qm		
int64x2_t [arm_]vmullbq_int_m[_s32](int64x2_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, mve_pred10_t p)	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
	b -> Qm p -> Rp	VMULLBT.U8 Qd,Qn,Qm		
uint32x4_t [arm_]vmullbq_int_m[_u16](uint32x4_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qm p -> Rp	VMULLBT.U16 Qd,Qn,Qm		
uint64x2_t [_arm_]vmullbq_int_m[_u32](uint64x2_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, um32x4_t a, um32x4_t b, mvc_pred10_t p)	b -> Qm p -> Rp	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, uint16x8_t b)	a -> Qn b -> Qm	VMULLT.P16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vmulltq_int[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMULLT.S8 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vmulltq_int[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMULLT.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [_arm_]vmulltq_int[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMULLT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vmulltq_int[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMULLT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmulltq_int[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMULLT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint64x2_t [_arm_]vmulltq_int[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMULLT.U32 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vmulltq_poly_m[_p8](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VPST VMULLTT.P8 Qd,Qn,Qm		
uint32x4_t [_arm_]vmulltq_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 p -> Rp	VPST VMULLTT.P16 Qd,Qn,Qm		
int16x8_t [_arm_]vmulltq_int_m[_s8](int16x8_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	VMULLTT.S8 Qd,Qn,Qm		
int32x4_t [_arm_]vmulltq_int_m[_s16](int32x4_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	VMULLTT.S16 Qd,Qn,Qm		
int64x2_t [arm_]vmulltq_int_m[_s32](int64x2_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
macuve, meszatla, meszatle o, mve_predio_t pj	a -> Qn b -> Qm	VMULLTT.S32 Qd,Qn,Qm		1

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
float16x8_t [_arm_]vmulq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VMUL.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]vmulq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VMUL.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vmulq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VMUL.F16 Qd,Qn,Rm	Qd -> result	MVE/NEON
float32x4_t [arm_]vmulq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VMUL.F32 Qd,Qn,Rm	Qd -> result	MVE/NEON
int8x16_t [arm_]vmulq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMUL.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vmulq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMUL.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vmulq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMUL.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vmulq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VMUL.I8 Qd,Qn,Rm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vmulq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VMUL.I16 Qd,Qn,Rm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vmulq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VMUL.I32 Qd,Qn,Rm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vmulq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMUL.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vmulq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMUL.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vmulq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMUL.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vmulq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VMUL.I8 Qd,Qn,Rm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vmulq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VMUL.I16 Qd,Qn,Rm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vmulq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VMUL.I32 Qd,Qn,Rm	Qd -> result	MVE/NEON
float16x8_t [arm_]vmulq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	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)	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.18 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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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.132 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
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 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_nzcvqc 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_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
int32x4_t [arm_]vsbcq[_s32](int32x4_t a, int32x4_t b, unsigned * carry)	a -> Qn b -> Qm *carry -> Rt	VMRS Rs,FPSCR_nzcvqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzcvqc,Rs VSBC.I32 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_nzevqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzevqc,Rs VSBC.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
int32x4_t [arm_]vsbcq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, unsigned * carry, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm *carry -> Rt p -> Rp	VMRS Rs,FPSCR_nzevqe BFI Rs,Rt,#29,#1 VMSR FPSCR_nzevqe,Rs VMSR P0,Rp VPST VSBCT.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzevqe LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vsbcq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, unsigned * carry, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm *carry -> Rt p -> Rp	VMRS Rs,FPSCR_nzcvqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzcvqc,Rs VMSR P0,Rp VPST VSBCT.132 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
int8x16_t [arm_]vsubq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VSUB.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vsubq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VSUB.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vsubq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VSUB.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vsubq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VSUB.I8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vsubq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VSUB.I16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [arm_]vsubq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VSUB.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vsubq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VSUB.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vsubq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VSUB.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vsubq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VSUB.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vsubq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VSUB.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [arm_]vsubq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VSUB.I16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [arm_]vsubq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VSUB.I32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [arm_]vsubq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VSUB.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [arm_]vsubq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Om	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.116 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vsubq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vsubq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vsubq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vsubq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vsubq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.18 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [arm_]vsubq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [arm_]vsubq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I32 Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [_arm_]vsubq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vsubq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vsubq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.I32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [arm_]vsubq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [arm_]vsubq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [arm_]vsubq_m[_n_f16](float16x8_t inactive, float16x8_t a, float16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.F16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [arm_]vsubq_m[_n_f32](float32x4_t inactive, float32x4_t a, float32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VSUBT.F32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [arm_]vcaddq_rot90[_f16](float16x8_t a, float16x8_t b) float32x4_t [arm_]vcaddq_rot90[_f32](float32x4_t a,	a -> Qn b -> Qm a -> Qn	VCADD.F16 Qd,Qn,Qm,#90 VCADD.F32 Qd,Qn,Qm,#90	Qd -> result Od -> result	MVE/NEON 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) int16x8_t [arm_]vcaddq_rot90[_s16](int16x8_t a,	b -> Qm a -> Qn	VCADD.I16 Qd,Qn,Qm,#90	Qd -> result	MVE
int16x8_t b) int32x4_t [_arm_]vcaddq_rot90[_s32](int32x4_t a,	b -> Qm a -> Qn	VCADD.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
int32x4_t b) uint8x16_t [arm_]vcaddq_rot90[_u8](uint8x16_t a,	b -> Qm a -> Qn	VCADD.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
uint8x16_t b) uint16x8_t [_arm_]vcaddq_rot90[_u16](uint16x8_t a,	b -> Qm a -> Qn	VCADD.I16 Qd,Qn,Qm,#90	Qd -> result	MVE
uint16x8_t b) uint32x4_t [_arm_]vcaddq_rot90[_u32](uint32x4_t a,	b -> Qm a -> Qn	VCADD.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
uint32x4_t b) float16x8_t [arm_]vcaddq_rot270[_f16](float16x8_t a, float16x8_t b)	b -> Qm a -> Qn b -> Qm	VCADD.F16 Qd,Qn,Qm,#270	Qd -> result	MVE/NEON
float32x4_t [_arm_]vcaddq_rot270[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCADD.F32 Qd,Qn,Qm,#270	Qd -> result	MVE/NEON
int8x16_t [_arm_]vcaddq_rot270[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VCADD.I8 Qd,Qn,Qm,#270	Qd -> result	MVE
int16x8_t [_arm_]vcaddq_rot270[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VCADD.I16 Qd,Qn,Qm,#270	Qd -> result	MVE
int32x4_t [arm_]vcaddq_rot270[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VCADD.I32 Qd,Qn,Qm,#270	Qd -> result	MVE
uint8x16_t [arm_]vcaddq_rot270[_u8](uint8x16_t a, 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, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.F16 Qd,Qn,Qm,#90	Qd -> result	MVE
float32x4_t [_arm_]vcaddq_rot90_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.F32 Qd,Qn,Qm,#90	Qd -> result	MVE
int8x16_t [arm_]vcaddq_rot90_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
int16x8_t [arm_]vcaddq_rot90_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I16 Qd,Qn,Qm,#90	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [_arm_]vcaddq_rot90_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
uint8x16_t [_arm_]vcaddq_rot90_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
uint16x8_t [_arm_]vcaddq_rot90_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I16 Qd,Qn,Qm,#90	Qd -> result	MVE
uint32x4_t [_arm_]vcaddq_rot90_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
float16x8_t [arm_]vcaddq_rot270_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.F16 Qd,Qn,Qm,#270	Qd -> result	MVE
float32x4_t [_arm_]vcaddq_rot270_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.F32 Qd,Qn,Qm,#270	Qd -> result	MVE
int8x16_t [arm_]vcaddq_rot270_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I8 Qd,Qn,Qm,#270	Qd -> result	MVE
int16x8_t [arm_]vcaddq_rot270_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I16 Qd,Qn,Qm,#270	Qd -> result	MVE
int32x4_t [_arm_]vcaddq_rot270_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#270	Qd -> result	MVE
uint8x16_t [arm_]vcaddq_rot270_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I8 Qd,Qn,Qm,#270	Qd -> result	MVE
uint16x8_t [_arm_]vcaddq_rot270_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I16 Qd,Qn,Qm,#270	Qd -> result	MVE
uint32x4_t [_arm_]vcaddq_rot270_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCADDT.I32 Qd,Qn,Qm,#270	Qd -> result	MVE
float16x8_t [arm_]vcmlaq[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F16 Qda,Qn,Qm,#0	Qda -> result	MVE/NEON
float32x4_t [arm_]vcmlaq[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F32 Qda,Qn,Qm,#0	Qda -> result	MVE/NEON
float16x8_t [arm_]vcmlaq_rot90[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F16 Qda,Qn,Qm,#90	Qda -> result	MVE/NEON
float32x4_t [arm_]vcmlaq_rot90[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F32 Qda,Qn,Qm,#90	Qda -> result	MVE/NEON
float16x8_t [_arm_]vcmlaq_rot180[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F16 Qda,Qn,Qm,#180	Qda -> result	MVE/NEON
float32x4_t [arm_]vcmlaq_rot180[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F32 Qda,Qn,Qm,#180	Qda -> result	MVE/NEON
float16x8_t [arm_]vcmlaq_rot270[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F16 Qda,Qn,Qm,#270	Qda -> result	MVE/NEON
float32x4_t [arm_]vcmlaq_rot270[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VCMLA.F32 Qda,Qn,Qm,#270	Qda -> result	MVE/NEON
float16x8_t [arm_]vcmlaq_m[_f16](float16x8_t a, float16x8_t b, float16x8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VCMLAT.F16 Qda,Qn,Qm,#0	Qda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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	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)	p -> Rp a -> Qda b -> Qn c -> Qm	VMSR P0,Rp VPST VCMLAT.F16 Qda,Qn,Qm,#90	Qda -> result	MVE
float32x4_t [_arm_]vcmlaq_rot90_m[_f32](float32x4_t a, float32x4_t b, float32x4_t c, mve_pred16_t p)	p -> Rp a -> Qda b -> Qn c -> Qm	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)	p -> Rp a -> Qda b -> Qn c -> Qm	VMSR P0,Rp VPST VCMLAT.F16 Qda,Qn,Qm,#180	Qda -> result	MVE
$\label{eq:control_float_32x4_t} $$ float_32x_4_t = \underset{\text{L a, float_32x_4_t$ b, float_32x_4_t$ c, $mve_pred_16_t$ p)} $$$	p -> Rp 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	VMSR P0,Rp VPST VCMLAT.F16 Qda,Qn,Qm,#270	Qda -> result	MVE
$ \begin{array}{lll} float32x4_t \; [_arm_]vcmlaq_rot270_m[_f32](float32x4_t \\ a, float32x4_t \; b, float32x4_t \; c, \; mve_pred16_t \; p) \end{array} $	p -> Rp 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)	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
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, float16x8_t b)	a -> Qn b -> Qm	VCMUL.F16 Qd,Qn,Qm,#180	Qd -> result	MVE
float32x4_t [_arm_]vcmulq_rot180[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMUL.F32 Qd,Qn,Qm,#180	Qd -> result	MVE
float16x8_t [_arm_]vcmulq_rot270[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VCMUL.F16 Qd,Qn,Qm,#270	Qd -> result	MVE
float32x4_t [_arm_]vcmulq_rot270[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCMUL.F32 Qd,Qn,Qm,#270	Qd -> result	MVE
float16x8_t [arm_]vcmulq_m[_f16](float16x8_t inactive, 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,#0	Qd -> result	MVE
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 p -> Rp	VMSR P0,Rp VPST VCMULT.F16 Qd,Qn,Qm,#90	Qd -> result	MVE
$\label{eq:control_float_32x4_t} $$ float_32x_4_t = $$ m_vcmulq_rot_90_m[_f32](float_32x_4_t inactive, float_32x_4_t a, float_32x_4_t b, mve_pred_16_t p) $$$	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F32 Qd,Qn,Qm,#90	Qd -> result	MVE
float16x8_t [_arm_]vcmulq_rot180_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F16 Qd,Qn,Qm,#180	Qd -> result	MVE
$\label{eq:float32x4_t} float32x4_t \ [_arm_]vcmulq_rot180_m[_f32](float32x4_t \ inactive, float32x4_t \ a, float32x4_t \ b, mve_pred16_t \ p)$	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F32 Qd,Qn,Qm,#180	Qd -> result	MVE
float16x8_t [arm_]vcmulq_rot270_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMULT.F16 Qd,Qn,Qm,#270	Qd -> result	MVE
$\label{eq:control_float_32x4_t} $$[_arm_]vcmulq_rot270_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)$$	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VCMULT.F32 Qd,Qn,Qm,#270	Qd -> result	MVE
int8x16_t [arm_]vqabsq[_s8](int8x16_t a)	p -> Rp a -> Qm	VQABS.S8 Qd,Qm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vqabsq[_s16](int16x8_t a)	a -> Qm	VQABS.S16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vqabsq[_s32](int32x4_t a)	a -> Qm	VQABS.S32 Qd,Qm	Qd -> result	MVE/NEON
int8x16 t [arm]vgabsq m[s8](int8x16 t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm	VPST		·
	p -> Rp	VQABST.S8 Qd,Qm		
int16x8_t [arm_]vqabsq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
1 22 4 1 2 2 2 4 1 2	p -> Rp	VQABST.S16 Qd,Qm	0.1) (T) (T)
int32x4_t [_arm_]vqabsq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
mts2x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VOABST.S32 Od,Om		
int8x16_t [arm_]vqaddq[_n_s8](int8x16_t a, int8_t b)	a -> Qn	VQADD.S8 Qd,Qn,Rm	Od -> result	MVE
mionio_t [am_] (qaaq[_n_so](mionio_t a, mio_t o)	b -> Rm	, 4.122.50 &a,4.1,1	Qu' > Tesun	
int16x8_t [arm_]vqaddq[_n_s16](int16x8_t a, int16_t b)	a -> Qn	VQADD.S16 Qd,Qn,Rm	Qd -> result	MVE
	b -> Rm		-	
int32x4_t [arm_]vqaddq[_n_s32](int32x4_t a, int32_t b)	a -> Qn	VQADD.S32 Qd,Qn,Rm	Qd -> result	MVE
	b -> Rm			
uint8x16_t [arm_]vqaddq[_n_u8](uint8x16_t a, uint8_t	a -> Qn	VQADD.U8 Qd,Qn,Rm	Qd -> result	MVE
b)	b -> Rm	VOLDD VILCOLO D	0.1	Nam.
uint16x8_t [_arm_]vqaddq[_n_u16](uint16x8_t a,	a -> Qn	VQADD.U16 Qd,Qn,Rm	Qd -> result	MVE
uint16_t b) uint32x4_t [arm_]vqaddq[_n_u32](uint32x4_t a,	b -> Rm a -> On	VQADD.U32 Qd,Qn,Rm	Od -> result	MVE
uint32_t b)	a -> Qn b -> Rm	VQADD.U32 Qu,Qli,Rlli	Qu -> resuit	NIVE
int8x16 t [arm]vqaddq[s8](int8x16 t a, int8x16 t b)	a -> On	VQADD.S8 Qd,Qn,Qm	Od -> result	MVE/NEON
mtox10_t[arm_jvqaudq[_so](mtox10_t a, mtox10_t b)	b -> Qm	VQADD.58 Qu,Qii,Qiii	Qu => resuit	WIVE/INDOIN
int16x8_t [arm_]vqaddq[_s16](int16x8_t a, int16x8_t b)	a -> On	VQADD.S16 Qd,Qn,Qm	Od -> result	MVE/NEON
	b -> Qm	. (Q	
int32x4_t [arm_]vqaddq[_s32](int32x4_t a, int32x4_t b)	a -> Qn	VQADD.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
	b -> Qm			
uint8x16_t [arm_]vqaddq[_u8](uint8x16_t a, uint8x16_t	a -> Qn	VQADD.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
b)	b -> Qm			
uint16x8_t [arm_]vqaddq[_u16](uint16x8_t a,	a -> Qn	VQADD.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t b)	b -> Qm	VO.1 P.D. V. V. O.1 O. O.	0.1	MEATON
uint32x4_t [arm_]vqaddq[_u32](uint32x4_t a,	a -> Qn	VQADD.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t b) int8x16_t [arm_]vqaddq_m[_n_s8](int8x16_t inactive,	b -> Qm inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int8x16_t a, int8_t b, mve_pred16_t p)	a -> On	VPST	Qu -> resurt	WYL
intoxro_t a, into_t b, inve_prearo_t p)	b -> Rm	VQADDT.S8 Qd,Qn,Rm		
	p -> Rp			
int16x8_t [arm_]vqaddq_m[_n_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm	VQADDT.S16 Qd,Qn,Rm		
	p -> Rp			
int32x4_t [_arm_]vqaddq_m[_n_s32](int32x4_t inactive,	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t a, int32_t b, mve_pred16_t p)	a -> QII b -> Rm	VQADDT.S32 Qd,Qn,Rm		
	p -> Rn	VQADD1:332 Qu,Qii,Kiii		
uint8x16 t [arm]vgaddg m[n u8](uint8x16 t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn	VPST	Q	
· - · - · - · - · - · - · · - · · ·	b -> Rm	VQADDT.U8 Qd,Qn,Rm		
	p -> Rp			
uint16x8_t [arm_]vqaddq_m[_n_u16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Rm	VQADDT.U16 Qd,Qn,Rm		
uint32x4_t [arm_]vqaddq_m[_n_u32](uint32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	a -> Qu	VMSK PO,KP VPST	Qu -> resuit	NIVE
mactive, unit32x4_t a, unit32_t b, mve_pred10_t p)	b -> Rm	VOADDT.U32 Od,On,Rm		
	p -> Rp	. (
int8x16_t [arm_]vqaddq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VQADDT.S8 Qd,Qn,Qm		
	p -> Rp	12 102 Po -	<u> </u>	1
int16x8_t [_arm_]vqaddq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm 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 -> Qu	VMSK FO,KP VPST	Za -> resuit	11111
	b -> Qm	VQADDT.S32 Qd,Qn,Qm		
	p -> Rp			
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	VPST		
	b -> Qm	VQADDT.U8 Qd,Qn,Qm		
	p -> Rp			

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [_arm_]vqrdmladhxq[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b)	inactive -> Qd a -> Qn b -> Qm	VQRDMLADHX.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqrdmladhxq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLADHXT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqrdmladhxq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLADHXT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vqrdmladhxq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLADHXT.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqdmlahq[_n_s8](int8x16_t a, int8x16_t b, int8_t c)	a -> Qda b -> Qn c -> Rm	VQDMLAH.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [_arm_]vqdmlahq[_n_s16](int16x8_t a, int16x8_t b, int16_t c)	a -> Qda b -> Qn c -> Rm	VQDMLAH.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vqdmlahq[_n_s32](int32x4_t a, int32x4_t b, int32_t c)	a -> Qda b -> Qn c -> Rm	VQDMLAH.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vqdmlahq[_n_u8](uint8x16_t a, uint8x16_t b, uint8_t c)	a -> Qda b -> Qn c -> Rm	VQDMLAH.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [_arm_]vqdmlahq[_n_u16](uint16x8_t a, uint16x8_t b, uint16_t c)	a -> Qda b -> Qn c -> Rm	VQDMLAH.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vqdmlahq[_n_u32](uint32x4_t a, uint32x4_t b, uint32_t c)	a -> Qda b -> Qn c -> Rm	VQDMLAH.U32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [arm_]vqdmlahq_m[_n_s8](int8x16_t a, int8x16_t b, int8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLAHT.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vqdmlahq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLAHT.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vqdmlahq_m[_n_s32](int32x4_t a, int32x4_t b, int32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLAHT.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vqdmlahq_m[_n_u8](uint8x16_t a, uint8x16_t b, uint8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLAHT.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [_arm_]vqdmlahq_m[_n_u16](uint16x8_t a, uint16x8_t b, uint16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLAHT.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vqdmlahq_m[_n_u32](uint32x4_t a, uint32x4_t b, uint32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLAHT.U32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [arm_]vqrdmlahq[_n_s8](int8x16_t a, int8x16_t b, int8_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLAH.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [arm_]vqrdmlahq[_n_s16](int16x8_t a, int16x8_t b, int16_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLAH.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vqrdmlahq[_n_s32](int32x4_t a, int32x4_t b, int32_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLAH.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [arm_]vqrdmlahq[_n_u8](uint8x16_t a, uint8x16_t b, uint8_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLAH.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [arm_]vqrdmlahq[_n_u16](uint16x8_t a, uint16x8_t b, uint16_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLAH.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vqrdmlahq[_n_u32](uint32x4_t a, uint32x4_t b, uint32_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLAH.U32 Qda,Qn,Rm	Qda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vqrdmlahq_m[_n_s8](int8x16_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int8x16_t b, int8_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Rm	VQRDMLAHT.S8 Qda,Qn,Rm		
:	p -> Rp	VMCD DO D.	0.116	MVE
int16x8_t [arm_]vqrdmlahq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> On	VMSR P0,Rp VPST	Qda -> result	MVE
mirrox8_t b, mirro_t c, mve_pred16_t p)	c -> Qn	VPS1 VQRDMLAHT.S16 Qda,Qn,Rm		
	p -> Rp	VQKDWLAII1.510 Qua,Qii,Kiii		
int32x4 t [arm]vqrdmlahq m[n s32](int32x4 t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32x4_t b, int32_t c, mve_pred16_t p)	b -> Qn	VPST		
•	c -> Rm	VQRDMLAHT.S32 Qda,Qn,Rm		
	p -> Rp			
uint8x16_t [arm_]vqrdmlahq_m[_n_u8](uint8x16_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint8x16_t b, uint8_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Rm	VQRDMLAHT.U8 Qda,Qn,Rm		
uint16x8_t [arm_]vqrdmlahq_m[_n_u16](uint16x8_t a,	p -> Rp a -> Qda	VMSR P0,Rp	Oda -> result	MVE
uint16x8_t b, uint16_t c, mve_pred16_t p)	b -> Qua	VMSK PO,RP VPST	Qua -> resuit	IVI V E
unitroxo_t b, unitro_t c, inve_preuro_t p)	c -> Rm	VQRDMLAHT.U16 Qda,Qn,Rm		
	p -> Rp	V QRENIER IIII. O TO Quiu, Qii, Riii		
uint32x4_t [arm_]vqrdmlahq_m[_n_u32](uint32x4_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint32x4_t b, uint32_t c, mve_pred16_t p)	b -> Qn	VPST		
_ · · _ · _ · _ •	c -> Rm	VQRDMLAHT.U32 Qda,Qn,Rm		
	p -> Rp			
int8x16_t [arm_]vqrdmlashq[_n_s8](int8x16_t a,	a -> Qda	VQRDMLASH.S8 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t b, int8_t c)	b -> Qn			
1.160.16	c -> Rm	VODDAM LOVIGICAL O. D.	0.1) am
int16x8_t [arm_]vqrdmlashq[_n_s16](int16x8_t a,	a -> Qda	VQRDMLASH.S16 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t b, int16_t c)	b -> Qn c -> Rm			
int32x4_t [arm_]vqrdmlashq[_n_s32](int32x4_t a,	a -> Qda	VQRDMLASH.S32 Qda,Qn,Rm	Oda -> result	MVE
int32x4_t b, int32_t c)	b -> On	V Q RDWILL ISTI. IS 32 Qua, QII, RIII	Qua -> resuit	MIVE
mcszkije o, mcszet c)	c -> Rm			
uint8x16_t [arm_]vqrdmlashq[_n_u8](uint8x16_t a,	a -> Oda	VQRDMLASH.U8 Qda,Qn,Rm	Oda -> result	MVE
uint8x16_t b, uint8_t c)	b -> Qn			
	c -> Rm			
uint16x8_t [arm_]vqrdmlashq[_n_u16](uint16x8_t a,	a -> Qda	VQRDMLASH.U16 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t b, uint16_t c)	b -> Qn			
	c -> Rm			
uint32x4_t [arm_]vqrdmlashq[_n_u32](uint32x4_t a,	a -> Qda	VQRDMLASH.U32 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t b, uint32_t c)	b -> Qn c -> Rm			
int8x16_t [arm_]vqrdmlashq_m[_n_s8](int8x16_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int8x16_t b, int8_t c, mve_pred16_t p)	b -> Qua	VPST	Qua -> resuit	WIVE
	c -> Rm	VORDMLASHT.S8 Qda,Qn,Rm		
	p -> Rp			
int16x8_t [arm_]vqrdmlashq_m[_n_s16](int16x8_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int16x8_t b, int16_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Rm	VQRDMLASHT.S16		
	p -> Rp	Qda,Qn,Rm		
int32x4_t [arm_]vqrdmlashq_m[_n_s32](int32x4_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32x4_t b, int32_t c, mve_pred16_t p)	b -> Qn	VPST VQRDMLASHT.S32		
	c -> Rm	_		
uint8x16_t [arm_]vqrdmlashq_m[_n_u8](uint8x16_t a,	p -> Rp a -> Qda	Qda,Qn,Rm VMSR P0,Rp	Qda -> result	MVE
uint8x16 t b, uint8 t c, mve pred16 t p)	b -> Qua	VMSK FO,KP VPST	Qua -> resuit	IVI V L
unitox10_t b, unito_t e, inve_preu10_t p)	c -> Rm	VQRDMLASHT.U8 Qda,Qn,Rm		
	p -> Rp	, 611211212111100 61111		
uint16x8_t [arm_]vqrdmlashq_m[_n_u16](uint16x8_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint16x8_t b, uint16_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Rm	VQRDMLASHT.U16		
	p -> Rp	Qda,Qn,Rm		
uint32x4_t [_arm_]vqrdmlashq_m[_n_u32](uint32x4_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint32x4_t b, uint32_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Rm	VQRDMLASHT.U32		
int8x16_t [arm_]vqdmlsdhq[_s8](int8x16_t inactive,	p -> Rp inactive -> Qd	Qda,Qn,Rm VQDMLSDH.S8 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [armjvqdmisdnq[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b)	a -> Qu	V QDIVILODIT.36 QU,QII,QIII	Qu -> resuit	IVI V I
moxio_t a, moxio_t b)	b -> Qm			
int16x8_t [_arm_]vqdmlsdhq[_s16](int16x8_t inactive,	inactive -> Qd	VQDMLSDH.S16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t a, int16x8_t b)	a -> Qn	. 22222	Za z rosuit	1 2
	b -> Qm			
	inactive -> Qd	VQDMLSDH.S32 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vqdmlsdhq[_s32](int32x4_t inactive,	mactive -> Qu	V QDIVIESDI1.552 Qu,Qii,Qiii	Qu / Tobuit	141 4 12
int32x4_t [_arm_]vqdmlsdhq[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b)	a -> Qn	V QDMLSD11.532 Qu,Qii,Qiii	Qu y resuit	W L

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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 p -> Rp	VMSR P0,Rp VPST VQDMLSDHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vqdmlsdhq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHT.S32 Qd,Qn,Qm	Qd -> result	MVE
$int8x16_t \ [_arm_]vqdmlsdhxq[_s8] (int8x16_t \ inactive, \\ int8x16_t \ a, int8x16_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLSDHX.S8 Qd,Qn,Qm	Qd -> result	MVE
$int16x8_t \ [_arm_]vqdmlsdhxq[_s16] (int16x8_t \ inactive, \\ int16x8_t \ a, int16x8_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLSDHX.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4_t \ [_arm_]vqdmlsdhxq[_s32](int32x4_t \ inactive, \\ int32x4_t \ a, int32x4_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQDMLSDHX.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]vqdmlsdhxq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHXT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vqdmlsdhxq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHXT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vqdmlsdhxq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHXT.S32 Qd,Qn,Qm	Qd -> result	MVE
$int8x16_t \ [_arm_] vqrdmlsdhq[_s8] (int8x16_t \ inactive, \\ int8x16_t \ a, int8x16_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDH.S8 Qd,Qn,Qm	Qd -> result	MVE
$int16x8_t \ [_arm_] vqrdmlsdhq[_s16] (int16x8_t \ inactive, \\ int16x8_t \ a, int16x8_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDH.S16 Qd,Qn,Qm	Qd -> result	MVE
$int32x4_t \ [_arm_]vqrdmlsdhq[_s32](int32x4_t \ inactive, \\ int32x4_t \ a, int32x4_t \ b)$	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDH.S32 Qd,Qn,Qm	Qd -> result	MVE
$int8x16_t \ [_arm_] vqrdmlsdhq_m[_s8] (int8x16_t \ inactive, \\ int8x16_t \ a, int8x16_t \ b, mve_pred16_t \ p)$	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLSDHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqrdmlsdhq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLSDHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [_arm_]vqrdmlsdhq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm	VMSR P0,Rp VPST VQRDMLSDHT.S32 Qd,Qn,Qm	Qd -> result	MVE
$int8x16_t \ [_arm_] vqrdmlsdhxq[_s8] (int8x16_t \ inactive, \\ int8x16_t \ a, int8x16_t \ b)$	p -> Rp inactive -> Qd a -> Qn b -> Qm	VQRDMLSDHX.S8 Qd,Qn,Qm	Qd -> result	MVE
$int16x8_t \ [_arm_] vqrdmlsdhxq[_s16] (int16x8_t \ inactive, \\ int16x8_t \ a, int16x8_t \ b)$	inactive -> Qd a -> Qn b -> Om	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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, miox10_t a, mio_t b, mvo_pred10_t p)	b -> Rm p -> Rp	VQDMULHT.S8 Qd,Qn,Rm		
int16x8_t [_arm_]vqdmulhq_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
mactive, inclose_c a, inclo_c o, inve_predio_c p)	b -> Rm p -> Rp	VQDMULHT.S16 Qd,Qn,Rm		
int32x4_t [_arm_]vqdmulhq_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
mactive, mis2x4_t a, mis2_t b, mve_pred10_t p)	b -> Rm p -> Rp	VQDMULHT.S32 Qd,Qn,Rm		
int8x16_t [arm_]vqdmulhq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VQDMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vqdmulhq[_s16](int16x8_t a, int16x8_t	a -> Qn b -> Qm	VQDMULH.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
b) int32x4_t [_arm_]vqdmulhq[_s32](int32x4_t a, int32x4_t	a -> Qm b -> Om	VQDMULH.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
b) int8x16_t [_arm_]vqdmulhq_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 VQDMULHT.S8 Qd,Qn,Qm		
int16x8_t [_arm_]vqdmulhq_m[_s16](int16x8_t inactive,	p -> Rp inactive -> Qd a -> On	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	b -> Qm	VPST VQDMULHT.S16 Qd,Qn,Qm		
int32x4_t [_arm_]vqdmulhq_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 b -> Qm	VPST VQDMULHT.S32 Qd,Qn,Qm		
int8x16_t [_arm_]vqrdmulhq[_n_s8](int8x16_t a, int8_t	p -> Rp a -> Qn b > Pm	VQRDMULH.S8 Qd,Qn,Rm	Qd -> result	MVE
b) int16x8_t [_arm_]vqrdmulhq[_n_s16](int16x8_t a,	b -> Rm a -> Qn	VQRDMULH.S16 Qd,Qn,Rm	Qd -> result	MVE/NEON
int16_t b) int32x4_t [_arm_]vqrdmulhq[_n_s32](int32x4_t a,	b -> Rm a -> Qn	VQRDMULH.S32 Qd,Qn,Rm	Qd -> result	MVE/NEON
int32_t b) int8x16_t [arm_]vqrdmulhq_m[_n_s8](int8x16_t	b -> Rm inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VQRDMULHT.S8 Qd,Qn,Rm		
int16x8_t [_arm_]vqrdmulhq_m[_n_s16](int16x8_t	p -> Rp inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
inactive, int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm	VQRDMULHT.S16 Qd,Qn,Rm		
int32x4_t [_arm_]vqrdmulhq_m[_n_s32](int32x4_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm	VPST VQRDMULHT.S32 Qd,Qn,Rm		
int8x16_t [arm_]vqrdmulhq[_s8](int8x16_t a, int8x16_t	p -> Rp a -> Qn b > Om	VQRDMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
b) int16x8_t [_arm_]vqrdmulhq[_s16](int16x8_t a, int16x8_t b)	b -> Qm a -> Qn b > Om	VQRDMULH.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vqrdmulhq[_s32](int32x4_t a,	b -> Qm a -> Qn	VQRDMULH.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t b) int8x16_t [_arm_]vqrdmulhq_m[_s8](int8x16_t inactive,	b -> Qm inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm	VPST VQRDMULHT.S8 Qd,Qn,Qm		
int16x8_t [_arm_]vqrdmulhq_m[_s16](int16x8_t	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VPST VQRDMULHT.S16 Qd,Qn,Qm		
int32x4_t [_arm_]vqrdmulhq_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 VQRDMULHT.S32 Qd,Qn,Qm		
int32x4_t [_arm_]vqdmullbq[_n_s16](int16x8_t a, int16_t b)	p -> Rp a -> Qn b -> Rm	VQDMULLB.S16 Qd,Qn,Rm	Qd -> result	MVE
int64x2_t [arm_]vqdmullbq[_n_s32](int32x4_t a,	a -> Qn	VQDMULLB.S32 Qd,Qn,Rm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [arm_]vqdmullbq_m[_n_s16](int32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VPST VQDMULLBT.S16 Qd,Qn,Rm		
int64x2_t [arm_]vqdmullbq_m[_n_s32](int64x2_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VPST VQDMULLBT.S32 Qd,Qn,Rm		
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 -> 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 VQDMULLBT.S32 Qd,Qn,Qm		
int32x4_t [arm_]vqdmulltq[_n_s16](int16x8_t a, int16_t	a -> Qn	VQDMULLT.S16 Qd,Qn,Rm	Qd -> result	MVE
b) int64x2_t [_arm_]vqdmulltq[_n_s32](int32x4_t a, int32_t	b -> Rm a -> Qn	VQDMULLT.S32 Qd,Qn,Rm	Qd -> result	MVE
b) int32x4_t [arm_]vqdmulltq_m[_n_s16](int32x4_t	b -> Rm inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VPST VQDMULLTT.S16 Qd,Qn,Rm		
int64x2_t [_arm_]vqdmulltq_m[_n_s32](int64x2_t	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
inactive, int32x4_t a, int32_t b, mve_pred16_t p)	b -> Rm p -> Rp	VPS1 VQDMULLTT.S32 Qd,Qn,Rm		
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,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VPST VQDMULLTT.S16 Qd,Qn,Qm		
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	VMSR P0,Rp VPST VQDMULLTT.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vqnegq[_s8](int8x16_t a)	p -> Rp a -> Om	VONEG.S8 Od,Om	Qd -> result	MVE/NEON
int16x8_t [arm_]vqnegq[_s16](int16x8_t a)	a -> Qm	VQNEG.S16 Qd,Qm	Od -> result	MVE/NEON MVE/NEON
int32x4_t [arm_]vqnegq[_s32](int32x4_t a)	a -> Qm	VQNEG.S32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vqnegq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VQNEGT.S8 Qd,Qm	Qd -> result	MVE
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,	p -> Rp inactive -> Qd	VQNEGT.S16 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VQNEGT.S32 Qd,Qm		
int8x16_t [arm_]vqsubq[_n_s8](int8x16_t a, int8_t b)	a -> Qn	VQSUB.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vqsubq[_n_s16](int16x8_t a, int16_t b)	b -> Rm 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	a -> Qn	VQSUB.U8 Qd,Qn,Rm	Qd -> result	MVE
b) uint16x8_t [arm_]vqsubq[_n_u16](uint16x8_t a, uint16_t b)	b -> Rm a -> Qn b -> Rm	VQSUB.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vqsubq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VQSUB.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]vqsubq_m[_n_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VPST VQSUBT.S8 Qd,Qn,Rm		
int16x8_t [_arm_]vqsubq_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 p -> Rp	VPST VQSUBT.S16 Qd,Qn,Rm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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	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)	p -> Rp 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 p -> Rp	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)	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
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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] 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] 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] 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
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] 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
	base -> Rn	VLDRB.S32 Qd,[Rn]	Qd -> result	MVE
int32x4_t [_arm_]vldrbq_s32(int8_t const * base) uint8x16_t [_arm_]vldrbq_u8(uint8_t const * base)	base -> Rn	VLDRB.8 Qd,[Rn]	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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	Qd -> result	MVE
int16x8_t [_arm_]vldrbq_z_s16(int8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VLDRBT.8 Qd,[Rn] 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	VLDRBT.S10 Qd,[Rn] 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	VLDRBT.U16 Qd,[Rn] 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	Qd -> result	MVE
int16x8_t [arm_]vldrhq_s16(int16_t const * base) int32x4_t [arm_]vldrhq_s32(int16_t const * base)	base -> Rn base -> Rn	VLDRBT.U32 Qd,[Rn] VLDRH.16 Qd,[Rn] VLDRH.S32 Qd,[Rn]	Qd -> result Qd -> result	MVE MVE
uint16x8_t [_arm_]vldrhq_u16(uint16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Od -> 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
uint32x4_t [_arm_]vldrwq_u32(uint32_t const * base)	base -> Rn	VLDRW.32 Qd,[Rn]	Qd -> result	MVE
float32x4_t [arm_]vldrwq_f32(float32_t const * base)	base -> Rn	VLDRW.32 Qd,[Rn]	Qd -> result	MVE
int32x4_t [_arm_]vldrwq_z_s32(int32_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRWT.32 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [_arm_]vldrwq_z_u32(uint32_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRWT.32 Qd,[Rn]	Qd -> result	MVE
float32x4_t [_arm_]vldrwq_z_f32(float32_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST	Qd -> result	MVE
int8x16_t [arm_]vld1q[_s8](int8_t const * base)	base -> Rn	VLDRWT.32 Qd,[Rn] VLDRB.8 Qd,[Rn]	Qd -> result	MVE/NEON
int16x8_t [arm_]vld1q[_s16](int16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE/NEON
int32x4_t [_arm_]vld1q[_s32](int32_t const * base)	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]	Qd -> result	MVE/NEON
uint32x4_t [arm_]vld1q[_u32](uint32_t const * base) float16x8_t [arm_]vld1q[_f16](float16_t const * base)	base -> Rn base -> Rn	VLDRW.32 Qd,[Rn] VLDRH.16 Qd,[Rn]	Qd -> result Qd -> result	MVE/NEON MVE/NEON
float32x4_t [_arm_]vld1q[_f16](float32_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn] VLDRW.32 Qd,[Rn]	Qd -> result	MVE/NEON
int16x8_t [arm_]vldrhq_gather_offset[_s16](int16_t const * base, uint16x8_t offset)	base -> Rn offset -> Qm	VLDRH.U16 Qd,[Rn,Qm]	Qd -> result	MVE
int32x4_t [_arm_]vldrhq_gather_offset[_s32](int16_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRH.S32 Qd,[Rn,Qm]	Qd -> result	MVE
uint16x8_t [_arm_]vldrhq_gather_offset[_u16](uint16_t const * base, uint16x8_t offset) uint32x4_t [_arm_]vldrhq_gather_offset[_u32](uint16_t	base -> Rn offset -> Qm base -> Rn	VLDRH.U16 Qd,[Rn,Qm] VLDRH.U32 Qd,[Rn,Qm]	Qd -> result Od -> result	MVE MVE
const * base, uint32x4_t offset) float16x8_t [_arm_]vldrhq_gather_offset[_fl6](float16_t	offset -> Rn offset -> Qm base -> Rn	VLDRH.U32 Qd,[Rn,Qm] VLDRH.F16 Qd,[Rn,Qm]	Qd -> result Qd -> result	MVE
const * base, uint16x8_t offset)	offset -> Qm			
int16x8_t [_arm_]vldrhq_gather_offset_z[_s16](int16_t const * base, uint16x8_t offset, mve_pred16_t p)	base -> Rn offset -> Qm	VMSR P0,Rp VPST VLDPHT U16 Od [Pn Om]	Qd -> result	MVE
int32x4_t [_arm_]vldrhq_gather_offset_z[_s32](int16_t	p -> Rp base -> Rn	VLDRHT.U16 Qd,[Rn,Qm] VMSR P0,Rp	Qd -> result	MVE
const * base, uint32x4_t offset, mve_pred16_t p)	offset -> Qm p -> Rp	VPST VLDRHT.S32 Qd,[Rn,Qm]	Qu -> resun	IVI V L

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [_arm_]vldrwq_gather_offset[_u32](uint32_t const * base, uint32x4_t offset)	base -> Rn offset -> Om	VLDRW.U32 Qd,[Rn,Qm]	Qd -> result	MVE
float32x4_t [_arm_]vldrwq_gather_offset[_f32](float32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRW.U32 Qd,[Rn,Qm]	Qd -> result	MVE
int32x4_t [_arm_]vldrwq_gather_offset_z[_s32](int32_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
uint32x4 t	p -> Rp base -> Rn	VLDRWT.U32 Qd,[Rn,Qm] VMSR P0,Rp	Od -> result	MVE
[arm_]vldrwq_gather_offset_z[_u32](uint32_t const * base, uint32x4_t offset, mve_pred16_t p)	$\begin{array}{c} \text{offset -> Qm} \\ \text{p -> Rp} \end{array}$	VPST VLDRWT.U32 Qd,[Rn,Qm]	`	
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 offset/ uint32x4_t [_arm_]vldrwq_gather_shifted_offset[_u32](uint32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRW.U32 Qd,[Rn,Qm,UXTW #2]	Qd -> result	MVE
float32x4_t (arm_]vldrwq_gather_shifted_offset[_f32](float32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRW.U32 Qd,[Rn,Qm,UXTW #2]	Qd -> result	MVE
int32x4_t offset) 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]	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)	p -> Rp *addr -> Qn offset in +/- 4*[0127] p -> Rp	VMSR P0,Rp VPST VLDRWT.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64x2_t [_arm_]vldrdq_gather_offset[_s64](int64_t	base -> Rn	VLDRD.U64 Qd,[Rn,Qm]	Qd -> result	MVE
const * base, uint64x2_t offset) uint64x2_t [_arm_]vldrdq_gather_offset[_u64](uint64_t const * base, uint64x2_t offset)	offset -> Qm base -> Rn offset -> Qm	VLDRD.U64 Qd,[Rn,Qm]	Qd -> result	MVE
int64x2_t [_arm_]vldrdq_gather_offset_z[_s64](int64_t const * base, uint64x2_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRDT.U64 Qd,[Rn,Qm]	Qd -> result	MVE
uint64x2_t [_arm_]vldrdq_gather_offset_z[_u64](uint64_t const * base, uint64x2_t offset, mve_pred16_t p)	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_eather_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
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] ->	VST20.32 {Qd - Qd2},[Rn] VST21.32 {Qd - Qd2},[Rn]	void -> result	MVE
void [_arm_]vst2q[_u8](uint8_t * addr, uint8x16x2_t value)	Qd2 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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
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] -> 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[_s32](int32_t * addr, int32x4x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.32 {Qd - Qd4},[Rn] VST41.32 {Qd - Qd4},[Rn] VST42.32 {Qd - Qd4},[Rn] VST43.32 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_u8](uint8_t * addr, uint8x16x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.8 {Qd - Qd4},[Rn] VST41.8 {Qd - Qd4},[Rn] VST42.8 {Qd - Qd4},[Rn] VST43.8 {Qd - Qd4},[Rn]	void -> result	MVE
void [_arm_]vst4q[_u16](uint16_t * addr, uint16x8x4_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2 value.val[2] -> Qd3 value.val[3] -> Qd4	VST40.16 {Qd - Qd4},[Rn] VST41.16 {Qd - Qd4},[Rn] VST42.16 {Qd - Qd4},[Rn] 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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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] ->	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)	Qd4 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
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 p -> Rp	VMSR P0,Rp VPST VSTRBT.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrbq_p[_s32](int8_t * base, int32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRBT.32 Qd,[Rn]	void -> result	MVE
void [arm_]vstrbq_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_]vstrbq_p[_u16](uint8_t * base, uint16x8_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRBT.16 Qd,[Rn]	void -> result	MVE
void [arm_]vstrbq_p[_u32](uint8_t * base, uint32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	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 p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrhq_p[_u32](uint16_t * base, uint32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn]	void -> result	MVE
$\label{eq:condition} $$ void [_arm_] vstrhq_p[_f16](float16_t * base, float16x8_t value, mve_pred16_t p) $$$	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrwq[_s32](int32_t * base, int32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrwq[_u32](uint32_t * base, uint32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrwq[_f32](float32_t * base, float32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE
void [_arm_]vstrwq_p[_s32](int32_t * base, int32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn]	void -> result	MVE
$\label{eq:condition} $$ void [_arm_] vstrwq_p[_u32](uint32_t * base, uint32x4_t value, mve_pred16_t p) $$$	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn]	void -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void [_arm_]vstrwq_p[_f32](float32_t * base,	base -> Rn	VMSR P0,Rp	void -> result	MVE
float32x4_t value, mve_pred16_t p)	value -> Qd	VPST		
void [arm]vst1q[s8](int8 t * base, int8x16 t value)	p -> Rp base -> Rn	VSTRWT.32 Qd,[Rn] VSTRB.8 Qd,[Rn]	void -> result	MVE/NEON
	value -> Qd			
void [_arm_]vst1q[_s16](int16_t * base, int16x8_t value)	base -> Rn value -> Qd	VSTRH.16 Qd,[Rn]	void -> result	MVE/NEON
void [_arm_]vst1q[_s32](int32_t * base, int32x4_t value)	base -> Rn	VSTRW.32 Qd,[Rn]	void -> result	MVE/NEON
	value -> Qd base -> Rn	VSTRB.8 Qd,[Rn]	void -> result	MVE/NEON
void [arm_]vst1q[_u8](uint8_t * base, uint8x16_t value)	value -> Qd	VSTRB.8 Qu,[RII]	void -> resuit	IVI V E/INEOIN
void [_arm_]vst1q[_u16](uint16_t * base, uint16x8_t value)	base -> Rn	VSTRH.16 Qd,[Rn]	void -> result	MVE/NEON
void [_arm_]vst1q[_u32](uint32_t * base, uint32x4_t	value -> Qd base -> Rn	VSTRW.32 Qd,[Rn]	void -> result	MVE/NEON
value)	value -> Qd	VCTDII 1 CO 1 CD 1		MENEON
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	base -> Rn	VSTRW.32 Qd,[Rn]	void -> result	MVE/NEON
value) void [_arm_]vstrbq_scatter_offset[_s8](int8_t * base,	value -> Qd base -> Rn	VSTRB.8 Qd,[Rn,Qm]	void -> result	MVE
uint8x16_t offset, int8x16_t value)	offset -> Qm			
void [_arm_]vstrbq_scatter_offset[_s16](int8_t * base,	value -> Qd base -> Rn	VSTRB.16 Qd,[Rn,Qm]	void -> result	MVE
uint16x8_t offset, int16x8_t value)	offset -> Qm	vo monto qui,[mi,qm]	void > Tesuit	11112
void [_arm_]vstrbq_scatter_offset[_s32](int8_t * base,	value -> Qd base -> Rn	VSTRB.32 Qd,[Rn,Qm]	void -> result	MVE
uint32x4_t offset, int32x4_t value)	offset -> Qm	VOTRB.32 Qu,[Rii,Qiii]	void > result	141 4 12
void [_arm_]vstrbq_scatter_offset[_u8](uint8_t * base,	value -> Qd base -> Rn	VSTRB.8 Qd,[Rn,Qm]	void -> result	MVE
uint8x16_t offset, uint8x16_t value)	offset -> Qm	V51Kb.0 Qu,[Kii,Qiii]	void -> resurt	IVI V L
void [_arm_]vstrbq_scatter_offset[_u16](uint8_t * base,	value -> Qd base -> Rn	VSTRB.16 Qd,[Rn,Qm]	void -> result	MVE
uint16x8_t offset, uint16x8_t value)	offset -> Qm	VSTRB.10 Qu,[Kii,Qiii]	void => resuit	IVI V E
void [arm_]vstrbq_scatter_offset[_u32](uint8_t * base,	value -> Qd base -> Rn	VSTRB.32 Qd,[Rn,Qm]	void -> result	MVE
uint32x4_t offset, uint32x4_t value)	offset -> Qm	VSTRB.32 Qu,[RII,QIII]	void -> resuit	MVE
.11	value -> Qd	VACD DO D		NOTE
void [_arm_]vstrbq_scatter_offset_p[_s8](int8_t * base, uint8x16_t offset, int8x16_t value, mve_pred16_t p)	base -> Rn offset -> Qm	VMSR P0,Rp VPST	void -> result	MVE
	value -> Qd	VSTRBT.8 Qd,[Rn,Qm]		
void [arm_]vstrbq_scatter_offset_p[_s16](int8_t * base,	p -> Rp base -> Rn	VMSR P0,Rp	void -> result	MVE
uint16x8_t offset, int16x8_t value, mve_pred16_t p)	offset -> Qm	VPST		
	value -> Qd p -> Rp	VSTRBT.16 Qd,[Rn,Qm]		
void [_arm_]vstrbq_scatter_offset_p[_s32](int8_t * base, uint32x4_t offset, int32x4_t value, mve_pred16_t p)	base -> Rn	VMSR P0,Rp	void -> result	MVE
umi32x4_t offset, mt32x4_t value, mve_pred16_t p)	offset -> Qm value -> Qd	VPST VSTRBT.32 Qd,[Rn,Qm]		
.11	p -> Rp	VACO DO D		NOTE
void [arm_]vstrbq_scatter_offset_p[_u8](uint8_t * base, uint8x16_t offset, uint8x16_t value, mve_pred16_t p)	base -> Rn offset -> Qm	VMSR P0,Rp VPST	void -> result	MVE
	value -> Qd	VSTRBT.8 Qd,[Rn,Qm]		
void [_arm_]vstrbq_scatter_offset_p[_u16](uint8_t *	p -> Rp base -> Rn	VMSR P0,Rp	void -> result	MVE
base, uint16x8_t offset, uint16x8_t value, mve_pred16_t p)	offset -> Qm	VPST		
	value -> Qd p -> Rp	VSTRBT.16 Qd,[Rn,Qm]		
void [_arm_]vstrbq_scatter_offset_p[_u32](uint8_t *	base -> Rn	VMSR P0,Rp	void -> result	MVE
base, uint32x4_t offset, uint32x4_t value, mve_pred16_t p)	offset -> Qm value -> Qd	VPST VSTRBT.32 Qd,[Rn,Qm]		
	p -> Rp	Vompy 1 c 0 1 m 0 1) am
void [_arm_]vstrhq_scatter_offset[_s16](int16_t * base, uint16x8_t offset, int16x8_t value)	base -> Rn offset -> Qm	VSTRH.16 Qd,[Rn,Qm]	void -> result	MVE
	value -> Qd	Mombilias of the con-		NOW
void [_arm_]vstrhq_scatter_offset[_s32](int16_t * base, uint32x4_t offset, int32x4_t value)	base -> Rn offset -> Qm	VSTRH.32 Qd,[Rn,Qm]	void -> result	MVE
	value -> Qd	Mombili 10 0 1 th o 3	., .	NOTE
void [_arm_]vstrhq_scatter_offset[_u16](uint16_t * base, uint16x8_t offset, uint16x8_t value)	base -> Rn offset -> Qm	VSTRH.16 Qd,[Rn,Qm]	void -> result	MVE
	value -> Qd	YYOMDYY AC CASE		1000
void [_arm_]vstrhq_scatter_offset[_u32](uint16_t * base, uint32x4_t offset, uint32x4_t value)	base -> Rn offset -> Qm	VSTRH.32 Qd,[Rn,Qm]	void -> result	MVE
	value -> Qd	YYOMDYY 1 - 0 - 1 - 0		1000
void [_arm_]vstrhq_scatter_offset[_f16](float16_t * base, uint16x8_t offset, float16x8_t value)	base -> Rn offset -> Qm	VSTRH.16 Qd,[Rn,Qm]	void -> result	MVE
_ · · · · · · · · · · · · · · · · · · ·	value -> Qd			

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
void [_arm_]vstrdq_scatter_base[_s64](uint64x2_t addr, const int offset, int64x2_t value)	addr -> Qn offset in +/- 8*[0127] value -> Qd	VSTRD.U64 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrdq_scatter_base[_u64](uint64x2_t addr, const int offset, uint64x2_t value)	addr -> Qn offset in +/- 8*[0127] value -> Qd	VSTRD.U64 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrdq_scatter_base_p[_s64](uint64x2_t addr, const int offset, int64x2_t value, mve_pred16_t p)	addr -> Qn offset in +/- 8*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrdq_scatter_base_p[_u64](uint64x2_t addr, const int offset, uint64x2_t value, mve_pred16_t p)	addr -> Qn offset in +/- 8*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]	void -> result	MVE
void [_arm_]vstrdq_scatter_base_wb[_s64](uint64x2_t * addr, const int offset, int64x2_t value)	*addr -> Qn offset in +/- 8*[0127] value -> Qd	VSTRD.U64 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrdq_scatter_base_wb[_u64](uint64x2_t * addr, const int offset, uint64x2_t value)	*addr -> Qn offset in +/- 8*[0127] value -> Qd	VSTRD.U64 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrdq_scatter_base_wb_p[_s64](uint64x2_t * addr, const int offset, int64x2_t value, mve_pred16_t p)	*addr -> Qn offset in +/- 8*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrdq_scatter_base_wb_p[_u64](uint64x2_t * addr, const int offset, uint64x2_t value, mve_pred16_t p)	*addr -> Qn offset in +/- 8*[0127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]!	void -> result	MVE
void [_arm_]vstrdq_scatter_offset[_s64](int64_t * base, uint64x2_t offset, int64x2_t value)	base -> Rn offset -> Qm value -> Qd	VSTRD.64 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrdq_scatter_offset[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value)	base -> Rn offset -> Qm value -> Qd	VSTRD.64 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrdq_scatter_offset_p[_s64](int64_t * base, uint64x2_t offset, int64x2_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrdq_scatter_offset_p[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm]	void -> result	MVE
void [_arm_]vstrdq_scatter_shifted_offset[_s64](int64_t * base, uint64x2_t offset, int64x2_t value)	base -> Rn offset -> Qm value -> Qd	VSTRD.64 Qd,[Rn,Qm,UXTW #3]	void -> result	MVE
void [_arm_]vstrdq_scatter_shifted_offset[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value)	base -> Rn offset -> Qm value -> Qd	VSTRD.64 Qd,[Rn,Qm,UXTW #3]	void -> result	MVE
void [arm_]vstrdq_scatter_shifted_offset_p[_s64](int64_t * base, uint64x2_t offset, int64x2_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm,UXTW #3]	void -> result	MVE
void [_arm_]vstrdq_scatter_shifted_offset_p[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value, mve_pred16_t p)	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm,UXTW #3]	void -> result	MVE
int64_t [_arm_]vaddlvaq[_s32](int64_t a, int32x4_t b)	a -> [RdaHi,RdaLo] b -> Qm	VADDLVA.S32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vaddlvaq[_u32](uint64_t a, uint32x4_t b)	a -> [RdaHi,RdaLo] b -> Qm	VADDLVA.U32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vaddlvaq_p[_s32](int64_t a, int32x4_t b, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qm p -> Rp	VMSR P0,Rp VPST VADDLVAT.S32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architecture:
uint64_t [arm_]vaddlvaq_p[_u32](uint64_t a,	a ->	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
uint32x4_t b, mve_pred16_t p)	[RdaHi,RdaLo]	VPST	-> result	
	b -> Qm	VADDLVAT.U32		
	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	[RdaHi,RdaLo]	MVE
		RdaLo,RdaHi,Qm	-> result	
int64_t [arm_]vaddlvq_p[_s32](int32x4_t a,	a -> Qm	VMSR P0,Rp	[RdaHi,RdaLo]	MVE
mve_pred16_t p)	p -> Rp	VPST	-> result	
		VADDLVT.S32		
-int(4 + f	0	RdaLo,RdaHi,Qm	(D.1-11; D.1-1 - 1	MVE
uint64_t [arm_]vaddlvq_p[_u32](uint32x4_t a, nve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST	[RdaHi,RdaLo] -> result	NIVE
nive_picuro_t p)	p -> Kp	VADDLVT.U32	-> icsuit	
		RdaLo,RdaHi,Qm		
int32_t [arm_]vaddvaq[_s8](int32_t a, int8x16_t b)	a -> Rda	VADDVA.S8 Rda,Qm	Rda -> result	MVE
mez_t [atm_] (add /aq[_so](mez_t a, mestro_t o)	b -> Om	TIDD TIME TOUR, QIII	Ttut > Tootil	,2
nt32_t [arm_]vaddvaq[_s16](int32_t a, int16x8_t b)	a -> Rda	VADDVA.S16 Rda,Qm	Rda -> result	MVE
	b -> Qm			
nt32_t [arm_]vaddvaq[_s32](int32_t a, int32x4_t b)	a -> Rda	VADDVA.S32 Rda,Qm	Rda -> result	MVE
	b -> Qm			
uint32_t [arm_]vaddvaq[_u8](uint32_t a, uint8x16_t b)	a -> Rda	VADDVA.U8 Rda,Qm	Rda -> result	MVE
	b -> Qm			
uint32_t [arm_]vaddvaq[_u16](uint32_t a, uint16x8_t b)	a -> Rda	VADDVA.U16 Rda,Qm	Rda -> result	MVE
	b -> Qm			
uint32_t [arm_]vaddvaq[_u32](uint32_t a, uint32x4_t b)	a -> Rda	VADDVA.U32 Rda,Qm	Rda -> result	MVE
	b -> Qm			
nt32_t [arm_]vaddvaq_p[_s8](int32_t a, int8x16_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
nve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VADDVAT.S8 Rda,Qm		
nt32_t [arm_]vaddvaq_p[_s16](int32_t a, int16x8_t b,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
nve_pred16_t p)	b -> Qm	VPST		
(20) [] 11	p -> Rp	VADDVAT.S16 Rda,Qm	D.1) (T/III)
int32_t [arm_]vaddvaq_p[_s32](int32_t a, int32x4_t b,	a -> Rda b -> Om	VMSR P0,Rp VPST	Rda -> result	MVE
mve_pred16_t p)	p -> QIII	VADDVAT.S32 Rda,Qm		
uint32_t [arm_]vaddvaq_p[_u8](uint32_t a, uint8x16_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
o, mve_pred16_t p)	b -> Om	VPST	Rua => 1Csuit	MVL
s, mvc_predio_t p)	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 -> Om	VPST		
	p -> Rp	VADDVAT.U16 Rda,Qm		
uint32_t [arm_]vaddvaq_p[_u32](uint32_t a, uint32x4_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
o, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VADDVAT.U32 Rda,Qm		
nt32_t [arm_]vaddvq[_s8](int8x16_t a)	a -> Qm	VADDV.S8 Rda,Qm	Rda -> result	MVE
nt32_t [arm_]vaddvq[_s16](int16x8_t a)	a -> Qm	VADDV.S16 Rda,Qm	Rda -> result	MVE
nt32_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
nt32_t [arm_]vaddvq_p[_s8](int8x16_t a,	a -> Qm	VMSR P0,Rp	Rda -> result	MVE
nve_pred16_t p)	p -> Rp	VPST		
20.1		VADDVT.S8 Rda,Qm	7.1) (T) (T)
nt32_t [arm_]vaddvq_p[_s16](int16x8_t a,	a -> Qm	VMSR P0,Rp	Rda -> result	MVE
mve_pred16_t p)	p -> Rp	VPST		
nt32_t [arm_]vaddvq_p[_s32](int32x4_t a,	a -> Qm	VADDVT.S16 Rda,Qm VMSR P0,Rp	Rda -> result	MVE
nts2_t [arm_]vaudvq_p[_s52](mts2x4_t a, nve_pred16_t p)	a -> QIII p -> Rp	VMSK PO,RP VPST	Kua -> resuit	NIVE
nive_picuro_t p)	p -> K p	VADDVT.S32 Rda,Qm		
uint32 t [arm]vaddvq p[u8](uint8x16 t a,	a -> Qm	VMSR P0,Rp	Rda -> result	MVE
nve_pred16_t p)	p -> Rp	VPST	-tau - rooun	
-1 · · · · · · 1 /	r ≔r	VADDVT.U8 Rda,Qm		
uint32_t [arm_]vaddvq_p[_u16](uint16x8_t a,	a -> Qm	VMSR P0,Rp	Rda -> result	MVE
nve_pred16_t p)	p -> Rp	VPST		
		VADDVT.U16 Rda,Qm		
	a -> Qm	VMSR P0,Rp	Rda -> result	MVE
uint32_t [arm_]vaddvq_p[_u32](uint32x4_t a,	p -> Rp	VPST		
uint32_t [arm_]vaddvq_p[_u32](uint32x4_t a, mve_pred16_t p)	P - TO	VADDVT.U32 Rda,Qm	ĺ	
	P - 14P	171DD 11.032 Rda,QIII		
mve_pred16_t p) int32_t [arm_]vmladavaq[_s8](int32_t a, int8x16_t b,	a -> Rda	VMLADAVA.S8 Rda,Qn,Qm	Rda -> result	MVE
mve_pred16_t p)	a -> Rda b -> Qn		Rda -> result	MVE
mve_pred16_t p) int32_t [arm_]vmladavaq[_s8](int32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVA.S8 Rda,Qn,Qm		
mve_pred16_t p) int32_t [arm_]vmladavaq[_s8](int32_t a, int8x16_t b,	a -> Rda b -> Qn		Rda -> result Rda -> result	MVE MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32_t [arm_]vmladavaq[_s32](int32_t a, int32x4_t b,	a -> Rda	VMLADAVA.S32 Rda,Qn,Qm	Rda -> result	MVE
int32x4_t c)	b -> Qn c -> Qm			
uint32_t [arm_]vmladavaq[_u8](uint32_t a, uint8x16_t	a -> Rda	VMLADAVA.U8 Rda,Qn,Qm	Rda -> result	MVE
b, uint8x16_t c)	b -> Qn			
uint32_t [arm_]vmladavaq[_u16](uint32_t a, uint16x8_t	c -> Qm a -> Rda	VMLADAVA.U16 Rda,Qn,Qm	Rda -> result	MVE
b, uint16x8_t c)	b -> Qn	, , , , , , , , , , , , , , , , , , ,	True > Toour	
uint32 t [arm]vmladavag[u32](uint32 t a, uint32x4 t	c -> Qm a -> Rda	VMLADAVA.U32 Rda,Qn,Qm	Rda -> result	MVE
b, uint32x4_t c)	b -> Qn	VIVIEZ IDZI VZI. USZ Kua, QII, QIII	Rua -> resuit	WIVE
int32_t [arm_]vmladavaq_p[_s8](int32_t a, int8x16_t b,	c -> Qm a -> Rda	VMCD DO D	Rda -> result	MVE
int8x16 t c, mve pred16 t p)	a -> Rda b -> On	VMSR P0,Rp VPST	Kda -> resuit	MVE
= ·	c -> Qm	VMLADAVAT.S8 Rda,Qn,Qm		
int32_t [arm_]vmladavaq_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		1
	c -> Qm p -> Rp	VMLADAVAT.S16 Rda,Qn,Qm		
int32_t [arm_]vmladavaq_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	VPST		
	c -> Qm p -> Rp	VMLADAVAT.S32 Rda,Qn,Qm		
uint32_t [arm_]vmladavaq_p[_u8](uint32_t a,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
uint8x16_t b, uint8x16_t c, mve_pred16_t p)	b -> Qn c -> Qm	VPST VMLADAVAT.U8 Rda,Qn,Qm		
	p -> Rp	VIVIEZIDZI VZII. OʻO Rda, Qii, Qiii		
uint32_t [arm_]vmladavaq_p[_u16](uint32_t a,	a -> Rda	VMSR P0,Rp VPST	Rda -> result	MVE
uint16x8_t b, uint16x8_t c, mve_pred16_t p)	b -> Qn c -> Qm	VMLADAVAT.U16 Rda,Qn,Qm		
	p -> Rp			
uint32_t [arm_]vmladavaq_p[_u32](uint32_t a, uint32x4_t b, uint32x4_t c, mve_pred16_t p)	a -> Rda b -> On	VMSR P0,Rp VPST	Rda -> result	MVE
umozkijt o, umozkijt o, mio prodrojt p)	c -> Qm	VMLADAVAT.U32 Rda,Qn,Qm		
int32_t [arm_]vmladavq[_s8](int8x16_t a, int8x16_t b)	p -> Rp a -> Qn	VMLADAV.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_Jvimadavq[_s8](int0x10_t a, int8x10_t b)	b -> Qm	VIVILADA V.So Kua,Qii,Qiii	Kua -> resuit	MIVE
int32_t [arm_]vmladavq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VMLADAV.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq[_s32](int32x4_t a, int32x4_t b)	b -> Qm a -> Qn	VMLADAV.S32 Rda,Qn,Qm	Rda -> result	MVE
<u> </u>	b->Qm			
uint32_t [arm_]vmladavq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMLADAV.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [arm_]vmladavq[_u16](uint16x8_t a,	a -> Qn	VMLADAV.U16 Rda,Qn,Qm	Rda -> result	MVE
uint16x8_t b)	b -> Qm	VIIII ADAM HOODE O	D.I. I	NOTE
uint32_t [arm_]vmladavq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMLADAV.U32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavq_p[_s8](int8x16_t a, int8x16_t	a -> Qn	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMLADAVT.S8 Rda,Qn,Qm		
int32_t [arm_]vmladavq_p[_s16](int16x8_t a, int16x8_t	a -> Qn	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
int32_t [arm_]vmladavq_p[_s32](int32x4_t a, int32x4_t	p -> Rp a -> Qn	VMLADAVT.S16 Rda,Qn,Qm VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
uint32 t[arm]vmladavq p[u8](uint8x16 t a,	p -> Rp a -> Qn	VMLADAVT.S32 Rda,Qn,Qm VMSR P0,Rp	Rda -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST	redu > result	141 4 E
uint32 t[arm]vmladavq p[u16](uint16x8 ta,	p -> Rp a -> On	VMLADAVT.U8 Rda,Qn,Qm VMSR P0,Rp	Rda -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VMSR PO,RP VPST	Rda -> resuit	MVE
	p -> Rp	VMLADAVT.U16 Rda,Qn,Qm	7.1) arm
uint32_t [arm_]vmladavq_p[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
	p -> Rp	VMLADAVT.U32 Rda,Qn,Qm		
int32_t [arm_]vmladavaxq[_s8](int32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn	VMLADAVAX.S8 Rda,Qn,Qm	Rda -> result	MVE
intox10_t c)	c -> Qn			<u> </u>
int32_t [arm_]vmladavaxq[_s16](int32_t a, int16x8_t b,	a -> Rda	VMLADAVAX.S16 Rda,Qn,Qm	Rda -> result	MVE
int16x8_t c)	b -> Qn c -> Qm			
int32_t [arm_]vmladavaxq[_s32](int32_t a, int32x4_t b,	a -> Rda	VMLADAVAX.S32 Rda,Qn,Qm	Rda -> result	MVE
int32x4_t c)	b -> Qn c -> Qm			
	~ -> QIII	1	1	1

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32_t [arm_]vmladavaxq[_u8](uint32_t a, uint8x16_t	a -> Rda	VMLADAVAX.U8 Rda,Qn,Qm	Rda -> result	MVE
b, uint8x16_t c)	b -> Qn			
uint32 t [arm]vmladavaxq[u16](uint32 t a,	c -> Qm a -> Rda	VMLADAVAX.U16 Rda,Qn,Qm	Rda -> result	MVE
uint16x8_t b, uint16x8_t c)	b -> Qn	VIVILABATVAAT. OTO Kua, Qii, Qiii	Rda -> resuit	WIVE
	c -> Qm			
uint32_t [arm_]vmladavaxq[_u32](uint32_t a, uint32x4_t b, uint32x4_t c)	a -> Rda b -> On	VMLADAVAX.U32 Rda,Qn,Qm	Rda -> result	MVE
um32x4_t b, um32x4_t c)	c -> Qm			
int32_t [arm_]vmladavaxq_p[_s8](int32_t a, int8x16_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, int8x16_t c, mve_pred16_t p)	b -> Qn c -> Qm	VPST VMLADAVAXT.S8 Rda,Qn,Qm		
	p -> Rp	VWLADA VAA 1.36 Kua,Qii,Qiii		
int32_t [arm_]vmladavaxq_p[_s16](int32_t a, int16x8_t	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
b, int16x8_t c, mve_pred16_t p)	b -> Qn c -> Qm	VPST VMLADAVAXT.S16		
	p -> Rp	Rda,Qn,Qm		
int32_t [arm_]vmladavaxq_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	VPST		
	c -> Qm p -> Rp	VMLADAVAXT.S32 Rda,Qn,Qm		
uint32_t [arm_]vmladavaxq_p[_u8](uint32_t a,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
uint8x16_t b, uint8x16_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Qm p -> Rp	VMLADAVAXT.U8 Rda,Qn,Qm		
uint32_t [arm_]vmladavaxq_p[_u16](uint32_t a,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
uint16x8_t b, uint16x8_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Qm p -> Rp	VMLADAVAXT.U16 Rda,Qn,Qm		
uint32_t [arm_]vmladavaxq_p[_u32](uint32_t a,	a -> Rda	VMSR P0,Rp	Rda -> result	MVE
uint32x4_t b, uint32x4_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Qm	VMLADAVAXT.U32		
int32_t [arm_]vmladavxq[_s8](int8x16_t a, int8x16_t b)	p -> Rp a -> Qn	Rda,Qn,Qm VMLADAVX.S8 Rda,Qn,Qm	Rda -> result	MVE
	b -> Qm		redu > resure	
int32_t [arm_]vmladavxq[_s16](int16x8_t a, int16x8_t	a -> Qn	VMLADAVX.S16 Rda,Qn,Qm	Rda -> result	MVE
b) int32_t [arm_]vmladavxq[_s32](int32x4_t a, int32x4_t	b -> Qm a -> On	VMLADAVX.S32 Rda,Qn,Qm	Rda -> result	MVE
b)	b -> Qm	7 2 7 2		
uint32_t [arm_]vmladavxq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMLADAVX.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [arm_]vmladavxq[_u16](uint16x8_t a,	a -> Qn	VMLADAVX.U16 Rda,Qn,Qm	Rda -> result	MVE
uint16x8_t b)	b -> Qm			
uint32_t [arm_]vmladavxq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMLADAVX.U32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmladavxq_p[_s8](int8x16_t a, int8x16_t	a -> Qn	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
int22 to ann lumb dayya no a161/int16y9 to	p -> Rp a -> On	VMLADAVXT.S8 Rda,Qn,Qm VMSR P0,Rp	Rda -> result	MVE
int32_t [arm_]vmladavxq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	b -> Qm	VMSK PO,KP VPST	Rda -> resuit	MIVE
_ · _ _ _ _	p -> Rp	VMLADAVXT.S16 Rda,Qn,Qm		
int32_t [arm_]vmladavxq_p[_s32](int32x4_t a,	a -> Qn b -> Om	VMSR P0,Rp	Rda -> result	MVE
int32x4_t b, mve_pred16_t p)	p -> QIII	VPST VMLADAVXT.S32 Rda,Qn,Qm		
uint32_t [arm_]vmladavxq_p[_u8](uint8x16_t a,	a -> Qn	VMSR P0,Rp	Rda -> result	MVE
uint8x16_t b, mve_pred16_t p)	b -> Qm	VPST		
uint32_t [arm_]vmladavxq_p[_u16](uint16x8_t a,	p -> Rp a -> Qn	VMLADAVXT.U8 Rda,Qn,Qm VMSR P0,Rp	Rda -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST	reda > resure	W V E
	p -> Rp	VMLADAVXT.U16 Rda,Qn,Qm	D.I.	NOTE
uint32_t [arm_]vmladavxq_p[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPST	Rda -> result	MVE
umozati o, mve_predio_t p)	p -> Rp	VMLADAVXT.U32 Rda,Qn,Qm	<u> </u>	<u> </u>
int64_t [arm_]vmlaldavaq[_s16](int64_t a, int16x8_t b,	a ->	VMLALDAVA.S16	[RdaHi,RdaLo]	MVE
antiferry to)		L Paul o Pauli On Om	-> result	
int16x8_t c)	[RdaHi,RdaLo]	RdaLo,RdaHi,Qn,Qm		
mttoxo_t c)		Kualo,Kuarii,Qii,Qiii		
int64_t [arm_]vmlaldavaq[_s32](int64_t a, int32x4_t b,	[RdaHi,RdaLo] b -> Qn c -> Qm a ->	VMLALDAVA.S32	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo] b-> Qn c-> Qm a-> [RdaHi,RdaLo]		[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavaq[_s32](int64_t a, int32x4_t b,	[RdaHi,RdaLo] b -> Qn c -> Qm a ->	VMLALDAVA.S32		MVE
int64_t [_arm_]vmlaldavaq[_s32](int64_t a, int32x4_t b, int32x4_t c) uint64_t [_arm_]vmlaldavaq[_u16](uint64_t a,	[RdaHi,RdaLo] b -> Qn c -> Qm a -> [RdaHi,RdaLo] b -> Qn c -> Qm a ->	VMLALDAVA.S32 RdaLo,RdaHi,Qn,Qm	-> result [RdaHi,RdaLo]	MVE MVE
int64_t [arm_]vmlaldavaq[_s32](int64_t a, int32x4_t b, int32x4_t c)	[RdaHi,RdaLo] b-> Qn c-> Qm a-> [RdaHi,RdaLo] b-> Qn c-> Qm	VMLALDAVA.S32 RdaLo,RdaHi,Qn,Qm	-> result	

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64_t [arm_]vmlaldavaq[_u32](uint64_t a, uint32x4_t b, uint32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVA.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavaq_p[_s16](int64_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavaq_p[_s32](int64_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]vmlaldavaq_p[_u16](uint64_t a, uint16x8_t b, uint16x8_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAT.U16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vmlaldavaq_p[_u32](uint64_t a, uint32x4_t b, uint32x4_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAT.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMLALDAV.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMLALDAV.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]vmlaldavq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMLALDAV.U16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]vmlaldavq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMLALDAV.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]vmlaldavq_p[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVT.U16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vmlaldavq_p[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVT.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavaxq[_s16](int64_t a, int16x8_t b, int16x8_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVAX.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
$ \begin{array}{lll} int64_t & [_arm_]vmlaldavaxq[_s32](int64_t \ a, \ int32x4_t \ b, \\ int32x4_t \ c) \end{array} $	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVAX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]vmlaldavaxq[_u16](uint64_t a, uint16x8_t b, uint16x8_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVAX.U16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [arm_]vmlaldavaxq[_u32](uint64_t a, uint32x4_t b, uint32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVAX.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlaldavaxq_p[_s16](int64_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAXT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlaldavaxq_p[_s32](int64_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLALDAVAXT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vmlasq[_n_s8](int8x16_t a, int8x16_t	a -> Qda	VMLAS.S8 Qda,Qn,Rm	Oda -> result	MVE
b, int8_t c)	b -> Qn c -> Rm	VIVILAS.58 Qua,Qii,Kiii	Qua -> resuit	MVE
$ \begin{array}{l} int16x8_t \ [_arm_]vmlasq[_n_s16](int16x8_t \ a, int16x8_t \\ b, int16_t \ c) \end{array} $	a -> Qda b -> Qn c -> Rm	VMLAS.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [arm_]vmlasq[_n_s32](int32x4_t a, int32x4_t	a -> Qda	VMLAS.S32 Qda,Qn,Rm	Oda -> result	MVE
b, int32_t c)	b -> Qn c -> Rm	7 . 7 .		
uint8x16_t [_arm_]vmlasq[_n_u8](uint8x16_t a, uint8x16_t b, uint8_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [arm_]vmlasq[_n_u16](uint16x8_t a,	a -> Qda	VMLAS.U16 Qda,Qn,Rm	Oda -> result	MVE
uint16x8_t b, uint16_t c)	b -> Qn c -> Rm			
uint32x4_t [_arm_]vmlasq[_n_u32](uint32x4_t a,	a -> Qda	VMLAS.U32 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t b, uint32_t c)	b -> Qn c -> Rm			
int8x16_t [arm_]vmlasq_m[_n_s8](int8x16_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int8x16_t b, int8_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Rm p -> Rp	VMLAST.S8 Qda,Qn,Rm		
int16x8_t [arm_]vmlasq_m[_n_s16](int16x8_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int16x8_t b, int16_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Rm p -> Rp	VMLAST.S16 Qda,Qn,Rm		
int32x4_t [arm_]vmlasq_m[_n_s32](int32x4_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32x4_t b, int32_t c, mve_pred16_t p)	b -> Qn	VPST		
	c -> Rm	VMLAST.S32 Qda,Qn,Rm		
uint8x16_t [arm_]vmlasq_m[_n_u8](uint8x16_t a,	p -> Rp a -> Qda	VMSR P0,Rp	Qda -> result	MVE
uint8x16_t b, uint8_t c, mve_pred16_t p)	b -> Qn	VPST	Q	
	c -> Rm	VMLAST.U8 Qda,Qn,Rm		
uint16x8_t [arm_]vmlasq_m[_n_u16](uint16x8_t a,	p -> Rp a -> Qda	VMSR P0,Rp	Oda -> result	MVE
uint16x8_t b, uint16_t c, mve_pred16_t p)	b -> On	VPST	Qua -> resuit	MVE
_ / _ / /	c -> Rm	VMLAST.U16 Qda,Qn,Rm		
wint22w4 t.f. com lymloss mf m y22l/wint22w4 t.s.	p -> Rp	VMCD DO Do	Odo > monule	MVE
uint32x4_t [arm_]vmlasq_m[_n_u32](uint32x4_t a, uint32x4_t b, uint32_t c, mve_pred16_t p)	a -> Qda b -> Qn	VMSR P0,Rp VPST	Qda -> result	MVE
	c -> Rm	VMLAST.U32 Qda,Qn,Rm		
	p -> Rp	ADDICE ANA CORL O	D.1	MATE
int32_t [arm_]vmlsdavaq[_s8](int32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn	VMLSDAVA.S8 Rda,Qn,Qm	Rda -> result	MVE
110/110_00	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	VPST	Rua -> resurt	MVE
-	c -> Qm	VMLSDAVAT.S8 Rda,Qn,Qm		
int32 t [arm]vmlsdayaq 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	VMSK FO,KP VPST	Kua -> iesuit	MVE
, <u> </u>	c -> Qm	VMLSDAVAT.S16 Rda,Qn,Qm		
int32_t [arm_]vmlsdavaq_p[_s32](int32_t a, int32x4_t	p -> Rp	VMSD DO Do	Ddo > ==14	MVE
b, int32x4_t c, mve_pred16_t p)	a -> Rda b -> Qn	VMSR P0,Rp VPST	Rda -> result	MVE
,	c -> Qm	VMLSDAVAT.S32 Rda,Qn,Qm		
2010 11 and 1 1 000 0 10 10 10 10	p -> Rp	VMI CDAY CORT C	D.4. · · ·	MVE
int32_t [arm_]vmlsdavq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMLSDAV.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMLSDAV.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMLSDAV.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavq_p[_s8](int8x16_t a, int8x16_t	a -> Qn	VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST		
int32_t [arm_]vmlsdavq_p[_s16](int16x8_t a, int16x8_t	p -> Rp a -> Qn	VMLSDAVT.S8 Rda,Qn,Qm VMSR P0,Rp	Rda -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPST	100000	
	p -> Rp	VMLSDAVT.S16 Rda,Qn,Qm		

101809

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32_t [arm_]vmlsdavq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVT.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavaxq[_s8](int32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn c -> Qm	VMLSDAVAX.S8 Rda,Qn,Qm	Rda -> result	MVE
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, int32x4_t c)	a -> Rda b -> Qn c -> Qm	VMLSDAVAX.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavaxq_p[_s8](int32_t a, int8x16_t b, int8x16_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVAXT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavaxq_p[_s16](int32_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVAXT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavaxq_p[_s32](int32_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVAXT.S32 Rda,Qn,Qm	Rda -> result	MVE
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 p -> Rp	VMSR P0,Rp VPST VMLSDAVXT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavxq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVXT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [arm_]vmlsdavxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVXT.S32 Rda,Qn,Qm	Rda -> result	MVE
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, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLSLDAVA.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlsldavaq_p[_s16](int64_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLSLDAVAT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlsldavaq_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 VMLSLDAVAT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlsldavq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMLSLDAV.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlsldavq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMLSLDAV.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlsldavq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLSLDAVT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [arm_]vmlsldavq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLSLDAVT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
$int64_t \ [_arm_]vmlsldavaxq[_s16](int64_t \ a, int16x8_t \ b, int16x8_t \ c)$	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLSLDAVAX.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [_arm_]vmlsldavaxq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLSLDAVAX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64_t [_arm_]vmlsldavaxq_p[_s16](int64_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> [RdaHi,RdaLo]	VMSR P0,Rp VPST	[RdaHi,RdaLo] -> result	MVE
	b -> Qn c -> Qm p -> Rp	VMLSLDAVAXT.S16 RdaLo,RdaHi,Qn,Qm		
int64_t [arm_]vmlsldavaxq_p[_s32](int64_t a, int32x4_t	a -> [RdaHi,RdaLo]	VMSR P0,Rp VPST	[RdaHi,RdaLo] -> result	MVE
b, int32x4_t c, mve_pred16_t p)	b -> Qn c -> Qm	VMLSLDAVAXT.S32 RdaLo,RdaHi,Qn,Qm	-> resuit	
int64_t [arm_]vmlsldavxq[_s16](int16x8_t a, int16x8_t	p -> Rp a -> Qn	VMLSLDAVX.S16	[RdaHi,RdaLo]	MVE
b) int64_t [_arm_]vmlsldavxq[_s32](int32x4_t a, int32x4_t	b -> Qm a -> Qn	RdaLo,RdaHi,Qn,Qm VMLSLDAVX.S32	-> result [RdaHi,RdaLo]	MVE
b) int64_t [arm_]vmlsldavxq_p[_s16](int16x8_t a,	b -> Qm a -> Qn	RdaLo,RdaHi,Qn,Qm VMSR P0,Rp	-> result [RdaHi,RdaLo]	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMLSLDAVXT.S16 RdaLo,RdaHi,Qn,Qm	-> result	
int64_t [_arm_]vmlsldavxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLSLDAVXT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int8x16_t [arm_]vhaddq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VHADD.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vhaddq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VHADD.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [arm_]vhaddq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VHADD.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vhaddq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VHADD.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [arm_]vhaddq[_n_u16](uint16x8_t a, uint16 t b)	a -> Qn b -> Rm	VHADD.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vhaddq[_n_u32](uint32x4_t a, 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 b -> Qm	VHADD.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vhaddq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VHADD.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vhaddq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VHADD.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vhaddq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VHADD.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vhaddq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VHADD.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vhaddq[_u32](uint32x4_t a, 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, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm	VMSR P0,Rp VPST VHADDT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vhaddq_m[_n_s16](int16x8_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VPST VHADDT.S16 Qd,Qn,Rm		
int32x4_t [_arm_]vhaddq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Rm p -> Rp	VHADDT.S32 Qd,Qn,Rm		
uint8x16_t [_arm_]vhaddq_m[_n_u8](uint8x16_t inactive, uint8x16 t a, uint8 t b, mve pred16 t p)	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Rm p -> Rp	VHADDT.U8 Qd,Qn,Rm		
uint16x8_t [_arm_]vhaddq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm	VMSR P0,Rp VPST VHADDT.U16 Qd,Qn,Rm	Qd -> result	MVE
nint20v4 t.f. orm luboddo	p -> Rp	2 / 2 /	Od > =====14	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	VMSR P0,Rp VPST	Qd -> result	MVE
	b -> Qm p -> Rp	VHADDT.S8 Qd,Qn,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [_arm_]vhaddq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vhaddq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vhaddq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vhaddq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vhaddq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [_arm_]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
int32x4_t [_arm_]vhcaddq_rot90[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Om	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_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 p -> Rp	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)	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 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 b -> Rm	VHSUB.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vhsubq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VHSUB.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [_arm_]vhsubq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VHSUB.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [_arm_]vhsubq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VHSUB.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [_arm_]vhsubq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VHSUB.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vhsubq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VHSUB.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vhsubq[_s32](int32x4_t a, int32x4_t b)	a -> Qn 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

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint8x16_t [_arm_]vrhaddq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRHADDT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vrhaddq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRHADDT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vrhaddq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VRHADDT.U32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [arm_]vfmaq[_n_f16](float16x8_t a, float16x8_t b, float16_t c)	a -> Qda b -> Qn c -> Rm	VFMA.F16 Qda,Qn,Rm	Qda -> result	MVE/NEON
float32x4_t [arm_]vfmaq[_n_f32](float32x4_t a, float32x4_t b, float32_t c)	a -> Qda b -> Qn c -> Rm	VFMA.F32 Qda,Qn,Rm	Qda -> result	MVE/NEON
float16x8_t [arm_]vfmaq_m[_n_f16](float16x8_t a, float16x8_t b, float16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VFMAT.F16 Qda,Qn,Rm	Qda -> result	MVE
float32x4_t [_arm_]vfmaq_m[_n_f32](float32x4_t a, float32x4_t b, float32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VFMAT.F32 Qda,Qn,Rm	Qda -> result	MVE
float16x8_t [arm_]vfmaq[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Qm	VFMA.F16 Qda,Qn,Qm	Qda -> result	MVE/NEON
float32x4_t [arm_]vfmaq[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VFMA.F32 Qda,Qn,Qm	Qda -> result	MVE/NEON
float16x8_t [arm_]vfmaq_m[_f16](float16x8_t a, float16x8_t b, float16x8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VFMAT.F16 Qda,Qn,Qm	Qda -> result	MVE
float32x4_t [arm_]vfmaq_m[_f32](float32x4_t a, float32x4_t b, float32x4_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VFMAT.F32 Qda,Qn,Qm	Qda -> result	MVE
float16x8_t [arm_]vfmasq[_n_f16](float16x8_t a, float16x8_t b, float16_t c)	a -> Qda b -> Qn c -> Rm	VFMAS.F16 Qda,Qn,Rm	Qda -> result	MVE
float32x4_t [arm_]vfmasq[_n_f32](float32x4_t a, float32x4_t b, float32_t c)	a -> Qda b -> Qn c -> Rm	VFMAS.F32 Qda,Qn,Rm	Qda -> result	MVE
float16x8_t [arm_]vfmasq_m[_n_f16](float16x8_t a, float16x8_t b, float16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VFMAST.F16 Qda,Qn,Rm	Qda -> result	MVE
float32x4_t [arm_]vfmasq_m[_n_f32](float32x4_t a, float32x4_t b, float32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VFMAST.F32 Qda,Qn,Rm	Qda -> result	MVE
float16x8_t [arm_]vfmsq[_f16](float16x8_t a, float16x8_t b, float16x8_t c)	a -> Qda b -> Qn c -> Qm	VFMS.F16 Qda,Qn,Qm	Qda -> result	MVE/NEON
float32x4_t [_arm_]vfmsq[_f32](float32x4_t a, float32x4_t b, float32x4_t c)	a -> Qda b -> Qn c -> Qm	VFMS.F32 Qda,Qn,Qm	Qda -> result	MVE/NEON
float16x8_t [arm_]vfmsq_m[_f16](float16x8_t a, float16x8_t b, float16x8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VFMST.F16 Qda,Qn,Qm	Qda -> result	MVE
float32x4_t [arm_]vfmsq_m[_f32](float32x4_t a, float32x4_t b, float32x4_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VFMST.F32 Qda,Qn,Qm	Qda -> result	MVE
int64_t [_arm_]vrmlaldavhaq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VRMLALDAVHA.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [_arm_]vrmlaldavhaq[_u32](uint64_t a, uint32x4_t b, uint32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VRMLALDAVHA.U32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
int16x8_t [arm_]vcvtaq_s16_f16(float16x8_t a)	a -> Qm	VCVTA.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vcvtaq_s32_f32(float32x4_t a)	a -> Qm	VCVTA.S32.F32 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vcvtaq_u16_f16(float16x8_t a)	a -> Qm	VCVTA.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vcvtaq_u32_f32(float32x4_t a)	a -> Qm	VCVTA.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vcvtaq_m[_s16_f16](int16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.S16.F16 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vcvtaq_m[_s32_f32](int32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.S32.F32 Qd,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vcvtaq_m[_u16_f16](uint16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.U16.F16 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vcvtaq_m[_u32_f32](uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTAT.U32.F32 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vcvtnq_s16_f16(float16x8_t a)	a -> Qm	VCVTN.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vcvtnq_s32_f32(float32x4_t a)	a -> Om	VCVTN.S32.F32 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vcvtnq_u16_f16(float16x8_t a)	a -> Qm	VCVTN.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vcvtnq_u32_f32(float32x4_t a)	a -> Qm	VCVTN.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vcvtnq_m[_s16_f16](int16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTNT.S16.F16 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vcvtnq_m[_s32_f32](int32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTNT.S32.F32 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vcvtnq_m[_u16_f16](uint16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST VCVTNT.U16.F16 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vcvtnq_m[_u32_f32](uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qm p -> Rp	VCVTNT.U10.F10 Qd,Qlll VMSR P0,Rp VPST VCVTNT.U32.F32 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vcvtpq_s16_f16(float16x8_t a)	a -> Qm	VCVTP.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vcvtpq_s32_f32(float32x4_t a)	a -> Qm	VCVTP.S32.F32 Qd,Qm	Qd -> result	MVE/NEON
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, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTPT.S16.F16 Qd,Qm	Qd -> result	MVE
int32x4_t [_arm_]vcvtpq_m[_s32_f32](int32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTPT.S32.F32 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vcvtpq_m[_u16_f16](uint16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTPT.U16.F16 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vcvtpq_m[_u32_f32](uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VCVTPT.U16.F16 Qd,Qml VMSR P0,Rp VPST VCVTPT.U32.F32 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vcvtmq_s16_f16(float16x8_t a)	a -> Qm	VCVTM.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vcvtmq_s32_f32(float32x4_t a)	a -> Qm	VCVTM.S32.F32 Qd,Qm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8 t [arm]vcvtmq u16 f16(float16x8 t a)	a -> Qm	VCVTM.U16.F16 Qd,Qm	Od -> 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_]vcvtmq_m[_s16_f16](int16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
int32x4_t [arm_]vcvtmq_m[_s32_f32](int32x4_t	p -> Rp inactive -> Od	VCVTMT.S16.F16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Om	VMSK FO,KP VPST	Qu -> resuit	MIVE
materie, notabezhiet u, mre_predio_t p)	p -> Rp	VCVTMT.S32.F32 Qd,Qm		
uint16x8_t [arm_]vcvtmq_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	VCVTMT.U16.F16 Qd,Qm	0.1 1:	MATE
uint32x4_t [_arm_]vcvtmq_m[_u32_f32](uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, noats2x4_t a, mve_pred16_t p)	a -> QIII p -> Rp	VCVTMT.U32.F32 Qd,Qm		
float16x8_t [arm_]vcvtbq_f16_f32(float16x8_t a,	a -> Od	VCVTB.F16.F32 Qd,Qm	Od -> result	MVE
float32x4_t b)	b -> Qm		Q	1
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		
G -22 4 - F - 3 - 4 - 22 CLC/G -22 4 -	p -> Rp	VCVTBT.F16.F32 Qd,Qm	0.1) (T) (T)
float32x4_t [arm_]vcvtbq_m_f32_f16(float32x4_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, noatroxo_t a, mve_pred1o_t p)	p -> Rp	VCVTBT.F32.F16 Qd,Qm		
float16x8_t [arm_]vcvttq_f16_f32(float16x8_t a,	a -> Od	VCVTT.F16.F32 Qd,Qm	Od -> result	MVE
float32x4_t b)	b -> Qm		Q	1
float32x4_t [arm_]vcvttq_f32_f16(float16x8_t a)	a -> Qm	VCVTT.F32.F16 Qd,Qm	Qd -> result	MVE
float16x8_t [arm_]vcvttq_m_f16_f32(float16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
float32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
GL 222 4 4 F 3 4 4 5 22 4 4	p -> Rp	VCVTTT.F16.F32 Qd,Qm	0.1 1:	MATE
float32x4_t [arm_]vcvttq_m_f32_f16(float32x4_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, noatroxo_t a, mve_pred1o_t p)	a -> QIII p -> Rp	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 -> Om	VCVT.F16.U16 Qd,Qm	Od -> 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 -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, int16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
float16x8_t [arm_]vcvtq_m[_f16_u16](float16x8_t	p -> Rp inactive -> Qd	VCVTT.F16.S16 Qd,Qm VMSR P0,Rp	Od -> result	MVE
inactive, uint16x8_t a, mve_pred16_t p)	a -> Qm	VMSK FO,KP VPST	Qu -> resuit	MIVE
	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 VPST	Qd -> result	MVE
inactive, uint32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VCVTT.F32.U32 Od,Om		
float16x8_t [arm_]vcvtq_n[_f16_s16](int16x8_t a, const	a -> Om	VCVT.F16.S16 Qd,Qm,imm6	Od -> result	MVE/NEON
int imm6)	1 <= imm6 <=		Q	
·	16			
float16x8_t [arm_]vcvtq_n[_f16_u16](uint16x8_t a,	a -> Qm	VCVT.F16.U16 Qd,Qm,imm6	Qd -> result	MVE/NEON
const int imm6)	1 <= imm6 <=			
float22x4 + [arm lyouta n[f22 s22]/int22x4 + a const	16 a -> Qm	VCVT.F32.S32 Qd,Qm,imm6	Qd -> result	MVE/NEON
float32x4_t [_arm_]vcvtq_n[_f32_s32](int32x4_t a, const int imm6)	a -> QIII 1 <= imm6 <=	VCV1.F32.S32 Qd,QIII,IIIIIII	Qu -> resuit	WIVE/INEOIN
	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	VD (GD DO F	0.1	1000
float16x8_t [_arm_]vcvtq_m_n[_f16_s16](float16x8_t	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
inactive, int16x8_t a, const int imm6, mve_pred16_t p)	a -> Qm 1 <= imm6 <=	VCVTT.F16.S16 Qd,Qm,imm6		
	16			
	p -> Rp		<u> </u>	<u> </u>
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 p -> Rp			
l	U = 2 IV II		1	MATE
float32x4 t [arm]vevtq m n[f32 s32](float32v4 t		VMSR P0 Rn	Od -> result	I IVI V E.
float32x4_t [_arm_]vcvtq_m_n[_f32_s32](float32x4_t inactive, int32x4_t a, const int imm6, mye_pred16_t p)	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
float32x4_t [_arm_]vcvtq_m_n[_f32_s32](float32x4_t inactive, int32x4_t a, const int imm6, mve_pred16_t p)			Qd -> result	MVE
	inactive -> Qd a -> Qm	VPST	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [_arm_]vcvtq_m_n[_f32_u32](float32x4_t inactive, uint32x4_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm6 <= 32 p -> Rp	VMSR P0,Rp VPST VCVTT.F32.U32 Qd,Qm,imm6	Qd -> result	MVE
int16x8_t [arm_]vcvtq_s16_f16(float16x8_t a)	a -> Qm	VCVT.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vcvtq_s32_f32(float32x4_t a)	a -> Qm	VCVT.S32.F32 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vcvtq_u16_f16(float16x8_t a)	a -> Qm	VCVT.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vcvtq_u32_f32(float32x4_t a) int16x8_t [_arm_]vcvtq_m[_s16_f16](int16x8_t inactive, float16x8_t a, mve_pred16_t p)	a -> Qm inactive -> Qd a -> Qm p -> Rp	VCVT.U32.F32 Qd,Qm VMSR P0,Rp VPST VCVTT.S16.F16 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE
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 p -> Rp	VMSR P0,Rp VPST VCVTT.U16.F16 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vcvtq_m[_u32_f32](uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.U32.F32 Qd,Qm	Qd -> result	MVE
int16x8_t [_arm_]vcvtq_n_s16_f16(float16x8_t a, const int imm6)	a -> Qm 1 <= imm6 <= 16	VCVT.S16.F16 Qd,Qm,imm6	Qd -> result	MVE/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 p -> Rp	VMSR P0,Rp VPST VCVTT.S16.F16 Qd,Qm,imm6	Qd -> result	MVE
int32x4_t [_arm_]vcvtq_m_n[_s32_f32](int32x4_t inactive, float32x4_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm6 <= 32 p -> Rp	VMSR P0,Rp VPST VCVTT.S32.F32 Qd,Qm,imm6	Qd -> result	MVE
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 p -> Rp	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)	inactive -> Qd a -> Qm 1 <= imm6 <= 32 p -> Rp	VMSR P0,Rp VPST VCVTT.U32.F32 Qd,Qm,imm6	Qd -> result	MVE
float16x8_t [_arm_]vrndq[_f16](float16x8_t a)	a -> Qm	VRINTZ.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vrndq[_f32](float32x4_t a) float16x8_t [_arm_]vrndq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	a -> Qm inactive -> Qd a -> Qm p -> Rp	VRINTZ.F32 Qd,Qm VMSR P0,Rp VPST VRINTZT.F16 Qd,Qm	Qd -> result Qd -> result	MVE/NEON 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_]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 p -> Rp	VMSR P0,Rp VPST VRINTNT.F32 Qd,Qm	Qd -> result	MVE
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float32x4_t [_arm_]vrndmq_m[_f32](float32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VRINTMT.F32 Qd,Qm		
float16x8_t [arm_]vrndpq[_f16](float16x8_t a)	a -> Qm	VRINTP.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vrndpq[_f32](float32x4_t a)	a -> Qm	VRINTP.F32 Qd,Qm	Qd -> result	MVE/NEON MVE
float16x8_t [arm_]vrndpq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, moatroxo_t a, mive_predio_t p)	p -> Rp	VRINTPT.F16 Qd,Qm		
float32x4_t [arm_]vrndpq_m[_f32](float32x4_t	inactive -> Od	VMSR P0,Rp	Od -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VRINTPT.F32 Qd,Qm		
float16x8_t [arm_]vrndaq[_f16](float16x8_t a)	a -> Qm	VRINTA.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [_arm_]vrndaq[_f32](float32x4_t a)	a -> Qm	VRINTA.F32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [arm_]vrndaq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
noatroxo_t a, nive_predio_t p)	p -> Rp	VRINTAT.F16 Qd,Qm		
float32x4_t [arm_]vrndaq_m[_f32](float32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
float32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Q	
	p -> Rp	VRINTAT.F32 Qd,Qm		
float16x8_t [arm_]vrndxq[_f16](float16x8_t a)	a -> Qm	VRINTX.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [arm_]vrndxq[_f32](float32x4_t a)	a -> Qm	VRINTX.F32 Qd,Qm	Qd -> result	MVE/NEON
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		
fl422-4 + f l l	p -> Rp inactive -> Od	VRINTXT.F16 Qd,Qm	0.1 >16	MVE
float32x4_t [_arm_]vrndxq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, moats2x4_t a, mive_pred10_t p)	p -> Rp	VRINTXT.F32 Od,Om		
int8x16 t [arm]vandq[s8](int8x16 t a, int8x16 t b)	a -> Qn	VAND Qd,Qn,Qm	Od -> result	MVE/NEON
	b -> Qm		Q	
int16x8_t [arm_]vandq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
	b -> Qm			
int32x4_t [arm_]vandq[_s32](int32x4_t a, int32x4_t b)	a -> Qn	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
1.0.16.1	b -> Qm	VIIIVE O LO O	0.1) A TEATRON
uint8x16_t [arm_]vandq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vandq[_u16](uint16x8_t a, uint16x8_t	a -> Qn	VAND Qd,Qn,Qm	Od -> result	MVE/NEON
b)	b -> Qm	VAIVD Qu,Qii,Qiii	Qu -> resuit	WIVE/NEON
uint32x4_t [arm_]vandq[_u32](uint32x4_t a, uint32x4_t	a -> Qn	VAND Qd,Qn,Qm	Od -> result	MVE/NEON
b)	b -> Qm			
float16x8_t [arm_]vandq[_f16](float16x8_t a,	a -> Qn	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t b)	b -> Qm			
float32x4_t [arm_]vandq[_f32](float32x4_t a,	a -> Qn	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t b)	b-> Qm	VMCD DO D.	Od > monule	MVE
int8x16_t [_arm_]vandq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> On	VMSR P0,Rp VPST	Qd -> result	MVE
intox10_t a, intox10_t b, inve_pred10_t p)	b -> Qm	VANDT Qd,Qn,Qm		
	p -> Rp			
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		
	p -> Rp	VILLOR DO D	0.1	\
int32x4_t [_arm_]vandq_m[_s32](int32x4_t inactive,	inactive -> Qd a -> Qn	VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t a, int32x4_t b, mve_pred16_t p)	b -> Qm	VANDT Qd,Qn,Qm		
	p -> Rp	VIII Qu,Qii,Qiii		
uint8x16_t [arm_]vandq_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VANDT Qd,Qn,Qm		
1.150.15	p -> Rp	VILLED DO D	0.1	\
uint16x8_t [arm_]vandq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
umittoxo_t a, umittoxo_t b, mve_pred1o_t p)	a -> Qn b -> Om	VANDT Qd,Qn,Qm		
	p -> Rp			1
uint32x4_t [arm_]vandq_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	-	1
	b -> Qm	VANDT Qd,Qn,Qm		1
	p -> Rp	19 49 P P P		
float16x8_t [_arm_]vandq_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 VANDT Od On Om		1
	b -> Qm p -> Rp	VANDT Qd,Qn,Qm		1
		VACD DO D.	0.1 >16	MVE
float32x4 t [arm]vanda ml f32](float32x4 t inactive	inactive -> Od	I VIVISK PU KD	()() => recitir	
float32x4_t [_arm_]vandq_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
float32x4_t [_arm_]vandq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)			Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
int16x8_t [_arm_]vbicq[_n_s16](int16x8_t a, const int imm)	a -> Qda imm in AdvSIMDExpa ndImm	VBIC.I16 Qda,#imm	Qda -> result	MVE
int32x4_t [_arm_]vbicq[_n_s32](int32x4_t a, const int imm)	a -> Qda imm in AdvSIMDExpa ndImm	VBIC.I32 Qda,#imm	Qda -> result	MVE
uint16x8_t [_arm_]vbicq[_n_u16](uint16x8_t a, const int imm)	a -> Qda imm in AdvSIMDExpa ndImm	VBIC.I16 Qda,#imm	Qda -> result	MVE
uint32x4_t [_arm_]vbicq[_n_u32](uint32x4_t a, const int imm)	a -> Qda imm in AdvSIMDExpa ndImm	VBIC.I32 Qda,#imm	Qda -> result	MVE
int16x8_t [_arm_]vbicq_m_n[_s16](int16x8_t a, const int imm, mve_pred16_t p)	a -> Qda imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VBICT.I16 Qda,#imm	Qda -> result	MVE
int32x4_t [_arm_]vbicq_m_n[_s32](int32x4_t a, const int imm, mve_pred16_t p)	a -> Qda imm in AdvSIMDExpa ndImm p -> Rp	VMSR P0,Rp VPST VBICT.I32 Qda,#imm	Qda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vbicq_m_n[_u16](uint16x8_t a, const int imm, mve_pred16_t p)	a -> Qda imm in AdvSIMDExpa ndImm	VMSR P0,Rp VPST VBICT.I16 Qda,#imm	Qda -> result	MVE
uint32x4_t [_arm_]vbicq_m_n[_u32](uint32x4_t a, const int imm, mve_pred16_t p)	p -> Rp a -> Qda imm in AdvSIMDExpa ndImm	VMSR P0,Rp VPST VBICT.I32 Qda,#imm	Qda -> result	MVE
int8x16_t [arm_]vbrsrq[_n_s8](int8x16_t a, int32_t b)	p -> Rp a -> Qn b -> Rm	VBRSR.8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [arm_]vbrsrq[_n_s16](int16x8_t a, int32_t b)	a -> Qn b -> Rm	VBRSR.16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [arm_]vbrsrq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VBRSR.32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [_arm_]vbrsrq[_n_u8](uint8x16_t a, int32_t b)	a -> Qn b -> Rm	VBRSR.8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [arm_]vbrsrq[_n_u16](uint16x8_t a, int32_t b) uint32x4_t [arm_]vbrsrq[_n_u32](uint32x4_t a, int32_t	a -> Qn b -> Rm a -> Qn	VBRSR.16 Qd,Qn,Rm VBRSR.32 Qd,Qn,Rm	Qd -> result Od -> result	MVE MVE
b) float16x8_t [arm_]vbrsrq[_n_f16](float16x8_t a, int32_t	b -> Rm a -> Qn	VBRSR.16 Qd,Qn,Rm	Od -> result	MVE
b) float32x4_t [_arm_]vbrsq[_n_f32](float32x4_t a, int32_t	b -> Rm a -> Qn	VBRSR.32 Qd,Qn,Rm	Qd -> result	MVE
b) int8x16_t [arm_]vbrsrq_m[_n_s8](int8x16_t inactive, int8x16_t a, int32_t b, mve_pred16_t p)	b -> Rm inactive -> Qd a -> Qn b -> Rm	VMSR P0,Rp VPST VBRSRT.8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [_arm_]vbrsrq_m[_n_s16](int16x8_t inactive, int16x8_t a, int32_t b, mve_pred16_t p)	p -> Rp inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [_arm_]vbrsrq_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 VBRSRT.32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [arm_]vbrsrq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [arm_]vbrsrq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [arm_]vbrsrq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [_arm_]vbrsrq_m[_n_f16](float16x8_t inactive, float16x8_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [_arm_]vbrsrq_m[_n_f32](float32x4_t inactive, float32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBRSRT.32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [arm_]veorq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]veorq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]veorq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]veorq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [_arm_]veorq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [_arm_]veorq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [_arm_]veorq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [_arm_]veorq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VEOR Qd,Qn,Qm	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]veorq_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	VEORT Qd,Qn,Qm		
	p -> Rp			
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	Od -> result	MVE
int32x4 t a, int32x4 t b, mve pred16 t p)	a -> Qn	VPST	Qu -> result	WYL
	b -> Qm	VEORT Qd,Qn,Qm		
	p -> Rp	2 . 2 . 2		
uint8x16_t [arm_]veorq_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn	VPST		
	b -> Qm	VEORT Qd,Qn,Qm		
	p -> Rp	VMCD DO D.	0.1 >16	MVE
uint16x8_t [_arm_]veorq_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
unitroxo_t a, unitroxo_t b, nive_pieuro_t p)	b -> Qm	VEORT Qd,Qn,Qm		
	p -> Rp	V LOKT Qu,Qii,Qiii		
uint32x4_t [arm_]veorq_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	VEORT Qd,Qn,Qm		
	p -> Rp			
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	VPST		
	b -> Qm	VEORT Qd,Qn,Qm		
float32x4_t [_arm_]veorq_m[_f32](float32x4_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
float32x4_t [arm_]veorq_m[_132](float32x4_t mactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn	VMSK PO,KP VPST	Qu -> resuit	IVI V E
noat32x4_t a, noat32x4_t b, inve_pred1o_t p)	b -> Qm	VEORT Qd,Qn,Qm		
	p -> Rp	, 2011		
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)	a -> Qm	VMOVLB.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vmovlbq[_u8](uint8x16_t a)	a -> Qm	VMOVLB.U8 Qd,Qm	Qd -> result	MVE
uint32x4_t [arm_]vmovlbq[_u16](uint16x8_t a)	a -> Qm	VMOVLB.U16 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vmovlbq_m[_s8](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VMOVLBT.S8 Qd,Qm	0.1 1:	MATE
int32x4_t [arm_]vmovlbq_m[_s16](int32x4_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
introxo_t a, inve_predio_t p)	p -> Rp	VMOVLBT.S16 Qd,Qm		
uint16x8_t [arm_]vmovlbq_m[_u8](uint16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
uint8x16_t a, mve_pred16_t p)	a -> Qm	VPST	Qu' y Tesun	
_ ,	p -> Rp	VMOVLBT.U8 Qd,Qm		
uint32x4_t [arm_]vmovlbq_m[_u16](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, uint16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	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)	a -> Qm	VMOVLT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vmovltq[_u8](uint8x16_t a)	a -> Qm	VMOVLT.U8 Qd,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vmovltq[_u16](uint16x8_t a)	a -> Qm	VMOVLT.U16 Qd,Qm	Qd -> result	MVE
int16x8_t [arm_]vmovltq_m[_s8](int16x8_t inactive,	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPS1 VMOVLTT.S8 Qd,Qm		
int32x4_t [arm_]vmovltq_m[_s16](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int16x8_t a, mve_pred16_t p)	a -> Qm	VPST	Qu > result	WY E
	p -> Rp	VMOVLTT.S16 Qd,Qm		
uint16x8_t [arm_]vmovltq_m[_u8](uint16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, mve_pred16_t p)	a -> Qm	VPST	-	
	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		1
1.0 16 (1)	p -> Rp	VMOVLTT.U16 Qd,Qm	01	NOTE
int8x16_t [arm_]vmovnbq[_s16](int8x16_t a, int16x8_t	a -> Qd	VMOVNB.I16 Qd,Qm	Qd -> result	MVE
b) ::::16::20::4	b -> Qm	VMOVND 122 O4 O	0414	MVE
int16x8_t [arm_]vmovnbq[_s32](int16x8_t a, int32x4_t b)	a -> Qd b > Om	VMOVNB.I32 Qd,Qm	Qd -> result	MVE
b) uint8x16_t [arm_]vmovnbq[_u16](uint8x16_t a,	b -> Qm	VMOVNB.I16 Qd,Qm	Qd -> result	MVE
uint8x16_t [arm_jvmovnbq[_u16](uint8x16_t a, uint16x8_t b)	a -> Qd b -> Qm	VIVIO VIND.110 Qu,Qm	Qu -> resuit	IVI V E
unitiono_t o)	a -> Qn	VMOVNB.I32 Qd,Qm	Qd -> result	MVE
uint16x8 t [arm]ymoynba[u32](uint16x8 t a			Zu > Icouit	1
uint16x8_t [arm_]vmovnbq[_u32](uint16x8_t a, uint32x4_t b)	-			
uint16x8_t [_arm_]vmovnbq[_u32](uint16x8_t a, uint32x4_t b) int8x16_t [_arm_]vmovnbq_m[_s16](int8x16_t a,	b -> Qm a -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b)	b->Qm	VMSR P0,Rp VPST	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vmovnbq_m[_s32](int16x8_t a,	a -> Od	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Q	
	p -> Rp	VMOVNBT.I32 Qd,Qm		
uint8x16_t [arm_]vmovnbq_m[_u16](uint8x16_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST VMOVNBT.I16 Qd,Qm		
uint16x8_t [arm_]vmovnbq_m[_u32](uint16x8_t a,	a -> Qd	VMSR P0,Rp	Od -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST	Qu'y resun	111.12
	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)	b -> Qm	VB 40 VB VE 122 0 1 0	0.1 1:	MALE
int16x8_t [arm_]vmovntq[_s32](int16x8_t a, int32x4_t b)	a -> Qd b -> Qm	VMOVNT.I32 Qd,Qm	Qd -> result	MVE
uint8x16_t [arm_]vmovntq[_u16](uint8x16_t a,	a -> Od	VMOVNT.I16 Qd,Qm	Od -> result	MVE
uint16x8_t b)	b -> Qm	(2,72	Q	
uint16x8_t [arm_]vmovntq[_u32](uint16x8_t a,	a -> Qd	VMOVNT.I32 Qd,Qm	Qd -> result	MVE
uint32x4_t b)	b -> Qm			
int8x16_t [arm_]vmovntq_m[_s16](int8x16_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPST 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	£ - 7 105an	
	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		
vint16v0 t [amm lymaymta mf v22]/vint16v0 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	Qd -> result	MVE
umiszki to, mio_prouto_t 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,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST 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	Qu'y resun	1,1,2
	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		
	p -> Rp	VMVNT Qd,Qm	0.1 1:	NOW
uint8x16_t [_arm_]vmvnq_m[_u8](uint8x16_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
unitox10_t a, nive_pied10_t p)	p -> Rp	VMVNT Qd,Qm		
uint16x8_t [arm_]vmvnq_m[_u16](uint16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
uint16x8_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VMVNT Qd,Qm		
uint32x4_t [arm_]vmvnq_m[_u32](uint32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t a, mve_pred16_t p)	a -> Qm	VPST		
int16x8_t [arm_]vmvnq_n_s16(const int16_t imm)	p -> Rp imm in	VMVNT Qd,Qm VMVN.I16 Qd,#imm	Od -> result	MVE
introxe_t [arm_]vinviiq_ii_sro(const intro_t iniiii)	AdvSIMDExpa	VMVN.116 Qu,#IIIIII	Qu -> lesuit	MIVE
	ndImm			
int32x4_t [arm_]vmvnq_n_s32(const int32_t imm)	imm in	VMVN.I32 Qd,#imm	Qd -> result	MVE
	AdvSIMDExpa			
	ndImm	VD GDV VI 6 O 1) am
uint16x8_t [arm_]vmvnq_n_u16(const uint16_t imm)	imm in	VMVN.I16 Qd,#imm	Qd -> result	MVE
	AdvSIMDExpa ndImm			
uint32x4 t [arm]vmvnq n u32(const uint32 t imm)	imm in	VMVN.I32 Qd,#imm	Qd -> result	MVE
umeszki-te [umi_]vmvnq_n_usz(const umesz_t mmn)	AdvSIMDExpa	71771132 Qu,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	Qu > result	III V E
	ndImm			
int16x8_t [arm_]vmvnq_m[_n_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
const int16_t imm, mve_pred16_t p)	imm in	VPST		
	AdvSIMDExpa	VMVNT.I16 Qd,#imm		
	ndImm			
int32x4_t [_arm_]vmvnq_m[_n_s32](int32x4_t inactive,	p -> Rp inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
const int32_t imm, mve_pred16_t p)	imm in	VPST	Qu -> Icsuit	141 4 1
, о_рьовьо_с ру	AdvSIMDExpa	VMVNT.I32 Qd,#imm		
	ndImm			
	p -> Rp		İ	1

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8 t [arm]vmvnq m[n u16](uint16x8 t	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
inactive, const uint16_t imm, mve_pred16_t p)	imm in	VPST	Q	
	AdvSIMDExpa ndImm	VMVNT.I16 Qd,#imm		
	p -> Rp			
uint32x4_t [arm_]vmvnq_m[_n_u32](uint32x4_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, const uint32_t imm, mve_pred16_t p)	imm in	VPST		
	AdvSIMDExpa ndImm	VMVNT.I32 Qd,#imm		
	p -> Rp			
mve_pred16_t [arm_]vpnot(mve_pred16_t a)	a -> Rp	VMSR P0,Rp	Rt -> result	MVE
		VPNOT VMRS Rt,P0		
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
mve_pred16_t p)	b -> Qm	VPSEL Qd,Qn,Qm	Qu -> result	WYL
	p -> Rp			
int32x4_t [arm_]vpselq[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qd -> result	MVE
mvc_prearo_t p)	p -> Rp	VI SEE Qu,Qii,Qiii		
int64x2_t [arm_]vpselq[_s64](int64x2_t a, int64x2_t b,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
mve_pred16_t p)	b -> Qm	VPSEL Qd,Qn,Qm		
uint8x16_t [arm_]vpselq[_u8](uint8x16_t a, uint8x16_t	p -> Rp a -> Qn	VMSR P0,Rp	Qd -> result	MVE
b, mve_pred16_t p)	b -> Qm	VPSEL Qd,Qn,Qm		
uint16x8 t[arm]vpselq[u16](uint16x8 t a,	p -> Rp	VMCD DO D.	Od -> result	MVE
uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qu -> resuit	MIVE
	p -> Rp			
uint32x4_t [_arm_]vpselq[_u32](uint32x4_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm p -> Rp	VPSEL Qd,Qn,Qm		
uint64x2_t [arm_]vpselq[_u64](uint64x2_t a,	a -> Qn	VMSR P0,Rp	Qd -> result	MVE
uint64x2_t b, mve_pred16_t p)	b -> Qm	VPSEL Qd,Qn,Qm		
float16x8_t [arm_]vpselq[_f16](float16x8_t a,	p -> Rp a -> On	VMSR P0,Rp	Od -> result	MVE
float16x8_t b, mve_pred16_t p)	b -> Qm	VPSEL Qd,Qn,Qm		
float32x4_t [_arm_]vpselq[_f32](float32x4_t a,	p -> Rp a -> Qn	VMSR P0,Rp	Od -> result	MVE
float32x4_t tarm_jvpscqt_132j(noat32x4_t a, float32x4_t b, mve_pred16_t p)	b -> Qm	VPSEL Qd,Qn,Qm	Qu -> resuit	WYL
	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,	a -> Qn	VORN Qd,Qn,Qm	Qd -> result	MVE
float32x4_t b) int8x16 t [arm vornq[s8](int8x16 t a, int8x16 t b)	b -> Qm	VODN 04 0 · O···	0.1	MATERIEON
int8x16_t [arm_jvornqt_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Om	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vornq[_s16](int16x8_t a, int16x8_t b)	a -> Qn	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
:::(22-4 + [] [-22](::(22-4 + - ::(22-4 + 1)	b -> Qm	VODN 04 0 0	0.1	MATERIEON
int32x4_t [arm_]vornq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vornq[_u8](uint8x16_t a, uint8x16_t	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)	b -> Qm	VOICIV Qu,Qii,Qiii	Qu -> resuit	WYE/NEON
uint32x4_t [arm_]vornq[_u32](uint32x4_t a, uint32x4_t	a -> Qn	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
b) float16x8_t [arm_]vornq_m[_f16](float16x8_t inactive,	b -> Qm inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn	VPST	2- > 100000	
	b -> Qm p -> Rp	VORNT Qd,Qn,Qm		
float32x4_t [arm_]vornq_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	VPST	-	
	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	VPST		
- / - 1 - 1/	b -> Qm	VORNT Qd,Qn,Qm		

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [_arm_]vornq_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 VORNT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vornq_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 VORNT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vornq_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 VORNT Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vornq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORNT Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [_arm_]vorrq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vorrq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vorrq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vorrq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [arm_]vorrq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vorrq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vorrq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vorrq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VORR Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [_arm_]vorrq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [_arm_]vorrq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [arm_]vorrq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [_arm_]vorrq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [arm_]vorrq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vorrq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [_arm_]vorrq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [_arm_]vorrq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VORRT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [arm_]vorrq[_n_s16](int16x8_t a, const int imm)	a -> Qda imm in AdvSIMDExpa ndImm	VORR.I16 Qda,#imm	Qda -> result	MVE
int32x4_t [arm_]vorrq[_n_s32](int32x4_t a, const int imm)	a -> Qda imm in AdvSIMDExpa ndImm	VORR.I32 Qda,#imm	Qda -> result	MVE
uint16x8_t [_arm_]vorrq[_n_u16](uint16x8_t a, const int imm)	a -> Qda imm in AdvSIMDExpa ndImm	VORR.I16 Qda,#imm	Qda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4 t [arm]vorrq[n u32](uint32x4 t a, const int	a -> Qda	VORR.I32 Qda,#imm	Qda -> result	MVE
imm)	imm in			
	AdvSIMDExpa			
intleve to some browns as at aleliantleve to constint	ndImm a -> Qda	VMCD DO Do	Odo > monite	MVE
int16x8_t [_arm_]vorrq_m_n[_s16](int16x8_t a, const int imm, mve_pred16_t p)	a -> Qda imm in	VMSR P0,Rp VPST	Qda -> result	MVE
mmi, mvc_pred10_t p)	AdvSIMDExpa	VORRT.I16 Qda,#imm		
	ndImm			
	p -> Rp			
int32x4_t [_arm_]vorrq_m_n[_s32](int32x4_t a, const int	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
imm, mve_pred16_t p)	imm in AdvSIMDExpa	VPST VORRT.I32 Qda,#imm		
	ndImm	VOKK1.132 Qua,πinini		
	p -> Rp			
uint16x8_t [arm_]vorrq_m_n[_u16](uint16x8_t a, const	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int imm, mve_pred16_t p)	imm in	VPST		
	AdvSIMDExpa ndImm	VORRT.I16 Qda,#imm		
	p -> Rp			
uint32x4_t [arm_]vorrq_m_n[_u32](uint32x4_t a, const	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int imm, mve_pred16_t p)	imm in	VPST		
	AdvSIMDExpa	VORRT.I32 Qda,#imm		
	ndImm			
int8x16_t [arm_]vqmovnbq[_s16](int8x16_t a,	p -> Rp a -> Qd	VQMOVNB.S16 Qd,Qm	Qd -> result	MVE
int16x8_t b) int16x8_t [arm]vgmovnbq[s32](int16x8_t a,	b -> Qm	VOMOVNIR COA O LO	Od -> result	MVE
int10x8_t [armjvqmovnbq[_s32](int10x8_t a, int32x4_t b)	a -> Qd b -> Qm	VQMOVNB.S32 Qd,Qm	Qu -> result	MVE
uint8x16_t [arm_]vqmovnbq[_u16](uint8x16_t a,	a -> Qd	VQMOVNB.U16 Qd,Qm	Qd -> result	MVE
uint16x8_t b)	b -> Qm	VOMOVNIR U22 O4 O	0.1 >16	MYE
uint16x8_t [arm_]vqmovnbq[_u32](uint16x8_t a, uint32x4_t b)	a -> Qd b -> Qm	VQMOVNB.U32 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vqmovnbq_m[_s16](int8x16_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
. 16.0 . 1	p -> Rp	VQMOVNBT.S16 Qd,Qm	0.1	MATE
int16x8_t [arm_]vqmovnbq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mio2x1_t o, mio_pred1o_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		
: 16.9 - 1	p -> Rp	VQMOVNBT.U16 Qd,Qm	0.1	MATE
uint16x8_t [arm_]vqmovnbq_m[_u32](uint16x8_t a, uint32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
umi32x4_t b, mvc_picu10_t p)	p -> Rp	VQMOVNBT.U32 Qd,Qm		
int8x16_t [arm_]vqmovntq[_s16](int8x16_t a, int16x8_t	a -> Qd	VQMOVNT.S16 Qd,Qm	Qd -> result	MVE
b)	b -> Qm			
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)	b -> Qm	VOMOUNT V22 O LO	0.1	ME
uint16x8_t [arm_]vqmovntq[_u32](uint16x8_t a, uint32x4_t b)	a -> Qd b -> Qm	VQMOVNT.U32 Qd,Qm	Qd -> result	MVE
int8x16_t [arm_]vqmovntq_m[_s16](int8x16_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST		
	p -> Rp	VQMOVNTT.S16 Qd,Qm		
int16x8_t [arm_]vqmovntq_m[_s32](int16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
uint8x16_t [arm_]vqmovntq_m[_u16](uint8x16_t a,	p -> Rp a -> Od	VQMOVNTT.S32 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
uint16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Qu > result	III V E
- / 1/	p -> Rp	VQMOVNTT.U16 Qd,Qm		
uint16x8_t [arm_]vqmovntq_m[_u32](uint16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t b, mve_pred16_t p)	b -> Qm	VPST		
uint8x16_t [arm_]vqmovunbq[_s16](uint8x16_t a,	p -> Rp a -> Qd	VQMOVNTT.U32 Qd,Qm VQMOVUNB.S16 Qd,Qm	Qd -> result	MVE
int16x8_t b)	a -> Qu b -> Qm	, Outo told paro for out	Qu -> icsuit	141 A 17
uint16x8_t [arm_]vqmovunbq[_s32](uint16x8_t a,	a -> Qd	VQMOVUNB.S32 Qd,Qm	Qd -> result	MVE
int32x4_t b) uint8x16_t [arm_]vqmovunbq_m[_s16](uint8x16_t a,	b -> Qm a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t b, mve_pred16_t p)	b -> Qm	VPST	Qu -> Iesuit	141.415
	p -> Rp	VQMOVUNBT.S16 Qd,Qm		
uint16x8_t [arm_]vqmovunbq_m[_s32](uint16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, mve_pred16_t p)	b -> Qm	VPST		1
uint8x16_t [arm_]vqmovuntq[_s16](uint8x16_t a,	p -> Rp a -> Qd	VQMOVUNBT.S32 Qd,Qm VQMOVUNT.S16 Qd,Qm	Qd -> result	MVE
		VUNIOVUNI SID UA UM	UG -> resuit	I IVI V C.

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [_arm_]vqmovuntq[_s32](uint16x8_t a, int32x4 t b)	a -> Qd b -> Qm	VQMOVUNT.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [_arm_]vqmovuntq_m[_s16](uint8x16_t a, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVUNTT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [arm_]vqmovuntq_m[_s32](uint16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
int8x16_t [_arm_]vqrshlq[_n_s8](int8x16_t a, int32_t b)	p -> Rp a -> Qda b -> Rm	VQMOVUNTT.S32 Qd,Qm VQRSHL.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [arm_]vqrshlq[_n_s16](int16x8_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vqrshlq[_n_s32](int32x4_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vqrshlq[_n_u8](uint8x16_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [arm_]vqrshlq[_n_u16](uint16x8_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vqrshlq[_n_u32](uint32x4_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [_arm_]vqrshlq_m_n[_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [arm_]vqrshlq_m_n[_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vqrshlq_m_n[_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vqrshlq_m_n[_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [_arm_]vqrshlq_m_n[_u16](uint16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vqrshlq_m_n[_u32](uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [arm_]vqrshlq[_s8](int8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VQRSHL.S8 Qd,Qm,Qn	Qd -> result	MVE/NEON
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
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

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

101809

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [_arm_]vqshlq_m_n[_s16](int16x8_t inactive, int16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qn 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VQSHLT.S16 Qd,Qn,#imm	Qd -> result	MVE
int32x4_t [_arm_]vqshlq_m_n[_s32](int32x4_t inactive, int32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qn 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VQSHLT.S32 Qd,Qn,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqshlq_m_n[_u8](uint8x16_t inactive, uint8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qn 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VQSHLT.U8 Qd,Qn,#imm	Qd -> result	MVE
uint16x8_t [arm_]vqshlq_m_n[_u16](uint16x8_t inactive, uint16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qn 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VQSHLT.U16 Qd,Qn,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vqshlq_m_n[_u32](uint32x4_t inactive, uint32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qn 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VQSHLT.U32 Qd,Qn,#imm	Qd -> result	MVE
int8x16_t [arm_]vqshlq_r[_s8](int8x16_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [arm_]vqshlq_r[_s16](int16x8_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vqshlq_r[_s32](int32x4_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [arm_]vqshlq_r[_u8](uint8x16_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [arm_]vqshlq_r[_u16](uint16x8_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vqshlq_r[_u32](uint32x4_t a, int32_t b)	a -> Qda b -> Rm	VQSHL.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [arm_]vqshlq_m_r[_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQSHLT.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [arm_]vqshlq_m_r[_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQSHLT.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [arm_]vqshlq_m_r[_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQSHLT.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vqshlq_m_r[_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQSHLT.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [_arm_]vqshlq_m_r[_u16](uint16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQSHLT.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vqshlq_m_r[_u32](uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQSHLT.U32 Qda,Rm	Qda -> result	MVE
uint8x16_t [arm_]vqshluq[_n_s8](int8x16_t a, const int imm)	a -> Qn 0 <= imm <= 7	VQSHLU.S8 Qd,Qn,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqshluq[_n_s16](int16x8_t a, const int imm)	a -> Qn 0 <= imm <= 15	VQSHLU.S16 Qd,Qn,#imm	Qd -> result	MVE
uint32x4_t [arm_]vqshluq[_n_s32](int32x4_t a, const int imm)	a -> Qn 0 <= imm <= 31	VQSHLU.S32 Qd,Qn,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqshluq_m[_n_s8](uint8x16_t inactive, int8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qn 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VQSHLUT.S8 Qd,Qn,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqshluq_m[_n_s16](uint16x8_t inactive, int16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qn 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VQSHLUT.S16 Qd,Qn,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vqshluq_m[_n_s32](uint32x4_t inactive, int32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qn 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VQSHLUT.S32 Qd,Qn,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [arm_]vqshrnbq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQSHRNB.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vqshrnbq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQSHRNB.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqshrnbq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQSHRNB.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqshrnbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQSHRNB.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vqshrnbq_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 VQSHRNBT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vqshrnbq_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 VQSHRNBT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vqshrnbq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQSHRNBT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vqshrnbq_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 VQSHRNBT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vqshrntq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VQSHRNT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vqshrntq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VQSHRNT.S32 Qd,Qm,#imm	Qd -> result	MVE
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vqshrunbq_m[_n_s32](uint16x8_t a,	a -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t b, const int imm, mve_pred16_t p)	b -> Qm	VPST		
	1 <= imm <=	VQSHRUNBT.S32 Od,Om,#imm		
	16 p -> Rp	Qa,Qiii,#iiiiiii		
uint8x16_t [arm_]vqshruntq[_n_s16](uint8x16_t a,	a -> Qd	VQSHRUNT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t b, const int imm)	b -> Qm			
	1 <= imm <= 8			
uint16x8_t [arm_]vqshruntq[_n_s32](uint16x8_t a,	a -> Qd	VQSHRUNT.S32 Qd,Qm,#imm	Qd -> result	MVE
nt32x4_t b, const int imm)	b -> Qm 1 <= imm <=			
	16			
uint8x16_t [arm_]vqshruntq_m[_n_s16](uint8x16_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	VQSHRUNTT.S16		
1 16 0 at 1 1 a 1 a 2017 i at 6 0 a	p -> Rp	Qd,Qm,#imm	0.1	MATE
uint16x8_t [_arm_]vqshruntq_m[_n_s32](uint16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Om	VMSR P0,Rp VPST	Qd -> result	MVE
int32x4_t b, const int inini, inve_pred1o_t p)	1 <= imm <=	VQSHRUNTT.S32		
	16	Qd,Qm,#imm		
	p -> Rp			
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)	a -> Qm	VREV16.8 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vrev16q_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VREV16T.8 Qd,Qm		
uint8x16_t [arm_]vrev16q_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
uint8x16_t a, mve_pred16_t p)	a -> Qm	VPST	Qu => result	IVIVE
umtonro_t u, mre_prouro_t p	p -> Rp	VREV16T.8 Qd,Qm		
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
float16x8_t [arm_]vrev32q[_f16](float16x8_t a)	a -> Qm	VREV32.16 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [_arm_]vrev32q_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp VPST	Qd -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VREV32T.8 Qd,Qm		
int16x8_t [arm_]vrev32q_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Od -> result	MVE
int16x8_t a, mve_pred16_t p)	a -> Qm	VPST	Qu > result	III V E
_ / _11/	p -> Rp	VREV32T.16 Qd,Qm		
uint8x16_t [arm_]vrev32q_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
: 16 0 · f	p -> Rp	VREV32T.8 Qd,Qm	0.1	NOTE
uint16x8_t [arm_]vrev32q_m[_u16](uint16x8_t inactive, uint16x8 t a, mve pred16 t p)	inactive -> Qd a -> Qm	VMSR P0,Rp VPST	Qd -> result	MVE
mactive, unitroxo_t a, nive_predio_t p)	p -> Rp	VREV32T.16 Qd,Qm		
float16x8_t [arm_]vrev32q_m[_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm	VPST		·
	p -> Rp	VREV32T.16 Qd,Qm		
int8x16_t [arm_]vrev64q[_s8](int8x16_t a)	a -> Qm	VREV64.8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [arm_]vrev64q[_s16](int16x8_t a)	a -> Qm	VREV64.16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vrev64q[_s32](int32x4_t a)	a -> Qm	VREV64.32 Qd,Qm	Qd -> result	MVE/NEON
uint8x16_t [arm_]vrev64q[_u8](uint8x16_t a) uint16x8_t [arm_]vrev64q[_u16](uint16x8_t a)	a -> Qm	VREV64.8 Qd,Qm VREV64.16 Qd,Qm	Qd -> result Qd -> result	MVE/NEON MVE/NEON
uint16x8_t [arm_]vrev64q[_u16](uint16x8_t a) uint32x4_t [arm_]vrev64q[_u32](uint32x4_t a)	a -> Qm a -> Qm	VREV64.16 Qd,Qm VREV64.32 Qd,Qm	Qd -> result	MVE/NEON MVE/NEON
float16x8_t [arm_]vrev64q[_d32](dint32x4_t a)	a -> Qm	VREV64.32 Qd,Qm	Od -> result	MVE/NEON MVE/NEON
float32x4_t [arm_]vrev64q[_f32](float32x4_t a)	a -> Qm	VREV64.32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [arm_]vrev64q_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, mve_pred16_t p)	a -> Qm	VPST	1	
	p -> Rp	VREV64T.8 Qd,Qm	1	
int16x8_t [arm_]vrev64q_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, mve_pred16_t p)	a -> Qm	VPST VPEV64T 16 Od Om		
int32x4_t [arm_]vrev64q_m[_s32](int32x4_t inactive,	p -> Rp inactive -> Qd	VREV64T.16 Qd,Qm VMSR P0,Rp	Qd -> result	MVE
int32x4_t [armjvrev64q_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	a -> Qm	VMSK PO,KP VPST	Qu -> resuit	IVI V E
······································	p -> Rp	VREV64T.32 Qd,Qm		
uint8x16_t [arm_]vrev64q_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, mve_pred16_t p)	a -> Qm	VPST		
	p -> Rp	VREV64T.8 Qd,Qm		
uint16x8_t [arm_]vrev64q_m[_u16](uint16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
	a -> Qm	VPST	1	
inactive, uint16x8_t a, mve_pred16_t p)	-			
inactive, uint16x8_t a, mve_pred16_t p)	p -> Rp	VREV64T.16 Qd,Qm	04 5 10	Myr
	-		Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [arm_]vrev64q_m[_f16](float16x8_t	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
inactive, float16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VPST VREV64T.16 Qd,Qm		
float32x4 t [arm]vrev64q m[f32](float32x4 t	inactive -> Qd	VMSR P0.Rp	Od -> result	MVE
inactive, float32x4_t a, mve_pred16_t p)	a -> Qm	VPST	Q	
1.0.16.17	p -> Rp	VREV64T.32 Qd,Qm	0.1) (T
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	VRSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [_arm_]vrshlq[_n_s32](int32x4_t a, int32_t b)	b -> Rm a -> Qda	VRSHL.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [arm_]vrshlq[_n_u8](uint8x16_t a, int32_t b)	b -> Rm a -> Qda	VRSHL.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [arm_]vrshlq[_n_u16](uint16x8_t a, int32_t	b -> Rm a -> Qda	VRSHL.U16 Qda,Rm	Qda -> result	MVE
b) uint32x4_t [arm_]vrshlq[_n_u32](uint32x4_t a, int32_t	b -> Rm a -> Oda	VRSHL.U32 Qda,Rm	Oda -> result	MVE
b)	b->Rm	7		
int8x16_t [_arm_]vrshlq_m_n[_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
	p -> Rp	VRSHLT.S8 Qda,Rm		
int16x8_t [arm_]vrshlq_m_n[_s16](int16x8_t a, int32_t	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
b, mve_pred16_t p)	b -> Rm p -> Rp	VPST VRSHLT.S16 Qda,Rm		
int32x4 t [arm]vrshlq m n[s32](int32x4 t a, int32 t	a -> Oda	VMSR P0,Rp	Oda -> result	MVE
b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VRSHLT.S32 Qda,Rm		
uint8x16_t [arm_]vrshlq_m_n[_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
misz_t b, mve_pied10_t p)	p -> Rp	VRSHLT.U8 Qda,Rm		
uint16x8_t [arm_]vrshlq_m_n[_u16](uint16x8_t a,	a -> Qda	VMSR P0,Rp	Qda -> result	MVE
int32_t b, mve_pred16_t p)	b -> Rm	VPST		
	p -> Rp	VRSHLT.U16 Qda,Rm		
uint32x4_t [_arm_]vrshlq_m_n[_u32](uint32x4_t a,	a -> Qda b -> Rm	VMSR P0,Rp VPST	Qda -> result	MVE
int32_t b, mve_pred16_t p)	p -> Rm p -> Rp	VRSHLT.U32 Qda,Rm		
int8x16_t [arm_]vrshlq[_s8](int8x16_t a, int8x16_t b)	a -> Qm	VRSHL.S8 Qd,Qm,Qn	Qd -> result	MVE/NEON
int16x8_t [arm_]vrshlq[_s16](int16x8_t a, int16x8_t b)	b -> Qn a -> Qm	VRSHL.S16 Qd,Qm,Qn	Qd -> result	MVE/NEON
int32x4_t [arm_]vrshlq[_s32](int32x4_t a, int32x4_t b)	b -> Qn a -> Qm	VRSHL.S32 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint8x16_t [arm_]vrshlq[_u8](uint8x16_t a, int8x16_t b)	b -> Qn a -> Om	VRSHL.U8 Qd,Qm,Qn	Od -> result	MVE/NEON
	b->Qn			
uint16x8_t [arm_]vrshlq[_u16](uint16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VRSHL.U16 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint32x4_t [arm_]vrshlq[_u32](uint32x4_t a, int32x4_t b)	a -> Qm b -> On	VRSHL.U32 Qd,Qm,Qn	Qd -> result	MVE/NEON
int8x16_t [arm_]vrshlq_m[_s8](int8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qm b -> On	VPST VRSHLT.S8 Qd,Qm,Qn		
	p -> Rp	VKSIE1.56 Qu,Qiii,Qii		
int16x8_t [arm_]vrshlq_m[_s16](int16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qm b -> Qn	VPST VRSHLT.S16 Qd,Qm,Qn		
	p -> Rp			
int32x4_t [_arm_]vrshlq_m[_s32](int32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qm b -> Qn	VPST VRSHLT.S32 Qd,Qm,Qn		
	p -> Rp	VKS11L1.532 Qu,QIII,QII		
uint8x16_t [arm_]vrshlq_m[_u8](uint8x16_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qm	VPST		
	b -> Qn p -> Rp	VRSHLT.U8 Qd,Qm,Qn		
uint16x8_t [arm_]vrshlq_m[_u16](uint16x8_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qm b -> Qn	VPST VRSHLT.U16 Qd,Qm,Qn		
	p -> Qn	, KSIILI. O IO Qu,QIII,QII		
uint32x4_t [arm_]vrshlq_m[_u32](uint32x4_t inactive,	inactive -> Qd	VMSR P0,Rp	Qd -> result	MVE
uint32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qm	VPST		
	b -> Qn p -> Rp	VRSHLT.U32 Qd,Qm,Qn		
	a -> Qda	VSHLC Qda,Rdm,#imm	Qda -> result	MVE
int8x16 t arm vshlcal s8l(int8x16 t a. uint32 t * h				
int8x16_t [_arm_]vshlcq[_s8](int8x16_t a, uint32_t * b, const int imm)	*b -> Rdm 1 <= imm <=		Rdm -> *b	

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [arm_]vshlcq[_s16](int16x8_t a, uint32_t * b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
int32x4_t [_arm_]vshlcq[_s32](int32x4_t a, uint32_t * b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint8x16_t [_arm_]vshlcq[_u8](uint8x16_t a, uint32_t * b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint16x8_t [_arm_]vshlcq[_u16](uint16x8_t a, uint32_t * b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint32x4_t [_arm_]vshlcq[_u32](uint32x4_t a, uint32_t * b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
int8x16_t [arm_]vshlcq_m[_s8](int8x16_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
int16x8_t [arm_]vshlcq_m[_s16](int16x8_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
int32x4_t [arm_]vshlcq_m[_s32](int32x4_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint8x16_t [_arm_]vshlcq_m[_u8](uint8x16_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint16x8_t [_arm_]vshlcq_m[_u16](uint16x8_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
uint32x4_t [_arm_]vshlcq_m[_u32](uint32x4_t a, uint32_t * b, const int imm, mve_pred16_t p)	a -> Qda *b -> Rdm 1 <= imm <= 32 p -> Rp	VMSR P0,Rp VPST VSHLCT Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE
int16x8_t [arm_]vshllbq[_n_s8](int8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VSHLLB.S8 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshllbq[_n_s16](int16x8_t a, const int imm)	a -> Qm 1 <= imm <= 16	VSHLLB.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshllbq[_n_u8](uint8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VSHLLB.U8 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vshllbq[_n_u16](uint16x8_t a, const int imm)	a -> Qm 1 <= imm <= 16	VSHLLB.U16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [_arm_]vshllbq_m[_n_s8](int16x8_t inactive, int8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHLLBT.S8 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshllbq_m[_n_s16](int32x4_t inactive, int16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VSHLLBT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vshllbq_m[_n_u8](uint16x8_t inactive, uint8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VSHLLBT.U8 Qd,Qm,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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_]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
int8x16_t [arm_]vshlq[_s8](int8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VSHL.S8 Qd,Qm,Qn	Qd -> result	MVE/NEON
int16x8_t [arm_]vshlq[_s16](int16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VSHL.S16 Qd,Qm,Qn	Qd -> result	MVE/NEON
int32x4_t [arm_]vshlq[_s32](int32x4_t a, int32x4_t b)	a -> Qm b -> Qn	VSHL.S32 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint8x16_t [arm_]vshlq[_u8](uint8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VSHL.U8 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint16x8_t [arm_]vshlq[_u16](uint16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VSHL.U16 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint32x4_t [arm_]vshlq[_u32](uint32x4_t a, int32x4_t b)	a -> Qm b -> Qn	VSHL.U32 Qd,Qm,Qn	Qd -> result	MVE/NEON
int8x16_t [_arm_]vshlq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.S8 Qd,Qm,Qn	Qd -> result	MVE
int16x8_t [_arm_]vshlq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.S16 Qd,Qm,Qn	Qd -> result	MVE
int32x4_t [_arm_]vshlq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.S32 Qd,Qm,Qn	Qd -> result	MVE
uint8x16_t [_arm_]vshlq_m[_u8](uint8x16_t inactive, uint8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.U8 Qd,Qm,Qn	Qd -> result	MVE
uint16x8_t [arm_]vshlq_m[_u16](uint16x8_t inactive, uint16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.U16 Qd,Qm,Qn	Qd -> result	MVE
uint32x4_t [arm_]vshlq_m[_u32](uint32x4_t inactive, uint32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VSHLT.U32 Qd,Qm,Qn	Qd -> result	MVE
int8x16_t [_arm_]vshlq_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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
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 p -> Rp	VMSR P0,Rp VPST VSHLT.S16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [_arm_]vshlq_m_n[_s32](int32x4_t inactive, int32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VSHLT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vshlq_m_n[_u8](uint8x16_t inactive, uint8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VSHLT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vshlq_m_n[_u16](uint16x8_t inactive, uint16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VSHLT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [_arm_]vshlq_m_n[_u32](uint32x4_t inactive, uint32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VSHLT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vshlq_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 Oda,Rm	Qda -> result	MVE
int16x8_t [arm_]vshlq_m_r[_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [_arm_]vshlq_m_r[_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [_arm_]vshlq_m_r[_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [_arm_]vshlq_m_r[_u16](uint16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [_arm_]vshlq_m_r[_u32](uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [arm_]vrshrnbq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VRSHRNB.II6 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vrshrnbq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VRSHRNB.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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [arm_]vrshrnbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VRSHRNB.132 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vrshrnbq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNBT.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vrshrnbq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNBT.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vrshrnbq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNBT.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [arm_]vrshrnbq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNBT.132 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vrshrntq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VRSHRNT.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vrshrntq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 16	VRSHRNT.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.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [arm_]vrshrntq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNTT.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [arm_]vrshrntq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNTT.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [_arm_]vrshrntq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNTT.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [_arm_]vrshrntq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm 1 <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNTT.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [_arm_]vrshrq[_n_s8](int8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VRSHR.S8 Qd,Qm,#imm	Qd -> result	MVE/NEON
int16x8_t [_arm_]vrshrq[_n_s16](int16x8_t a, const int imm)	a -> Qm 1 <= imm <= 16	VRSHR.S16 Qd,Qm,#imm	Qd -> result	MVE/NEON
int32x4_t [_arm_]vrshrq[_n_s32](int32x4_t a, const int imm)	a -> Qm 1 <= imm <= 32	VRSHR.S32 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vrshrq[_n_u8](uint8x16_t a, const int imm)	a -> Qm 1 <= imm <= 8	VRSHR.U8 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint16x8_t [arm_]vrshrq[_n_u16](uint16x8_t a, const int imm)	a -> Qm 1 <= imm <= 16	VRSHR.U16 Qd,Qm,#imm	Qd -> result	MVE/NEON
uint32x4_t [arm_]vrshrq[_n_u32](uint32x4_t a, const int imm)	a -> Qm 1 <= imm <= 32	VRSHR.U32 Qd,Qm,#imm	Qd -> result	MVE/NEON
int8x16_t [arm_]vrshrq_m[_n_s8](int8x16_t inactive, int8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRT.S8 Qd,Qm,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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_]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
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

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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
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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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[_fl6](float16x8_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 7$	VMOV.U16 Rt,Qn[idx]	Rt -> result	MVE/NEON
float32_t [arm_]vgetq_lane[_f32](float32x4_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 3$	VMOV.32 Rt,Qn[idx]	Rt -> result	MVE/NEON
int8_t [arm_]vgetq_lane[_s8](int8x16_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 15$	VMOV.S8 Rt,Qn[idx]	Rt -> result	MVE/NEON
int16_t [arm_]vgetq_lane[_s16](int16x8_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 7$	VMOV.S16 Rt,Qn[idx]	Rt -> result	MVE/NEON
int32_t [arm_]vgetq_lane[_s32](int32x4_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 3$	VMOV.32 Rt,Qn[idx]	Rt -> result	MVE/NEON
int64_t [arm_]vgetq_lane[_s64](int64x2_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 1$	VMOV Rt1,Rt2,D(2*n+idx)	[Rt1,Rt2] -> result	MVE/NEON
uint8_t [arm_]vgetq_lane[_u8](uint8x16_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 15$	VMOV.U8 Rt,Qn[idx]	Rt -> result	MVE/NEON
uint16_t [arm_]vgetq_lane[_u16](uint16x8_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 7$	VMOV.U16 Rt,Qn[idx]	Rt -> result	MVE/NEON
uint32_t [arm_]vgetq_lane[_u32](uint32x4_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 3$	VMOV.32 Rt,Qn[idx]	Rt -> result	MVE/NEON
uint64_t [arm_]vgetq_lane[_u64](uint64x2_t a, const int idx)	$a \rightarrow Qn$ $0 <= idx <= 1$	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)	$a \rightarrow Rt$ $b \rightarrow Qd$ $0 \leftarrow idx \leftarrow 7$	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

101809

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [arm_]vctp64q_m(uint32_t a,	a -> Rn	VMSR P0,Rp	Rd -> result	MVE
mve_pred16_t p)	p -> Rp	VPST VCTPT.64 Rn		
		VMRS Rd,P0		
int8x16_t [_arm_]vuninitializedq_s8(void)			Qd -> result	MVE
int16x8_t [arm_]vuninitializedq_s16(void) int32x4_t [arm_]vuninitializedq_s32(void)			Qd -> result Od -> result	MVE 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
float16x8_t [arm_]vuninitializedq_f16(void)			Qd -> result	MVE
float32x4_t [arm_]vuninitializedq_f32(void)		1100	Qd -> result	MVE
int16x8_t [_arm_]vreinterpretq_s16[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [_arm_]vreinterpretq_s32[_s8](int8x16_t a) float32x4_t [_arm_]vreinterpretq_f32[_s8](int8x16_t a)	a -> Qd a -> Od	NOP NOP	Qd -> result Od -> result	MVE/NEON MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_s8](int8x16_t a)	a -> Qd a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vreinterpretq_u16[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [_arm_]vreinterpretq_s64[_s8](int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [_arm_]vreinterpretq_f16[_s8](int8x16_t a) int8x16_t [_arm_]vreinterpretq_s8[_s16](int16x8_t a)	a -> Qd a -> Qd	NOP NOP	Qd -> result Od -> result	MVE/NEON MVE/NEON
int32x4_t [_arm_]vreinterpretq_s32[_s16](int16x8_t a)	a -> Qd 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) float16x8_t [arm_]vreinterpretq_f16[_s16](int16x8_t a)	a -> Qd a -> Qd	NOP NOP	Qd -> result Qd -> result	MVE/NEON MVE/NEON
int8x16_t [arm_]vreinterpretq_s8[_s32](int32x4_t a)	a -> Qd a -> Od	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) uint64x2_t [_arm_]vreinterpretq_u64[_s32](int32x4_t a)	a -> Qd a -> Qd	NOP NOP	Qd -> result Qd -> result	MVE/NEON MVE/NEON
uint64x2_t [_arm_]vreinterpretq_u64[_s32](int32x4_t a) int64x2_t [_arm_]vreinterpretq_s64[_s32](int32x4_t a)	a -> Qd 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 a -> Qd	NOP NOP	Qd -> result Od -> result	MVE/NEON MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_f32](float32x4_t a)	Ì			
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 NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_u8](uint8x16_t a) int64x2_t [arm_]vreinterpretq_s64[_u8](uint8x16_t a)	a -> Qd a -> Qd	NOP	Qd -> result Qd -> result	MVE/NEON MVE/NEON
float16x8_t [_arm_]vreinterpretq_f16[_u8](uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [_arm_]vreinterpretq_s8[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [arm_]vreinterpretq_s16[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [arm_]vreinterpretq_s32[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [arm_]vreinterpretq_u8[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [arm_]vreinterpretq_u32[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [_arm_]vreinterpretq_u64[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [arm_]vreinterpretq_s64[_u16](uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_u16](uint16x8_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)				

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_s32[_u32](uint32x4_t	a -> Qd a -> Od	NOP	Od -> result	MVE/NEON
a)	a -> Qu	1101	Qu -> resuit	WIVE/ILON
uint8x16_t [arm_]vreinterpretq_u8[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [_arm_]vreinterpretq_u16[_u32](uint32x4_t	a -> Qd	NOP	Od -> result	MVE/NEON
a)	u > Qu	1101	Qu > resun	WYENEON
uint64x2_t [arm_]vreinterpretq_u64[_u32](uint32x4_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)			Q	
int64x2_t [arm_]vreinterpretq_s64[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8 t [arm]vreinterpretq f16[u32](uint32x4 t	a -> Qd	NOP	Od -> result	MVE/NEON
a)				
int8x16_t [arm_]vreinterpretq_s8[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [arm_]vreinterpretq_s16[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [arm_]vreinterpretq_s32[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_u64](uint64x2_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)				
uint8x16_t [arm_]vreinterpretq_u8[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_u64](uint64x2_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)				
uint32x4_t [arm_]vreinterpretq_u32[_u64](uint64x2_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)			1	
int64x2_t [arm_]vreinterpretq_s64[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_u64](uint64x2_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)			1	
int8x16_t [arm_]vreinterpretq_s8[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [arm_]vreinterpretq_s16[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [arm_]vreinterpretq_s32[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [_arm_]vreinterpretq_u8[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [arm_]vreinterpretq_u16[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [_arm_]vreinterpretq_u32[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [arm_]vreinterpretq_u64[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [arm_]vreinterpretq_f16[_s64](int64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [_arm_]vreinterpretq_s8[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [_arm_]vreinterpretq_s16[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [_arm_]vreinterpretq_s32[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [arm_]vreinterpretq_f32[_f16](float16x8_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)	0.1	NOR	0.1 >16	MATERIEON
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 > O4	NOP	Od > monule	MVE/NEON
a)	a -> Qd	NOP	Qd -> result	IVI V E/INEOIN
uint64x2_t [arm_]vreinterpretq_u64[_f16](float16x8_t	a -> Qd	NOP	Qd -> result	MVE/NEON
a)	a -> Qu	NOI	Qu => resuit	WIVE/INEON
int64x2_t [arm_]vreinterpretq_s64[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64_t [arm_]lsll(uint64_t value, int32_t shift)	value ->	LSLL RdaLo,RdaHi,Rm	[RdaHi,RdaLo]	MVE
unto	[RdaHi,RdaLo]	ESEE Ruaes, Ruarii, Riii	-> result	III V E
	shift -> Rm			
int64 t [arm]asrl(int64 t value, int32 t shift)	value ->	ASRL RdaLo,RdaHi,Rm	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]		-> result	
	shift -> Rm			
uint64_t [arm_]uqrshll(uint64_t value, int32_t shift)	value ->	UQRSHLL RdaLo,RdaHi,Rm	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]		-> result	
	shift -> Rm			
int64_t [arm_]sqrshrl(int64_t value, int32_t shift)	value ->	SQRSHRL RdaLo,RdaHi,Rm	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]		-> result	
	shift -> Rm			
uint64_t [arm_]uqshll(uint64_t value, const int shift)	value ->	UQSHLL RdaLo,RdaHi,#shift	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]		-> result	
winted a France Translation of Art 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 <= shift <= 32	IIDCIDI DAL DAMAN	marin para	MVE
uint64_t [arm_]urshrl(uint64_t value, const int shift)	value ->	URSHRL RdaLo,RdaHi,#shift	[RdaHi,RdaLo] -> result	MVE
	[RdaHi,RdaLo]		-> result	
int64_t [arm_]srshrl(int64_t value, const int shift)	1 <= shift <= 32 value ->	SRSHRL RdaLo,RdaHi,#shift	[RdaHi,RdaLo]	MVE
into+_t [arm_jsrsmr(into+_t value, const int smift)	[RdaHi,RdaLo]	SKORKE KUALO,KUARI,#SIIIT	-> result	IVIVE
	1 <= shift <= 32		-/ icsuit	
int64_t [arm_]sqshll(int64_t value, const int shift)	value ->	SQSHLL RdaLo,RdaHi,#shift	[RdaHi,RdaLo]	MVE
	[RdaHi,RdaLo]		-> result	
	1 <= shift <= 32			
uint32_t [arm_]uqrshl(uint32_t value, int32_t shift)	value -> Rda	UQRSHL Rda,Rm	Rda -> result	MVE
	shift -> Rm			

Arm MVE Intrinsics Reference 101809

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
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