

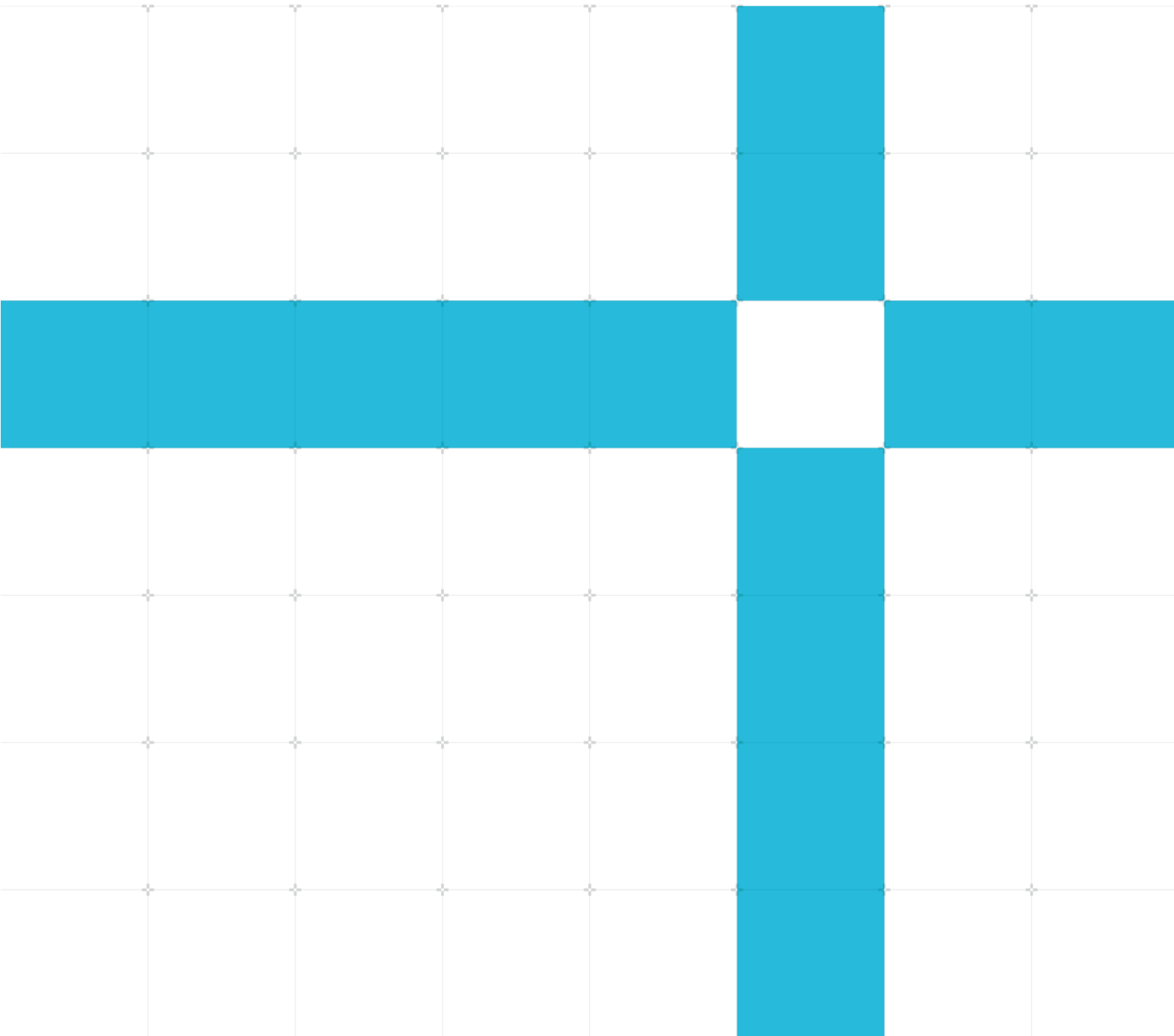


Arm MVE Intrinsics

Reference for ACLE Q2 2019

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

Issue Q219-00
101809



Arm MVE Intrinsic

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

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[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
int32x4_t [__arm_]vcreateq_s32(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
int64x2_t [__arm_]vcreateq_s64(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint8x16_t [__arm_]vcreateq_u8(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint16x8_t [__arm_]vcreateq_u16(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint32x4_t [__arm_]vcreateq_u32(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint64x2_t [__arm_]vcreateq_u64(uint64_t a, uint64_t b)	a -> [Rt, Rt2] b -> [Rt3, Rt4]	VMOV Qd[2],Qd[0],Rt3,Rt VMOV Qd[3],Qd[1],Rt4,Rt2	Qd -> result	MVE
uint8x16_t [__arm_]vddupq[_n]_u8(uint32_t a, const int imm)	a -> Rn imm in [1,2,4,8]	VDDUP.U8 Qd,Rn,imm	Qd -> result	MVE
uint16x8_t [__arm_]vddupq[_n]_u16(uint32_t a, const int imm)	a -> Rn imm in [1,2,4,8]	VDDUP.U16 Qd,Rn,imm	Qd -> result	MVE
uint32x4_t [__arm_]vddupq[_n]_u32(uint32_t a, const int imm)	a -> Rn imm in [1,2,4,8]	VDDUP.U32 Qd,Rn,imm	Qd -> result	MVE
uint8x16_t [__arm_]vddupq[_wb]_u8(uint32_t *a, const int imm)	*a -> Rn imm in [1,2,4,8]	VDDUP.U8 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [__arm_]vddupq[_wb]_u16(uint32_t *a, const int imm)	*a -> Rn imm in [1,2,4,8]	VDDUP.U16 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [__arm_]vddupq[_wb]_u32(uint32_t *a, const int imm)	*a -> Rn imm in [1,2,4,8]	VDDUP.U32 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [__arm_]vddupq[_m]_n_u8(uint8x16_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U8 Qd,Rn,imm	Qd -> result	MVE
uint16x8_t [__arm_]vddupq[_m]_n_u16(uint16x8_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U16 Qd,Rn,imm	Qd -> result	MVE
uint32x4_t [__arm_]vddupq[_m]_n_u32(uint32x4_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U32 Qd,Rn,imm	Qd -> result	MVE
uint8x16_t [__arm_]vddupq[_m]_wb_u8(uint8x16_t inactive, uint32_t *a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U8 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [__arm_]vddupq[_m]_wb_u16(uint16x8_t inactive, uint32_t *a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U16 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [__arm_]vddupq[_m]_wb_u32(uint32x4_t inactive, uint32_t *a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VDDUPT.U32 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [__arm_]vdwdupq[_n]_u8(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U8 Qd,Rn,Rm,imm	Qd -> result	MVE
uint16x8_t [__arm_]vdwdupq[_n]_u16(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U16 Qd,Rn,Rm,imm	Qd -> result	MVE
uint32x4_t [__arm_]vdwdupq[_n]_u32(uint32_t a, uint32_t b, const int imm)	a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U32 Qd,Rn,Rm,imm	Qd -> result	MVE
uint8x16_t [__arm_]vdwdupq[_wb]_u8(uint32_t *a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U8 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [__arm_]vdwdupq[_wb]_u16(uint32_t *a, uint32_t b, const int imm)	*a -> Rn b -> Rm imm in [1,2,4,8]	VDWDUP.U16 Qd,Rn,Rm,imm	Qd -> result Rn -> *a	MVE

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)	*a -> Rn imm in [1,2,4,8]	VIDUP.U8 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [__arm_]vidupq[_wb_]u16(uint32_t *a, const int imm)	*a -> Rn imm in [1,2,4,8]	VIDUP.U16 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [__arm_]vidupq[_wb_]u32(uint32_t *a, const int imm)	*a -> Rn imm in [1,2,4,8]	VIDUP.U32 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [__arm_]vidupq_m[_n_]u8(uint8x16_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U8 Qd,Rn,imm	Qd -> result	MVE
uint16x8_t [__arm_]vidupq_m[_n_]u16(uint16x8_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U16 Qd,Rn,imm	Qd -> result	MVE
uint32x4_t [__arm_]vidupq_m[_n_]u32(uint32x4_t inactive, uint32_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U32 Qd,Rn,imm	Qd -> result	MVE
uint8x16_t [__arm_]vidupq_m[_wb_]u8(uint8x16_t inactive, uint32_t *a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U8 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint16x8_t [__arm_]vidupq_m[_wb_]u16(uint16x8_t inactive, uint32_t *a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U16 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint32x4_t [__arm_]vidupq_m[_wb_]u32(uint32x4_t inactive, uint32_t *a, const int imm, mve_pred16_t p)	inactive -> Qd *a -> Rn imm in [1,2,4,8] p -> Rp	VMSR P0,Rp VPST VIDUPT.U32 Qd,Rn,imm	Qd -> result Rn -> *a	MVE
uint8x16_t [__arm_]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)	a -> Rt	VDUP.32 Qd,Rt	Qd -> result	MVE/NEON
int8x16_t [__arm_]vdupq_m[_n_s8](int8x16_t inactive, int8_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.8 Qd,Rt	Qd -> result	MVE
int16x8_t [__arm_]vdupq_m[_n_s16](int16x8_t inactive, int16_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.16 Qd,Rt	Qd -> result	MVE
int32x4_t [__arm_]vdupq_m[_n_s32](int32x4_t inactive, int32_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.32 Qd,Rt	Qd -> result	MVE
uint8x16_t [__arm_]vdupq_m[_n_u8](uint8x16_t inactive, uint8_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.8 Qd,Rt	Qd -> result	MVE
uint16x8_t [__arm_]vdupq_m[_n_u16](uint16x8_t inactive, uint16_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.16 Qd,Rt	Qd -> result	MVE
uint32x4_t [__arm_]vdupq_m[_n_u32](uint32x4_t inactive, uint32_t a, mve_pred16_t p)	inactive -> Qd a -> Rt p -> Rp	VMSR P0,Rp VPST VDUPT.32 Qd,Rt	Qd -> result	MVE
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_]vcmpqq[_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
<code>mve_pred16_t [__arm_]vcmpqq[_f32](float32x4_t a, float32x4_t b)</code>	a -> Qn b -> Qm	VCMP.F32 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_s8](int8x16_t a, int8x16_t b)</code>	a -> Qn b -> Qm	VCMP.I8 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_s16](int16x8_t a, int16x8_t b)</code>	a -> Qn b -> Qm	VCMP.I16 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_s32](int32x4_t a, int32x4_t b)</code>	a -> Qn b -> Qm	VCMP.I32 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_u8](uint8x16_t a, uint8x16_t b)</code>	a -> Qn b -> Qm	VCMP.I8 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_u16](uint16x8_t a, uint16x8_t b)</code>	a -> Qn b -> Qm	VCMP.I16 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_u32](uint32x4_t a, uint32x4_t b)</code>	a -> Qn b -> Qm	VCMP.I32 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_n_f16](float16x8_t a, float16_t b)</code>	a -> Qn b -> Rm	VCMP.F16 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_n_f32](float32x4_t a, float32_t b)</code>	a -> Qn b -> Rm	VCMP.F32 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_n_s8](int8x16_t a, int8_t b)</code>	a -> Qn b -> Rm	VCMP.I8 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_n_s16](int16x8_t a, int16_t b)</code>	a -> Qn b -> Rm	VCMP.I16 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_n_s32](int32x4_t a, int32_t b)</code>	a -> Qn b -> Rm	VCMP.I32 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_n_u8](uint8x16_t a, uint8_t b)</code>	a -> Qn b -> Rm	VCMP.I8 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_n_u16](uint16x8_t a, uint16_t b)</code>	a -> Qn b -> Rm	VCMP.I16 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq[_n_u32](uint32x4_t a, uint32_t b)</code>	a -> Qn b -> Rm	VCMP.I32 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.F16 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.F32 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.I8 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.I16 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.I32 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.I8 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.I16 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.I32 eq,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.F16 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.F32 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.I8 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>mve_pred16_t [__arm_]vcmpqq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.I16 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.I32 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.I8 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.I16 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqq_m[_n_u32](uint32x4_t a, uint32_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.I32 eq,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_f16](float16x8_t a, float16x8_t b)</code>	a -> Qn b -> Qm	VCMP.F16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_f32](float32x4_t a, float32x4_t b)</code>	a -> Qn b -> Qm	VCMP.F32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_s8](int8x16_t a, int8x16_t b)</code>	a -> Qn b -> Qm	VCMP.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_s16](int16x8_t a, int16x8_t b)</code>	a -> Qn b -> Qm	VCMP.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_s32](int32x4_t a, int32x4_t b)</code>	a -> Qn b -> Qm	VCMP.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_u8](uint8x16_t a, uint8x16_t b)</code>	a -> Qn b -> Qm	VCMP.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_u16](uint16x8_t a, uint16x8_t b)</code>	a -> Qn b -> Qm	VCMP.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_u32](uint32x4_t a, uint32x4_t b)</code>	a -> Qn b -> Qm	VCMP.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.F16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.F32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq_m[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.I8 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq_m[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.I16 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.I32 ne,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_n_f16](float16x8_t a, float16_t b)</code>	a -> Qn b -> Rm	VCMP.F16 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_n_f32](float32x4_t a, float32_t b)</code>	a -> Qn b -> Rm	VCMP.F32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_n_s8](int8x16_t a, int8_t b)</code>	a -> Qn b -> Rm	VCMP.I8 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpneq[_n_s16](int16x8_t a, int16_t b)</code>	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 VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.I32 ne,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [__arm_]vcmpneq_m[_n_u8](uint8x16_t a, uint8_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.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
<code>mve_pred16_t [__arm_]vcmpgeq[_n_s16](int16x8_t a, int16_t b)</code>	a -> Qn b -> Rm	VCMP.S16 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgeq[_n_s32](int32x4_t a, int32_t b)</code>	a -> Qn b -> Rm	VCMP.S32 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgeq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.F16 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgeq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.F32 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgeq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S8 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgeq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S16 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgeq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S32 ge,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq[_f16](float16x8_t a, float16x8_t b)</code>	a -> Qn b -> Qm	VCMP.F16 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq[_f32](float32x4_t a, float32x4_t b)</code>	a -> Qn b -> Qm	VCMP.F32 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq[_s8](int8x16_t a, int8x16_t b)</code>	a -> Qn b -> Qm	VCMP.S8 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq[_s16](int16x8_t a, int16x8_t b)</code>	a -> Qn b -> Qm	VCMP.S16 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq[_s32](int32x4_t a, int32x4_t b)</code>	a -> Qn b -> Qm	VCMP.S32 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.F16 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.F32 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.S8 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.S16 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.S32 gt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq[_n_f16](float16x8_t a, float16_t b)</code>	a -> Qn b -> Rm	VCMP.F16 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq[_n_f32](float32x4_t a, float32_t b)</code>	a -> Qn b -> Rm	VCMP.F32 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq[_n_s8](int8x16_t a, int8_t b)</code>	a -> Qn b -> Rm	VCMP.S8 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq[_n_s16](int16x8_t a, int16_t b)</code>	a -> Qn b -> Rm	VCMP.S16 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq[_n_s32](int32x4_t a, int32_t b)</code>	a -> Qn b -> Rm	VCMP.S32 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.F16 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.F32 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S8 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>mve_pred16_t [__arm_]vcmpgtq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S16 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpgtq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S32 gt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq[_f16](float16x8_t a, float16x8_t b)</code>	a -> Qn b -> Qm	VCMP.F16 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq[_f32](float32x4_t a, float32x4_t b)</code>	a -> Qn b -> Qm	VCMP.F32 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq[_s8](int8x16_t a, int8x16_t b)</code>	a -> Qn b -> Qm	VCMP.S8 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq[_s16](int16x8_t a, int16x8_t b)</code>	a -> Qn b -> Qm	VCMP.S16 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq[_s32](int32x4_t a, int32x4_t b)</code>	a -> Qn b -> Qm	VCMP.S32 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.F16 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.F32 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.S8 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.S16 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.S32 le,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq[_n_f16](float16x8_t a, float16_t b)</code>	a -> Qn b -> Rm	VCMP.F16 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq[_n_f32](float32x4_t a, float32_t b)</code>	a -> Qn b -> Rm	VCMP.F32 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq[_n_s8](int8x16_t a, int8_t b)</code>	a -> Qn b -> Rm	VCMP.S8 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq[_n_s16](int16x8_t a, int16_t b)</code>	a -> Qn b -> Rm	VCMP.S16 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq[_n_s32](int32x4_t a, int32_t b)</code>	a -> Qn b -> Rm	VCMP.S32 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.F16 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.F32 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S8 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S16 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpqleq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S32 le,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpltq[_f16](float16x8_t a, float16x8_t b)</code>	a -> Qn b -> Qm	VCMP.F16 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpltq[_f32](float32x4_t a, float32x4_t b)</code>	a -> Qn b -> Qm	VCMP.F32 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpltq[_s8](int8x16_t a, int8x16_t b)</code>	a -> Qn b -> Qm	VCMP.S8 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpltq[_s16](int16x8_t a, int16x8_t b)</code>	a -> Qn b -> Qm	VCMP.S16 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>mve_pred16_t [__arm_]vcmplttq[_s32](int32x4_t a, int32x4_t b)</code>	a -> Qn b -> Qm p -> Rp	VCMP.S32 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.F16 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.F32 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq_m[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.S8 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq_m[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.S16 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq_m[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.S32 lt,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq[_n_f16](float16x8_t a, float16_t b)</code>	a -> Qn b -> Rm	VCMP.F16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq[_n_f32](float32x4_t a, float32_t b)</code>	a -> Qn b -> Rm	VCMP.F32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq[_n_s8](int8x16_t a, int8_t b)</code>	a -> Qn b -> Rm	VCMP.S8 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq[_n_s16](int16x8_t a, int16_t b)</code>	a -> Qn b -> Rm	VCMP.S16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq[_n_s32](int32x4_t a, int32_t b)</code>	a -> Qn b -> Rm	VCMP.S32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq_m[_n_f16](float16x8_t a, float16_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.F16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq_m[_n_f32](float32x4_t a, float32_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.F32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq_m[_n_s8](int8x16_t a, int8_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S8 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq_m[_n_s16](int16x8_t a, int16_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S16 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmplttq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)</code>	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.S32 lt,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpcsq[_u8](uint8x16_t a, uint8x16_t b)</code>	a -> Qn b -> Qm	VCMP.U8 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpcsq[_u16](uint16x8_t a, uint16x8_t b)</code>	a -> Qn b -> Qm	VCMP.U16 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpcsq[_u32](uint32x4_t a, uint32x4_t b)</code>	a -> Qn b -> Qm	VCMP.U32 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpcsq_m[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.U8 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpcsq_m[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.U16 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpcsq_m[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)</code>	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VCMP.T.U32 cs,Qn,Qm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpcsq[_n_u8](uint8x16_t a, uint8_t b)</code>	a -> Qn b -> Rm	VCMP.U8 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpcsq[_n_u16](uint16x8_t a, uint16_t b)</code>	a -> Qn b -> Rm	VCMP.U16 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
<code>mve_pred16_t [__arm_]vcmpcsq[_n_u32](uint32x4_t a, uint32_t b)</code>	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 VCMP.T.U8 cs,Qn,Rm VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [__arm_]vcmpcsq_m[_n_u16](uint16x8_t a, uint16_t b, mve_pred16_t p)	a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.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 VCMP.T.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 -> Qm	VMINA.S8 Qda,Qm	Qda -> result	MVE
uint16x8_t [__arm_]vminaq[s16](uint16x8_t a, int16x8_t b)	a -> Qda b -> Qm	VMINA.S16 Qda,Qm	Qda -> result	MVE
uint32x4_t [__arm_]vminaq[s32](uint32x4_t a, int32x4_t b)	a -> Qda b -> Qm	VMINA.S32 Qda,Qm	Qda -> result	MVE
uint8x16_t [__arm_]vminaq_m[s8](uint8x16_t a, 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,Qm	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,Qm	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_]vminnmaq_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 VMINNMA.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 VMINNMA.F32 Qda,Qm	Qda -> result	MVE
float16_t [__arm_]vminnmvq[_f16](float16_t a, float16x8_t b)	a -> Rda b -> Qm	VMINNMA.F16 Rda,Qm	Rda -> result	MVE
float32_t [__arm_]vminnmvq[_f32](float32_t a, float32x4_t b)	a -> Rda b -> Qm	VMINNMA.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 VMINNMA.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 VMINNMA.F32 Rda,Qm	Rda -> result	MVE
float16_t [__arm_]vminnmavq[_f16](float16_t a, float16x8_t b)	a -> Rda b -> Qm	VMINNMA.F16 Rda,Qm	Rda -> result	MVE
float32_t [__arm_]vminnmavq[_f32](float32_t a, float32x4_t b)	a -> Rda b -> Qm	VMINNMA.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 VMINNMA.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 VMINNMA.F32 Rda,Qm	Rda -> result	MVE
int8x16_t [__arm_]vmaxq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMAX.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vmaxq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMAX.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vmaxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMAX.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vmaxq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMAX.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vmaxq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMAX.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vmaxq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VMAX.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vmaxq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vmaxq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vmaxq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm 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, int8x16_t b, mve_pred16_t p)	a -> Qda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXAT.S8 Qda,Qm	Qda -> result	MVE
uint16x8_t [__arm_]vmaxaq_m[_s16](uint16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXAT.S16 Qda,Qm	Qda -> result	MVE
uint32x4_t [__arm_]vmaxaq_m[_s32](uint32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXAT.S32 Qda,Qm	Qda -> result	MVE
int8_t [__arm_]vmaxvq[_s8](int8_t a, int8x16_t b)	a -> Rda b -> Qm	VMAXV.S8 Rda,Qm	Rda -> result	MVE
int16_t [__arm_]vmaxvq[_s16](int16_t a, int16x8_t b)	a -> Rda b -> Qm	VMAXV.S16 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vmaxvq[_s32](int32_t a, int32x4_t b)	a -> Rda b -> Qm	VMAXV.S32 Rda,Qm	Rda -> result	MVE
uint8_t [__arm_]vmaxvq[_u8](uint8_t a, uint8x16_t b)	a -> Rda b -> Qm	VMAXV.U8 Rda,Qm	Rda -> result	MVE
uint16_t [__arm_]vmaxvq[_u16](uint16_t a, uint16x8_t b)	a -> Rda b -> Qm	VMAXV.U16 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vmaxvq[_u32](uint32_t a, uint32x4_t b)	a -> Rda b -> Qm	VMAXV.U32 Rda,Qm	Rda -> result	MVE
int8_t [__arm_]vmaxvq_p[_s8](int8_t a, int8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.S8 Rda,Qm	Rda -> result	MVE
int16_t [__arm_]vmaxvq_p[_s16](int16_t a, int16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.S16 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vmaxvq_p[_s32](int32_t a, int32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.S32 Rda,Qm	Rda -> result	MVE
uint8_t [__arm_]vmaxvq_p[_u8](uint8_t a, uint8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.U8 Rda,Qm	Rda -> result	MVE
uint16_t [__arm_]vmaxvq_p[_u16](uint16_t a, uint16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.U16 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vmaxvq_p[_u32](uint32_t a, uint32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXVT.U32 Rda,Qm	Rda -> result	MVE
uint8_t [__arm_]vmaxavq[_s8](uint8_t a, int8x16_t b)	a -> Rda b -> Qm	VMAXAV.S8 Rda,Qm	Rda -> result	MVE
uint16_t [__arm_]vmaxavq[_s16](uint16_t a, int16x8_t b)	a -> Rda b -> Qm	VMAXAV.S16 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vmaxavq[_s32](uint32_t a, int32x4_t b)	a -> Rda b -> Qm	VMAXAV.S32 Rda,Qm	Rda -> result	MVE
uint8_t [__arm_]vmaxavq_p[_s8](uint8_t a, int8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXAVT.S8 Rda,Qm	Rda -> result	MVE
uint16_t [__arm_]vmaxavq_p[_s16](uint16_t a, int16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXAVT.S16 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vmaxavq_p[_s32](uint32_t a, int32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXAVT.S32 Rda,Qm	Rda -> result	MVE
float16x8_t [__arm_]vmaxnmq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VMAXNM.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [__arm_]vmaxnmq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VMAXNM.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vmaxnmq_m[_f16](float16x8_t a, inactive, float16x8_t b, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [__arm_]vmaxnmq_m[_f32](float32x4_t a, inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [__arm_]vmaxnmaq[_f16](float16x8_t a, float16x8_t b)	a -> Qda b -> Qm	VMAXNMA.F16 Qda,Qm	Qda -> result	MVE
float32x4_t [__arm_]vmaxnmaq[_f32](float32x4_t a, float32x4_t b)	a -> Qda b -> Qm	VMAXNMA.F32 Qda,Qm	Qda -> result	MVE
float16x8_t [__arm_]vmaxnmaq_m[_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMAT.F16 Qda,Qm	Qda -> result	MVE
float32x4_t [__arm_]vmaxnmaq_m[_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qda b -> Qm p -> Rp	VMSR P0,Rp VPST VMAXNMAT.F32 Qda,Qm	Qda -> result	MVE

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 p -> Rp	VMSR P0,Rp VPST VABDT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vabdq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vabdq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vabdq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vabdq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [__arm_]vabdq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.U32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [__arm_]vabdq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [__arm_]vabdq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VABDT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [__arm_]vabsq[_f16](float16x8_t a)	a -> Qm	VABS.F16 Qd,Qm	Qd -> result	MVE/NEON
float32x4_t [__arm_]vabsq[_f32](float32x4_t a)	a -> Qm	VABS.F32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vabsq[_s8](int8x16_t a)	a -> Qm	VABS.S8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vabsq[_s16](int16x8_t a)	a -> Qm	VABS.S16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vabsq[_s32](int32x4_t a)	a -> Qm	VABS.S32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vabsq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	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_]vadcq[_s32](int32x4_t a, int32x4_t b, unsigned * carry_out)	a -> Qn b -> Qm	VADC.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
uint32x4_t [__arm_]vadcq[_u32](uint32x4_t a, uint32x4_t b, unsigned * carry_out)	a -> Qn b -> Qm	VADC.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
int32x4_t [__arm_]vadcq_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 VADC.IT32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry_out	MVE
uint32x4_t [__arm_]vadcq_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 VADC.IT32 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_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
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_nzcvqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzcvqc,Rs VMSR P0,Rp VPST VADCT.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
uint32x4_t [__arm_]vadcq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, unsigned * carry, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm *carry -> Rt p -> Rp	VMRS Rs,FPSCR_nzcvqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzcvqc,Rs VMSR P0,Rp VPST VADCT.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
float16x8_t [__arm_]vaddq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VADD.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [__arm_]vaddq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VADD.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vaddq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VADD.F16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [__arm_]vaddq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VADD.F32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [__arm_]vaddq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VADD.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vaddq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VADD.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vaddq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VADD.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vaddq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VADD.I8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vaddq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VADD.I16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vaddq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VADD.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [__arm_]vaddq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VADD.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vaddq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VADD.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vaddq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VADD.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vaddq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VADD.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [__arm_]vaddq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VADD.I16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [__arm_]vaddq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VADD.I32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [__arm_]vaddq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.F16 Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [__arm_]vaddq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VADDT.F32 Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [__arm_]vaddq_m[_n_f16](float16x8_t inactive, float16x8_t a, float16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VADDT.F16 Qd,Qn,Rm	Qd -> result	MVE

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)	a -> Qm	VCLS.S32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vclsq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCLST.S8 Qd,Qm	Qd -> result	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)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCLST.S32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vclzq[_s8](int8x16_t a)	a -> Qm	VCLZ.I8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vclzq[_s16](int16x8_t a)	a -> Qm	VCLZ.I16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vclzq[_s32](int32x4_t a)	a -> Qm	VCLZ.I32 Qd,Qm	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vclzq[_u8](uint8x16_t a)	a -> Qm	VCLZ.I8 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vclzq[_u16](uint16x8_t a)	a -> Qm	VCLZ.I16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vclzq[_u32](uint32x4_t a)	a -> Qm	VCLZ.I32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vclzq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCLZT.I8 Qd,Qm	Qd -> result	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 p -> Rp	VMSR P0,Rp VPST VCLZT.I32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vclzq_m[_u8](uint8x16_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCLZT.I8 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vclzq_m[_u16](uint16x8_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	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)	a -> Qm	VNEG.F16 Qd,Qm	Qd -> result	MVE/NEON
float32x4_t [__arm_]vnegq[_f32](float32x4_t a)	a -> Qm	VNEG.F32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vnegq[_s8](int8x16_t a)	a -> Qm	VNEG.S8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vnegq[_s16](int16x8_t a)	a -> Qm	VNEG.S16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vnegq[_s32](int32x4_t a)	a -> Qm	VNEG.S32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vnegq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VNEG.T.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 VNEG.T.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 VNEG.T.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 VNEG.T.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 VNEG.T.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)	a -> Qn b -> Qm	VMULH.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vmulhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMULH.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vmulhq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VMULH.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vmulhq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VMULH.U16 Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [__arm_]vmulhq[_u32](uint32x4_t a, 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 p -> Rp	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)	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)	a -> Qn b -> Qm	VMULLB.P16 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vmullbq_int[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMULLB.S8 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vmullbq_int[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMULLB.S16 Qd,Qn,Qm	Qd -> result	MVE

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

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.I8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vmulq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vmulq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMULT.I32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [__arm_]vmulq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I8 Qd,Qn,Rm	Qd -> result	MVE

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.I32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vmulq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [__arm_]vmulq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VMULT.I16 Qd,Qn,Rm	Qd -> result	MVE
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.I32 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.I32 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_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
int32x4_t [__arm_]vsbcq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, unsigned * carry, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm *carry -> Rt p -> Rp	VMRS Rs,FPSCR_nzcvqc BFI Rs,Rt,#29,#1 VMSR FPSCR_nzcvqc,Rs VMSR P0,Rp VPST VSBCI.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvqc LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [__arm_]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_nzcvcq BFI Rs,Rt,#29,#1 VMSR FPSCR_nzcvcq,Rs VMSR P0,Rp VPST VSBCT.I32 Qd,Qn,Qm VMRS Rt,FPSCR_nzcvcq LSR Rt,#29 AND Rt,#1	Qd -> result Rt -> *carry	MVE
int8x16_t [__arm_]vsubq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VSUB.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vsubq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VSUB.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vsubq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VSUB.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vsubq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VSUB.I8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vsubq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VSUB.I16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vsubq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VSUB.I32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [__arm_]vsubq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VSUB.I8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vsubq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VSUB.I16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vsubq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VSUB.I32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vsubq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VSUB.I8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [__arm_]vsubq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VSUB.I16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [__arm_]vsubq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VSUB.I32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [__arm_]vsubq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VSUB.F16 Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [__arm_]vsubq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VSUB.F32 Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vsubq[_n_f16](float16x8_t a, float16_t b)	a -> Qn b -> Rm	VSUB.F16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [__arm_]vsubq[_n_f32](float32x4_t a, float32_t b)	a -> Qn b -> Rm	VSUB.F32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [__arm_]vsubq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vsubq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vsubq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VSUBT.I32 Qd,Qn,Qm	Qd -> result	MVE
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.I8 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)	a -> Qn b -> Qm	VCADD.F16 Qd,Qn,Qm,#90	Qd -> result	MVE/NEON
float32x4_t [__arm_]vcaddq_rot90[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VCADD.F32 Qd,Qn,Qm,#90	Qd -> result	MVE/NEON
int8x16_t [__arm_]vcaddq_rot90[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VCADD.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
int16x8_t [__arm_]vcaddq_rot90[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VCADD.I16 Qd,Qn,Qm,#90	Qd -> result	MVE
int32x4_t [__arm_]vcaddq_rot90[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VCADD.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
uint8x16_t [__arm_]vcaddq_rot90[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VCADD.I8 Qd,Qn,Qm,#90	Qd -> result	MVE
uint16x8_t [__arm_]vcaddq_rot90[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VCADD.I16 Qd,Qn,Qm,#90	Qd -> result	MVE
uint32x4_t [__arm_]vcaddq_rot90[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VCADD.I32 Qd,Qn,Qm,#90	Qd -> result	MVE
float16x8_t [__arm_]vcaddq_rot270[_f16](float16x8_t a, float16x8_t b)	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 p -> Rp	VMSR P0,Rp VPST VCMLAT.F32 Qda,Qn,Qm,#0	Qda -> result	MVE
float16x8_t [__arm_]vcmlaq_rot90_m[_f16](float16x8_t a, float16x8_t b, float16x8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VCMLAT.F16 Qda,Qn,Qm,#90	Qda -> result	MVE
float32x4_t [__arm_]vcmlaq_rot90_m[_f32](float32x4_t a, float32x4_t b, float32x4_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VCMLAT.F32 Qda,Qn,Qm,#90	Qda -> result	MVE
float16x8_t [__arm_]vcmlaq_rot180_m[_f16](float16x8_t a, float16x8_t b, float16x8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VCMLAT.F16 Qda,Qn,Qm,#180	Qda -> result	MVE
float32x4_t [__arm_]vcmlaq_rot180_m[_f32](float32x4_t a, float32x4_t b, float32x4_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VCMLAT.F32 Qda,Qn,Qm,#180	Qda -> result	MVE
float16x8_t [__arm_]vcmlaq_rot270_m[_f16](float16x8_t a, float16x8_t b, float16x8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VCMLAT.F16 Qda,Qn,Qm,#270	Qda -> result	MVE
float32x4_t [__arm_]vcmlaq_rot270_m[_f32](float32x4_t a, float32x4_t b, float32x4_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VCMLAT.F32 Qda,Qn,Qm,#270	Qda -> result	MVE
float16x8_t [__arm_]vcmlq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VMUL.F16 Qd,Qn,Qm,#0	Qd -> result	MVE
float32x4_t [__arm_]vcmlq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VMUL.F32 Qd,Qn,Qm,#0	Qd -> result	MVE
float16x8_t [__arm_]vcmlq_rot90[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VMUL.F16 Qd,Qn,Qm,#90	Qd -> result	MVE
float32x4_t [__arm_]vcmlq_rot90[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VMUL.F32 Qd,Qn,Qm,#90	Qd -> result	MVE
float16x8_t [__arm_]vcmlq_rot180[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VMUL.F16 Qd,Qn,Qm,#180	Qd -> result	MVE
float32x4_t [__arm_]vcmlq_rot180[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VMUL.F32 Qd,Qn,Qm,#180	Qd -> result	MVE
float16x8_t [__arm_]vcmlq_rot270[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VMUL.F16 Qd,Qn,Qm,#270	Qd -> result	MVE
float32x4_t [__arm_]vcmlq_rot270[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VMUL.F32 Qd,Qn,Qm,#270	Qd -> result	MVE
float16x8_t [__arm_]vcmlq_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,#0	Qd -> result	MVE
float32x4_t [__arm_]vcmlq_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,#0	Qd -> result	MVE
float16x8_t [__arm_]vcmlq_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 VMULT.F16 Qd,Qn,Qm,#90	Qd -> result	MVE
float32x4_t [__arm_]vcmlq_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 VMULT.F32 Qd,Qn,Qm,#90	Qd -> result	MVE
float16x8_t [__arm_]vcmlq_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 VMULT.F16 Qd,Qn,Qm,#180	Qd -> result	MVE
float32x4_t [__arm_]vcmlq_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 VMULT.F32 Qd,Qn,Qm,#180	Qd -> result	MVE
float16x8_t [__arm_]vcmlq_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 VMULT.F16 Qd,Qn,Qm,#270	Qd -> result	MVE
float32x4_t [__arm_]vcmlq_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 VMULT.F32 Qd,Qn,Qm,#270	Qd -> result	MVE
int8x16_t [__arm_]vqabsq[_s8](int8x16_t a)	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_]vqabsq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VQABST.S8 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqabsq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VQABST.S16 Qd,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqabsq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VQABST.S32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vqaddq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VQADD.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vqaddq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQADD.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vqaddq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VQADD.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [__arm_]vqaddq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VQADD.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [__arm_]vqaddq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VQADD.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [__arm_]vqaddq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VQADD.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [__arm_]vqaddq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VQADD.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vqaddq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQADD.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vqaddq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQADD.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vqaddq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VQADD.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vqaddq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VQADD.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vqaddq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VQADD.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vqaddq_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 VQADDT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vqaddq_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 VQADDT.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vqaddq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQADDT.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [__arm_]vqaddq_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 VQADDT.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [__arm_]vqaddq_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 VQADDT.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [__arm_]vqaddq_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 VQADDT.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [__arm_]vqaddq_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 VQADDT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqaddq_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 VQADDT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqaddq_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 VQADDT.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vqaddq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQADDT.U8 Qd,Qn,Qm	Qd -> result	MVE

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 VQADD.T.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 VQADD.T.U32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [__arm_]vqdmldhq[_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_]vqdmldhq[_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_]vqdmldhq[_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_]vqdmldhq_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_]vqdmldhq_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_]vqdmldhq_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_]vqdmldhxxq[_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_]vqdmldhxxq[_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_]vqdmldhxxq[_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_]vqdmldhxxq_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_]vqdmldhxxq_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_]vqdmldhxxq_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_]vqrdmladhxxq[_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_]vqrdmladhxxq[_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_]vqrdmladhqxq[_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_]vqrdmladhqxq_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_]vqrdmladhqxq_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_]vqrdmladhqxq_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_]vqrdmlahq[_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_]vqrdmlahq[_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_]vqrdmlahq[_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_]vqrdmlahq[_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_]vqrdmlahq[_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_]vqrdmlahq[_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_]vqrdmlahq_m[_n_s8](int8x16_t a, int8x16_t b, int8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLAHT.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [__arm_]vqrdmlahq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLAHT.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [__arm_]vqrdmlahq_m[_n_s32](int32x4_t a, int32x4_t b, int32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQDMLAHT.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vqrdmlahq_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_]vqrdmlahq_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_]vqrdmlahq_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, int8x16_t b, int8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLAHT.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [__arm_]vqrdmlahq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLAHT.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [__arm_]vqrdmlahq_m[_n_s32](int32x4_t a, int32x4_t b, int32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLAHT.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vqrdmlahq_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 VQRDMLAHT.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [__arm_]vqrdmlahq_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 VQRDMLAHT.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [__arm_]vqrdmlahq_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 VQRDMLAHT.U32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [__arm_]vqrdmlashq[_n_s8](int8x16_t a, int8x16_t b, int8_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLASH.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [__arm_]vqrdmlashq[_n_s16](int16x8_t a, int16x8_t b, int16_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLASH.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [__arm_]vqrdmlashq[_n_s32](int32x4_t a, int32x4_t b, int32_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLASH.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vqrdmlashq[_n_u8](uint8x16_t a, uint8x16_t b, uint8_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLASH.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [__arm_]vqrdmlashq[_n_u16](uint16x8_t a, uint16x8_t b, uint16_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLASH.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [__arm_]vqrdmlashq[_n_u32](uint32x4_t a, uint32x4_t b, uint32_t c)	a -> Qda b -> Qn c -> Rm	VQRDMLASH.U32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [__arm_]vqrdmlashq_m[_n_s8](int8x16_t a, int8x16_t b, int8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLASHT.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [__arm_]vqrdmlashq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLASHT.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [__arm_]vqrdmlashq_m[_n_s32](int32x4_t a, int32x4_t b, int32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMLASHT.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vqrdmlashq_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 VQRDMLASHT.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [__arm_]vqrdmlashq_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 VQRDMLASHT.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [__arm_]vqrdmlashq_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 VQRDMLASHT.U32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [__arm_]vqdmldshq[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b)	inactive -> Qd a -> Qn b -> Qm	VQDMLSDH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqdmldshq[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b)	inactive -> Qd a -> Qn b -> Qm	VQDMLSDH.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqdmldshq[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b)	inactive -> Qd a -> Qn b -> Qm	VQDMLSDH.S32 Qd,Qn,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [__arm_]vqdmldshq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqdmldshq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMLSDHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqdmldshq_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_]vqdmldshxq[_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_]vqdmldshxq[_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_]vqdmldshxq[_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_]vqdmldshxq_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_]vqdmldshxq_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_]vqdmldshxq_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_]vqrdmldshq[_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_]vqrdmldshq[_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_]vqrdmldshq[_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_]vqrdmldshq_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_]vqrdmldshq_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_]vqrdmldshq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMLSDHT.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [__arm_]vqrdmldshxq[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b)	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDHX.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqrdmldshxq[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b)	inactive -> Qd a -> Qn b -> Qm	VQRDMLSDHX.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqrdmldshxq[_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_]vqrdmldshxq_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_]vqrdmldshxq_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_]vqrdmldshxq_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 b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULHT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vqdmulhq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULHT.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vqdmulhq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULHT.S32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [__arm_]vqdmulhq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VQDMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqdmulhq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQDMULH.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vqdmulhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQDMULH.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vqdmulhq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMULHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqdmulhq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMULHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqdmulhq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMULHT.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [__arm_]vqrdmulhq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VQRDMULH.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vqrdmulhq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQRDMULH.S16 Qd,Qn,Rm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vqrdmulhq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VQRDMULH.S32 Qd,Qn,Rm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vqrdmulhq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMULHT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vqrdmulhq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMULHT.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vqrdmulhq_m[_n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQRDMULHT.S32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [__arm_]vqrdmulhq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VQRDMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqrdmulhq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQRDMULH.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vqrdmulhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQRDMULH.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vqrdmulhq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMULHT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqrdmulhq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMULHT.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqrdmulhq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQRDMULHT.S32 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqdmullbq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQDMULLB.S16 Qd,Qn,Rm	Qd -> result	MVE
int64x2_t [__arm_]vqdmullbq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VQDMULLB.S32 Qd,Qn,Rm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32x4_t [__arm_]vqdmullbq_m[_n_s16](int32x4_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULLBT.S16 Qd,Qn,Rm	Qd -> result	MVE
int64x2_t [__arm_]vqdmullbq_m[_n_s32](int64x2_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULLBT.S32 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vqdmullbq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQDMULLB.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [__arm_]vqdmullbq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQDMULLB.S32 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqdmullbq_m[_s16](int32x4_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMULLBT.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [__arm_]vqdmullbq_m[_s32](int64x2_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMULLBT.S32 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqdmulltq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQDMULLT.S16 Qd,Qn,Rm	Qd -> result	MVE
int64x2_t [__arm_]vqdmulltq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VQDMULLT.S32 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vqdmulltq_m[_n_s16](int32x4_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULLTT.S16 Qd,Qn,Rm	Qd -> result	MVE
int64x2_t [__arm_]vqdmulltq_m[_n_s32](int64x2_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQDMULLTT.S32 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vqdmulltq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQDMULLT.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [__arm_]vqdmulltq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQDMULLT.S32 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqdmulltq_m[_s16](int32x4_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMULLTT.S16 Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [__arm_]vqdmulltq_m[_s32](int64x2_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQDMULLTT.S32 Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [__arm_]vqnegq[_s8](int8x16_t a)	a -> Qm	VQNEG.S8 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vqnegq[_s16](int16x8_t a)	a -> Qm	VQNEG.S16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vqnegq[_s32](int32x4_t a)	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 p -> Rp	VMSR P0,Rp VPST VQNEGT.S16 Qd,Qm	Qd -> result	MVE
int32x4_t [__arm_]vqnegq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VQNEGT.S32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vqsubq[_n_s8](int8x16_t a, int8_t b)	a -> Qn b -> Rm	VQSUB.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vqsubq[_n_s16](int16x8_t a, int16_t b)	a -> Qn b -> Rm	VQSUB.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vqsubq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VQSUB.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [__arm_]vqsubq[_n_u8](uint8x16_t a, uint8_t b)	a -> Qn b -> Rm	VQSUB.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [__arm_]vqsubq[_n_u16](uint16x8_t a, uint16_t b)	a -> Qn b -> Rm	VQSUB.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [__arm_]vqsubq[_n_u32](uint32x4_t a, uint32_t b)	a -> Qn b -> Rm	VQSUB.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [__arm_]vqsubq_m[_n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQSUBT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vqsubq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQSUBT.S16 Qd,Qn,Rm	Qd -> result	MVE

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 p -> Rp	VMSR P0,Rp VPST VQSUBT.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [__arm_]vqsubq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQSUBT.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [__arm_]vqsubq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQSUBT.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [__arm_]vqsubq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VQSUBT.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [__arm_]vqsubq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VQSUB.S8 Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vqsubq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VQSUB.S16 Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vqsubq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VQSUB.S32 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vqsubq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VQSUB.U8 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vqsubq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VQSUB.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vqsubq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VQSUB.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vqsubq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VQSUBT.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqsubq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm 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] VLD43.16 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
int32x4x4_t [__arm_]vld4q[_s32](int32_t const * addr)	addr -> Rn	VLD40.32 {Qd - Qd4},[Rn] VLD41.32 {Qd - Qd4},[Rn] VLD42.32 {Qd - Qd4},[Rn] VLD43.32 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
uint8x16x4_t [__arm_]vld4q[_u8](uint8_t const * addr)	addr -> Rn	VLD40.8 {Qd - Qd4},[Rn] VLD41.8 {Qd - Qd4},[Rn] VLD42.8 {Qd - Qd4},[Rn] VLD43.8 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
uint16x8x4_t [__arm_]vld4q[_u16](uint16_t const * addr)	addr -> Rn	VLD40.16 {Qd - Qd4},[Rn] VLD41.16 {Qd - Qd4},[Rn] VLD42.16 {Qd - Qd4},[Rn] VLD43.16 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
uint32x4x4_t [__arm_]vld4q[_u32](uint32_t const * addr)	addr -> Rn	VLD40.32 {Qd - Qd4},[Rn] VLD41.32 {Qd - Qd4},[Rn] VLD42.32 {Qd - Qd4},[Rn] VLD43.32 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
float16x8x4_t [__arm_]vld4q[_f16](float16_t const * addr)	addr -> Rn	VLD40.16 {Qd - Qd4},[Rn] VLD41.16 {Qd - Qd4},[Rn] VLD42.16 {Qd - Qd4},[Rn] VLD43.16 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
float32x4x4_t [__arm_]vld4q[_f32](float32_t const * addr)	addr -> Rn	VLD40.32 {Qd - Qd4},[Rn] VLD41.32 {Qd - Qd4},[Rn] VLD42.32 {Qd - Qd4},[Rn] VLD43.32 {Qd - Qd4},[Rn]	Qd -> result.val[0] Qd2 -> result.val[1] Qd3 -> result.val[2] Qd4 -> result.val[3]	MVE
int8x16_t [__arm_]vldrbq_s8(int8_t const * base)	base -> Rn	VLDRB.8 Qd,[Rn]	Qd -> result	MVE
int16x8_t [__arm_]vldrbq_s16(int16_t const * base)	base -> Rn	VLDRB.S16 Qd,[Rn]	Qd -> result	MVE
int32x4_t [__arm_]vldrbq_s32(int32_t const * base)	base -> Rn	VLDRB.S32 Qd,[Rn]	Qd -> result	MVE
uint8x16_t [__arm_]vldrbq_u8(uint8_t const * base)	base -> Rn	VLDRB.8 Qd,[Rn]	Qd -> result	MVE
uint16x8_t [__arm_]vldrbq_u16(uint16_t const * base)	base -> Rn	VLDRB.U16 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 VLDRBT.8 Qd,[Rn]	Qd -> result	MVE
int16x8_t [__arm_]vldrbq_z_s16(int8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.S16 Qd,[Rn]	Qd -> result	MVE
int32x4_t [__arm_]vldrbq_z_s32(int8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.S32 Qd,[Rn]	Qd -> result	MVE
uint8x16_t [__arm_]vldrbq_z_u8(uint8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.8 Qd,[Rn]	Qd -> result	MVE
uint16x8_t [__arm_]vldrbq_z_u16(uint8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.U16 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [__arm_]vldrbq_z_u32(uint8_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRBT.U32 Qd,[Rn]	Qd -> result	MVE
int16x8_t [__arm_]vldrhq_s16(int16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE
int32x4_t [__arm_]vldrhq_s32(int16_t const * base)	base -> Rn	VLDRH.S32 Qd,[Rn]	Qd -> result	MVE
uint16x8_t [__arm_]vldrhq_u16(uint16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [__arm_]vldrhq_u32(uint16_t const * base)	base -> Rn	VLDRH.U32 Qd,[Rn]	Qd -> result	MVE
float16x8_t [__arm_]vldrhq_f16(float16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE
int16x8_t [__arm_]vldrhq_z_s16(int16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRHT.S16 Qd,[Rn]	Qd -> result	MVE
int32x4_t [__arm_]vldrhq_z_s32(int16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRHT.S32 Qd,[Rn]	Qd -> result	MVE
uint16x8_t [__arm_]vldrhq_z_u16(uint16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRHT.U16 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [__arm_]vldrhq_z_u32(uint16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRHT.U32 Qd,[Rn]	Qd -> result	MVE
float16x8_t [__arm_]vldrhq_z_f16(float16_t const * base, mve_pred16_t p)	base -> Rn p -> Rp	VMSR P0,Rp VPST VLDRHT.F16 Qd,[Rn]	Qd -> result	MVE
int32x4_t [__arm_]vldrwq_s32(int32_t const * base)	base -> Rn	VLDRAW.32 Qd,[Rn]	Qd -> result	MVE
uint32x4_t [__arm_]vldrwq_u32(uint32_t const * base)	base -> Rn	VLDRAW.32 Qd,[Rn]	Qd -> result	MVE
float32x4_t [__arm_]vldrwq_f32(float32_t const * base)	base -> Rn	VLDRAW.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 VLDRAWT.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 VLDRAWT.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 VLDRAWT.32 Qd,[Rn]	Qd -> result	MVE
int8x16_t [__arm_]vld1q_s8(int8_t const * base)	base -> Rn	VLDRB.8 Qd,[Rn]	Qd -> result	MVE/NEON
int16x8_t [__arm_]vld1q_s16(int16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE/NEON
int32x4_t [__arm_]vld1q_s32(int32_t const * base)	base -> Rn	VLDRAW.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)	base -> Rn	VLDRAW.32 Qd,[Rn]	Qd -> result	MVE/NEON
float16x8_t [__arm_]vld1q_f16(float16_t const * base)	base -> Rn	VLDRH.16 Qd,[Rn]	Qd -> result	MVE/NEON
float32x4_t [__arm_]vld1q_f32(float32_t const * base)	base -> Rn	VLDRAW.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)	base -> Rn offset -> Qm	VLDRH.U16 Qd,[Rn,Qm]	Qd -> result	MVE
uint32x4_t [__arm_]vldrhq_gather_offset_u32(uint16_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRH.U32 Qd,[Rn,Qm]	Qd -> result	MVE
float16x8_t [__arm_]vldrhq_gather_offset_f16(float16_t const * base, uint16x8_t offset)	base -> Rn offset -> Qm	VLDRH.F16 Qd,[Rn,Qm]	Qd -> result	MVE
int16x8_t [__arm_]vldrhq_gather_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]	Qd -> result	MVE
int32x4_t [__arm_]vldrhq_gather_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]	Qd -> result	MVE

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 VLD RHT.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 VLD RHT.U32 Qd,[Rn,Qm]	Qd -> result	MVE
float16x8_t [__arm_]vldrhq_gather_offset_z[f16](float16_t const * base, uint16x8_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLD RHT.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	VLD RH.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	VLD RH.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	VLD RH.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	VLD RH.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	VLD RH.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 VLD RHT.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 VLD RHT.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 VLD RHT.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 VLD RHT.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 VLD RHT.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	VLD RB.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	VLD RB.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	VLD RB.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	VLD RB.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	VLD RB.U16 Qd,[Rn,Qm]	Qd -> result	MVE
uint32x4_t [__arm_]vldrbq_gather_offset[u32](uint8_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLD RB.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 VLD RBT.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 VLD RBT.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 VLD RBT.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 VLD RBT.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 VLD RBT.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 VLD RBT.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	VLD RW.U32 Qd,[Rn,Qm]	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32x4_t [<u>arm_</u>]vldrwq_gather_offset[_u32](uint32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRAW.U32 Qd,[Rn,Qm]	Qd -> result	MVE
float32x4_t [<u>arm_</u>]vldrwq_gather_offset[_f32](float32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRAW.U32 Qd,[Rn,Qm]	Qd -> result	MVE
int32x4_t [<u>arm_</u>]vldrwq_gather_offset_z[_s32](int32_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRAWT.U32 Qd,[Rn,Qm]	Qd -> result	MVE
uint32x4_t [<u>arm_</u>]vldrwq_gather_offset_z[_u32](uint32_t const * base, uint32x4_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRAWT.U32 Qd,[Rn,Qm]	Qd -> result	MVE
float32x4_t [<u>arm_</u>]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 VLDRAWT.U32 Qd,[Rn,Qm]	Qd -> result	MVE
int32x4_t [<u>arm_</u>]vldrwq_gather_shifted_offset[_s32](int32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRAW.U32 Qd,[Rn,Qm, UXTW #2]	Qd -> result	MVE
uint32x4_t [<u>arm_</u>]vldrwq_gather_shifted_offset[_u32](uint32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRAW.U32 Qd,[Rn,Qm, UXTW #2]	Qd -> result	MVE
float32x4_t [<u>arm_</u>]vldrwq_gather_shifted_offset[_f32](float32_t const * base, uint32x4_t offset)	base -> Rn offset -> Qm	VLDRAW.U32 Qd,[Rn,Qm, UXTW #2]	Qd -> result	MVE
int32x4_t [<u>arm_</u>]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 VLDRAWT.U32 Qd,[Rn,Qm, UXTW #2]	Qd -> result	MVE
uint32x4_t [<u>arm_</u>]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 VLDRAWT.U32 Qd,[Rn,Qm, UXTW #2]	Qd -> result	MVE
float32x4_t [<u>arm_</u>]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 VLDRAWT.U32 Qd,[Rn,Qm, UXTW #2]	Qd -> result	MVE
int32x4_t [<u>arm_</u>]vldrwq_gather_base_s32(uint32x4_t addr, const int offset)	addr -> Qn offset in +/- 4*[0..127]	VLDRAW.U32 Qd,[Qn,#offset]	Qd -> result	MVE
uint32x4_t [<u>arm_</u>]vldrwq_gather_base_u32(uint32x4_t addr, const int offset)	addr -> Qn offset in +/- 4*[0..127]	VLDRAW.U32 Qd,[Qn,#offset]	Qd -> result	MVE
float32x4_t [<u>arm_</u>]vldrwq_gather_base_f32(uint32x4_t addr, const int offset)	addr -> Qn offset in +/- 4*[0..127]	VLDRAW.U32 Qd,[Qn,#offset]	Qd -> result	MVE
int32x4_t [<u>arm_</u>]vldrwq_gather_base_z_s32(uint32x4_t addr, const int offset, mve_pred16_t p)	addr -> Qn offset in +/- 4*[0..127] p -> Rp	VMSR P0,Rp VPST VLDRAWT.U32 Qd,[Qn,#offset]	Qd -> result	MVE
uint32x4_t [<u>arm_</u>]vldrwq_gather_base_z_u32(uint32x4_t addr, const int offset, mve_pred16_t p)	addr -> Qn offset in +/- 4*[0..127] p -> Rp	VMSR P0,Rp VPST VLDRAWT.U32 Qd,[Qn,#offset]	Qd -> result	MVE
float32x4_t [<u>arm_</u>]vldrwq_gather_base_z_f32(uint32x4_t addr, const int offset, mve_pred16_t p)	addr -> Qn offset in +/- 4*[0..127] p -> Rp	VMSR P0,Rp VPST VLDRAWT.U32 Qd,[Qn,#offset]	Qd -> result	MVE
int32x4_t [<u>arm_</u>]vldrwq_gather_base_wb_s32(uint32x4_t * addr, const int offset)	*addr -> Qn offset in +/- 4*[0..127]	VLDRAW.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
uint32x4_t [<u>arm_</u>]vldrwq_gather_base_wb_u32(uint32x4_t * addr, const int offset)	*addr -> Qn offset in +/- 4*[0..127]	VLDRAW.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
float32x4_t [<u>arm_</u>]vldrwq_gather_base_wb_f32(uint32x4_t * addr, const int offset)	*addr -> Qn offset in +/- 4*[0..127]	VLDRAW.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
int32x4_t [<u>arm_</u>]vldrwq_gather_base_wb_z_s32(uint32x4_t * addr, const int offset, mve_pred16_t p)	*addr -> Qn offset in +/- 4*[0..127] p -> Rp	VMSR P0,Rp VPST VLDRAWT.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
uint32x4_t [<u>arm_</u>]vldrwq_gather_base_wb_z_u32(uint32x4_t * addr, const int offset, mve_pred16_t p)	*addr -> Qn offset in +/- 4*[0..127] p -> Rp	VMSR P0,Rp VPST VLDRAWT.U32 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
float32x4_t [<u>arm_</u>]vldrwq_gather_base_wb_z_f32(uint32x4_t * addr, const int offset, mve_pred16_t p)	*addr -> Qn offset in +/- 4*[0..127] p -> Rp	VMSR P0,Rp VPST VLDRAWT.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 const * base, uint64x2_t offset)	base -> Rn offset -> Qm	VLDRD.U64 Qd,[Rn,Qm]	Qd -> result	MVE
uint64x2_t [__arm_]vldrdq_gather_offset[_u64](uint64_t const * base, uint64x2_t offset)	base -> Rn offset -> Qm	VLDRD.U64 Qd,[Rn,Qm]	Qd -> result	MVE
int64x2_t [__arm_]vldrdq_gather_offset_z[_s64](int64_t const * base, uint64x2_t offset, mve_pred16_t p)	base -> Rn offset -> Qm p -> Rp	VMSR P0,Rp VPST VLDRDT.U64 Qd,[Rn,Qm]	Qd -> result	MVE
uint64x2_t [__arm_]vldrdq_gather_offset_z[_u64](uint64_t const * base, uint64x2_t offset, mve_pred16_t p)	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,UCTW #3]	Qd -> result	MVE
uint64x2_t [__arm_]vldrdq_gather_shifted_offset[_u64](uint64_t const * base, uint64x2_t offset)	base -> Rn offset -> Qm	VLDRD.U64 Qd,[Rn,Qm,UCTW #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,UCTW #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,UCTW #3]	Qd -> result	MVE
int64x2_t [__arm_]vldrdq_gather_base_s64(uint64x2_t addr, const int offset)	addr -> Qn offset in +/- 8*[0..127]	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*[0..127]	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*[0..127] 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*[0..127] 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*[0..127]	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*[0..127]	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*[0..127] 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*[0..127] p -> Rp	VMSR P0,Rp VPST VLDRDT.U64 Qd,[Qn,#offset]!	Qd -> result Qn -> *addr	MVE
void [__arm_]vst2q[_s8](int8_t * addr, int8x16x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2	VST20.8 {Qd - Qd2},[Rn] VST21.8 {Qd - Qd2},[Rn]	void -> result	MVE
void [__arm_]vst2q[_s16](int16_t * addr, int16x8x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2	VST20.16 {Qd - Qd2},[Rn] VST21.16 {Qd - Qd2},[Rn]	void -> result	MVE
void [__arm_]vst2q[_s32](int32_t * addr, int32x4x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2	VST20.32 {Qd - Qd2},[Rn] VST21.32 {Qd - Qd2},[Rn]	void -> result	MVE
void [__arm_]vst2q[_u8](uint8_t * addr, uint8x16x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2	VST20.8 {Qd - Qd2},[Rn] VST21.8 {Qd - Qd2},[Rn]	void -> result	MVE
void [__arm_]vst2q[_u16](uint16_t * addr, uint16x8x2_t value)	addr -> Rn value.val[0] -> Qd value.val[1] -> Qd2	VST20.16 {Qd - Qd2},[Rn] VST21.16 {Qd - Qd2},[Rn]	void -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void [__arm_]vst2q[_u32](uint32_t * addr, uint32x4x2_t value)</code>	<code>addr -> Rn</code> <code>value.val[0] -> Qd</code> <code>value.val[1] -> Qd2</code>	VST20.32 {Qd - Qd2},[Rn] VST21.32 {Qd - Qd2},[Rn]	void -> result	MVE
<code>void [__arm_]vst2q[_f16](float16_t * addr, float16x8x2_t value)</code>	<code>addr -> Rn</code> <code>value.val[0] -> Qd</code> <code>value.val[1] -> Qd2</code>	VST20.16 {Qd - Qd2},[Rn] VST21.16 {Qd - Qd2},[Rn]	void -> result	MVE
<code>void [__arm_]vst2q[_f32](float32_t * addr, float32x4x2_t value)</code>	<code>addr -> Rn</code> <code>value.val[0] -> Qd</code> <code>value.val[1] -> Qd2</code>	VST20.32 {Qd - Qd2},[Rn] VST21.32 {Qd - Qd2},[Rn]	void -> result	MVE
<code>void [__arm_]vst4q[_s8](int8_t * addr, int8x16x4_t value)</code>	<code>addr -> Rn</code> <code>value.val[0] -> Qd</code> <code>value.val[1] -> Qd2</code> <code>value.val[2] -> Qd3</code> <code>value.val[3] -> Qd4</code>	VST40.8 {Qd - Qd4},[Rn] VST41.8 {Qd - Qd4},[Rn] VST42.8 {Qd - Qd4},[Rn] VST43.8 {Qd - Qd4},[Rn]	void -> result	MVE
<code>void [__arm_]vst4q[_s16](int16_t * addr, int16x8x4_t value)</code>	<code>addr -> Rn</code> <code>value.val[0] -> Qd</code> <code>value.val[1] -> Qd2</code> <code>value.val[2] -> Qd3</code> <code>value.val[3] -> Qd4</code>	VST40.16 {Qd - Qd4},[Rn] VST41.16 {Qd - Qd4},[Rn] VST42.16 {Qd - Qd4},[Rn] VST43.16 {Qd - Qd4},[Rn]	void -> result	MVE
<code>void [__arm_]vst4q[_s32](int32_t * addr, int32x4x4_t value)</code>	<code>addr -> Rn</code> <code>value.val[0] -> Qd</code> <code>value.val[1] -> Qd2</code> <code>value.val[2] -> Qd3</code> <code>value.val[3] -> Qd4</code>	VST40.32 {Qd - Qd4},[Rn] VST41.32 {Qd - Qd4},[Rn] VST42.32 {Qd - Qd4},[Rn] VST43.32 {Qd - Qd4},[Rn]	void -> result	MVE
<code>void [__arm_]vst4q[_u8](uint8_t * addr, uint8x16x4_t value)</code>	<code>addr -> Rn</code> <code>value.val[0] -> Qd</code> <code>value.val[1] -> Qd2</code> <code>value.val[2] -> Qd3</code> <code>value.val[3] -> Qd4</code>	VST40.8 {Qd - Qd4},[Rn] VST41.8 {Qd - Qd4},[Rn] VST42.8 {Qd - Qd4},[Rn] VST43.8 {Qd - Qd4},[Rn]	void -> result	MVE
<code>void [__arm_]vst4q[_u16](uint16_t * addr, uint16x8x4_t value)</code>	<code>addr -> Rn</code> <code>value.val[0] -> Qd</code> <code>value.val[1] -> Qd2</code> <code>value.val[2] -> Qd3</code> <code>value.val[3] -> Qd4</code>	VST40.16 {Qd - Qd4},[Rn] VST41.16 {Qd - Qd4},[Rn] VST42.16 {Qd - Qd4},[Rn] VST43.16 {Qd - Qd4},[Rn]	void -> result	MVE
<code>void [__arm_]vst4q[_u32](uint32_t * addr, uint32x4x4_t value)</code>	<code>addr -> Rn</code> <code>value.val[0] -> Qd</code> <code>value.val[1] -> Qd2</code> <code>value.val[2] -> Qd3</code> <code>value.val[3] -> Qd4</code>	VST40.32 {Qd - Qd4},[Rn] VST41.32 {Qd - Qd4},[Rn] VST42.32 {Qd - Qd4},[Rn] VST43.32 {Qd - Qd4},[Rn]	void -> result	MVE
<code>void [__arm_]vst4q[_f16](float16_t * addr, float16x8x4_t value)</code>	<code>addr -> Rn</code> <code>value.val[0] -> Qd</code> <code>value.val[1] -> Qd2</code> <code>value.val[2] -> Qd3</code> <code>value.val[3] -> Qd4</code>	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] -> Qd4	VST40.32 {Qd - Qd4},[Rn] VST41.32 {Qd - Qd4},[Rn] VST42.32 {Qd - Qd4},[Rn] VST43.32 {Qd - Qd4},[Rn]	void -> result	MVE
void [__arm_]vstrbq[_s8](int8_t * base, int8x16_t value)	base -> Rn value -> Qd	VSTRB.8 Qd,[Rn]	void -> result	MVE
void [__arm_]vstrbq[_s16](int8_t * base, int16x8_t value)	base -> Rn value -> Qd	VSTRB.16 Qd,[Rn]	void -> result	MVE
void [__arm_]vstrbq[_s32](int8_t * base, int32x4_t value)	base -> Rn value -> Qd	VSTRB.32 Qd,[Rn]	void -> result	MVE
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
void [__arm_]vstrhq_p[_f16](float16_t * base, float16x8_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn]	void -> result	MVE
void [__arm_]vstrwq[_s32](int32_t * base, int32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE
void [__arm_]vstrwq[_u32](uint32_t * base, uint32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE
void [__arm_]vstrwq[_f32](float32_t * base, float32x4_t value)	base -> Rn value -> Qd	VSTRW.32 Qd,[Rn]	void -> result	MVE
void [__arm_]vstrwq_p[_s32](int32_t * base, int32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn]	void -> result	MVE
void [__arm_]vstrwq_p[_u32](uint32_t * base, uint32x4_t value, mve_pred16_t p)	base -> Rn value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn]	void -> result	MVE

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void [__arm_]vstrhq_scatter_offset_p[_s16](int16_t * base, uint16x8_t offset, int16x8_t value, mve_pred16_t p)</code>	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_offset_p[_s32](int16_t * base, uint32x4_t offset, int32x4_t value, mve_pred16_t p)</code>	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_offset_p[_u16](uint16_t * base, uint16x8_t offset, uint16x8_t value, mve_pred16_t p)</code>	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_offset_p[_u32](uint16_t * base, uint32x4_t offset, uint32x4_t value, mve_pred16_t p)</code>	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_offset_p[_f16](float16_t * base, uint16x8_t offset, float16x8_t value, mve_pred16_t p)</code>	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_shifted_offset[_s16](int16_t * base, uint16x8_t offset, int16x8_t value)</code>	base -> Rn offset -> Qm value -> Qd	VSTRH.16 Qd,[Rn,Qm,UCTW #1]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_shifted_offset[_s32](int16_t * base, uint32x4_t offset, int32x4_t value)</code>	base -> Rn offset -> Qm value -> Qd	VSTRH.32 Qd,[Rn,Qm,UCTW #1]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_shifted_offset[_u16](uint16_t * base, uint16x8_t offset, uint16x8_t value)</code>	base -> Rn offset -> Qm value -> Qd	VSTRH.16 Qd,[Rn,Qm,UCTW #1]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_shifted_offset[_u32](uint16_t * base, uint32x4_t offset, uint32x4_t value)</code>	base -> Rn offset -> Qm value -> Qd	VSTRH.32 Qd,[Rn,Qm,UCTW #1]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_shifted_offset[_f16](float16_t * base, uint16x8_t offset, float16x8_t value)</code>	base -> Rn offset -> Qm value -> Qd	VSTRH.16 Qd,[Rn,Qm,UCTW #1]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_shifted_offset_p[_s16](int16_t * base, uint16x8_t offset, int16x8_t value, mve_pred16_t p)</code>	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm,UCTW #1]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_shifted_offset_p[_s32](int16_t * base, uint32x4_t offset, int32x4_t value, mve_pred16_t p)</code>	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn,Qm,UCTW #1]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_shifted_offset_p[_u16](uint16_t * base, uint16x8_t offset, uint16x8_t value, mve_pred16_t p)</code>	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm,UCTW #1]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_shifted_offset_p[_u32](uint16_t * base, uint32x4_t offset, uint32x4_t value, mve_pred16_t p)</code>	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.32 Qd,[Rn,Qm,UCTW #1]	void -> result	MVE
<code>void [__arm_]vstrhq_scatter_shifted_offset_p[_f16](float16_t * base, uint16x8_t offset, float16x8_t value, mve_pred16_t p)</code>	base -> Rn offset -> Qm value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRHT.16 Qd,[Rn,Qm,UCTW #1]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_base[_s32](uint32x4_t addr, const int offset, int32x4_t value)</code>	addr -> Qn offset in +/- 4*[0..127] value -> Qd	VSTRW.U32 Qd,[Qn,#offset]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_base[_u32](uint32x4_t addr, const int offset, uint32x4_t value)</code>	addr -> Qn offset in +/- 4*[0..127] value -> Qd	VSTRW.U32 Qd,[Qn,#offset]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_base[_f32](uint32x4_t addr, const int offset, float32x4_t value)</code>	addr -> Qn offset in +/- 4*[0..127] value -> Qd	VSTRW.U32 Qd,[Qn,#offset]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_base_p[_s32](uint32x4_t addr, const int offset, int32x4_t value, mve_pred16_t p)</code>	addr -> Qn offset in +/- 4*[0..127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_base_p[_u32](uint32x4_t addr, const int offset, uint32x4_t value, mve_pred16_t p)</code>	addr -> Qn offset in +/- 4*[0..127] value -> Qd p -> Rp	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]	void -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void [__arm_]vstrwq_scatter_base_p[_f32](uint32x4_t addr, const int offset, float32x4_t value, mve_pred16_t p)</code>	<code>addr -> Qn offset in +/- 4*[0..127] value -> Qd p -> Rp</code>	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_base_wb[_s32](uint32x4_t * addr, const int offset, int32x4_t value)</code>	<code>*addr -> Qn offset in +/- 4*[0..127] value -> Qd</code>	VSTRW.U32 Qd,[Qn,#offset]!	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_base_wb[_u32](uint32x4_t * addr, const int offset, uint32x4_t value)</code>	<code>*addr -> Qn offset in +/- 4*[0..127] value -> Qd</code>	VSTRW.U32 Qd,[Qn,#offset]!	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_base_wb[_f32](uint32x4_t * addr, const int offset, float32x4_t value)</code>	<code>*addr -> Qn offset in +/- 4*[0..127] value -> Qd</code>	VSTRW.U32 Qd,[Qn,#offset]!	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_base_wb_p[_s32](uint32x4_t * addr, const int offset, int32x4_t value, mve_pred16_t p)</code>	<code>*addr -> Qn offset in +/- 4*[0..127] value -> Qd p -> Rp</code>	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]!	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_base_wb_p[_u32](uint32x4_t * addr, const int offset, uint32x4_t value, mve_pred16_t p)</code>	<code>*addr -> Qn offset in +/- 4*[0..127] value -> Qd p -> Rp</code>	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]!	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_base_wb_p[_f32](uint32x4_t * addr, const int offset, float32x4_t value, mve_pred16_t p)</code>	<code>*addr -> Qn offset in +/- 4*[0..127] value -> Qd p -> Rp</code>	VMSR P0,Rp VPST VSTRWT.U32 Qd,[Qn,#offset]!	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_offset[_s32](int32_t * base, uint32x4_t offset, int32x4_t value)</code>	<code>base -> Rn offset -> Qm value -> Qd</code>	VSTRW.32 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_offset[_u32](uint32_t * base, uint32x4_t offset, uint32x4_t value)</code>	<code>base -> Rn offset -> Qm value -> Qd</code>	VSTRW.32 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_offset[_f32](float32_t * base, uint32x4_t offset, float32x4_t value)</code>	<code>base -> Rn offset -> Qm value -> Qd</code>	VSTRW.32 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_offset_p[_s32](int32_t * base, uint32x4_t offset, int32x4_t value, mve_pred16_t p)</code>	<code>base -> Rn offset -> Qm value -> Qd p -> Rp</code>	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_offset_p[_u32](uint32_t * base, uint32x4_t offset, uint32x4_t value, mve_pred16_t p)</code>	<code>base -> Rn offset -> Qm value -> Qd p -> Rp</code>	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_offset_p[_f32](float32_t * base, uint32x4_t offset, float32x4_t value, mve_pred16_t p)</code>	<code>base -> Rn offset -> Qm value -> Qd p -> Rp</code>	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_shifted_offset[_s32](int32_t * base, uint32x4_t offset, int32x4_t value)</code>	<code>base -> Rn offset -> Qm value -> Qd</code>	VSTRW.32 Qd,[Rn,Qm,UCTW #2]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_shifted_offset[_u32](uint32_t * base, uint32x4_t offset, uint32x4_t value)</code>	<code>base -> Rn offset -> Qm value -> Qd</code>	VSTRW.32 Qd,[Rn,Qm,UCTW #2]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_shifted_offset[_f32](float32_t * base, uint32x4_t offset, float32x4_t value)</code>	<code>base -> Rn offset -> Qm value -> Qd</code>	VSTRW.32 Qd,[Rn,Qm,UCTW #2]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_shifted_offset_p[_s32](int32_t * base, uint32x4_t offset, int32x4_t value, mve_pred16_t p)</code>	<code>base -> Rn offset -> Qm value -> Qd p -> Rp</code>	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm,UCTW #2]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_shifted_offset_p[_u32](uint32_t * base, uint32x4_t offset, uint32x4_t value, mve_pred16_t p)</code>	<code>base -> Rn offset -> Qm value -> Qd p -> Rp</code>	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm,UCTW #2]	void -> result	MVE
<code>void [__arm_]vstrwq_scatter_shifted_offset_p[_f32](float32_t * base, uint32x4_t offset, float32x4_t value, mve_pred16_t p)</code>	<code>base -> Rn offset -> Qm value -> Qd p -> Rp</code>	VMSR P0,Rp VPST VSTRWT.32 Qd,[Rn,Qm,UCTW #2]	void -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
<code>void [__arm_]vstrdq_scatter_base[_s64](uint64x2_t addr, const int offset, int64x2_t value)</code>	<code>addr -> Qn</code> <code>offset in +/- 8*[0..127]</code> <code>value -> Qd</code>	VSTRD.U64 Qd,[Qn,#offset]	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_base[_u64](uint64x2_t addr, const int offset, uint64x2_t value)</code>	<code>addr -> Qn</code> <code>offset in +/- 8*[0..127]</code> <code>value -> Qd</code>	VSTRD.U64 Qd,[Qn,#offset]	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_base_p[_s64](uint64x2_t addr, const int offset, int64x2_t value, mve_pred16_t p)</code>	<code>addr -> Qn</code> <code>offset in +/- 8*[0..127]</code> <code>value -> Qd</code> <code>p -> Rp</code>	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_base_p[_u64](uint64x2_t addr, const int offset, uint64x2_t value, mve_pred16_t p)</code>	<code>addr -> Qn</code> <code>offset in +/- 8*[0..127]</code> <code>value -> Qd</code> <code>p -> Rp</code>	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_base_wb[_s64](uint64x2_t * addr, const int offset, int64x2_t value)</code>	<code>*addr -> Qn</code> <code>offset in +/- 8*[0..127]</code> <code>value -> Qd</code>	VSTRD.U64 Qd,[Qn,#offset]!	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_base_wb[_u64](uint64x2_t * addr, const int offset, uint64x2_t value)</code>	<code>*addr -> Qn</code> <code>offset in +/- 8*[0..127]</code> <code>value -> Qd</code>	VSTRD.U64 Qd,[Qn,#offset]!	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_base_wb_p[_s64](uint64x2_t * addr, const int offset, int64x2_t value, mve_pred16_t p)</code>	<code>*addr -> Qn</code> <code>offset in +/- 8*[0..127]</code> <code>value -> Qd</code> <code>p -> Rp</code>	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]!	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_base_wb_p[_u64](uint64x2_t * addr, const int offset, uint64x2_t value, mve_pred16_t p)</code>	<code>*addr -> Qn</code> <code>offset in +/- 8*[0..127]</code> <code>value -> Qd</code> <code>p -> Rp</code>	VMSR P0,Rp VPST VSTRDT.U64 Qd,[Qn,#offset]!	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_offset[_s64](int64_t * base, uint64x2_t offset, int64x2_t value)</code>	<code>base -> Rn</code> <code>offset -> Qm</code> <code>value -> Qd</code>	VSTRD.64 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_offset[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value)</code>	<code>base -> Rn</code> <code>offset -> Qm</code> <code>value -> Qd</code>	VSTRD.64 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_offset_p[_s64](int64_t * base, uint64x2_t offset, int64x2_t value, mve_pred16_t p)</code>	<code>base -> Rn</code> <code>offset -> Qm</code> <code>value -> Qd</code> <code>p -> Rp</code>	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_offset_p[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value, mve_pred16_t p)</code>	<code>base -> Rn</code> <code>offset -> Qm</code> <code>value -> Qd</code> <code>p -> Rp</code>	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm]	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_shifted_offset[_s64](int64_t * base, uint64x2_t offset, int64x2_t value)</code>	<code>base -> Rn</code> <code>offset -> Qm</code> <code>value -> Qd</code>	VSTRD.64 Qd,[Rn,Qm, UXTW #3]	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_shifted_offset[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value)</code>	<code>base -> Rn</code> <code>offset -> Qm</code> <code>value -> Qd</code>	VSTRD.64 Qd,[Rn,Qm, UXTW #3]	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_shifted_offset_p[_s64](int64_t * base, uint64x2_t offset, int64x2_t value, mve_pred16_t p)</code>	<code>base -> Rn</code> <code>offset -> Qm</code> <code>value -> Qd</code> <code>p -> Rp</code>	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm, UXTW #3]	void -> result	MVE
<code>void [__arm_]vstrdq_scatter_shifted_offset_p[_u64](uint64_t * base, uint64x2_t offset, uint64x2_t value, mve_pred16_t p)</code>	<code>base -> Rn</code> <code>offset -> Qm</code> <code>value -> Qd</code> <code>p -> Rp</code>	VMSR P0,Rp VPST VSTRDT.64 Qd,[Rn,Qm, UXTW #3]	void -> result	MVE
<code>int64_t [__arm_]vaddlvaq[_s32](int64_t a, int32x4_t b)</code>	<code>a -> [RdaHi,RdaLo]</code> <code>b -> Qm</code>	VADDLVA.S32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE
<code>uint64_t [__arm_]vaddlvaq[_u32](uint64_t a, uint32x4_t b)</code>	<code>a -> [RdaHi,RdaLo]</code> <code>b -> Qm</code>	VADDLVA.U32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE
<code>int64_t [__arm_]vaddlvaq_p[_s32](int64_t a, int32x4_t b, mve_pred16_t p)</code>	<code>a -> [RdaHi,RdaLo]</code> <code>b -> Qm</code> <code>p -> Rp</code>	VMSR P0,Rp VPST VADDLVAT.S32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint64_t [__arm_]vaddlvq_p[_u32](uint64_t a, uint32x4_t b, mve_pred16_t p)	a -> [RdaHi,RdaLo] b -> Qm p -> Rp	VMSR P0,Rp VPST VADDLVAT.U32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vaddlvq[_s32](int32x4_t a)	a -> Qm	VADDLV.S32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [__arm_]vaddlvq[_u32](uint32x4_t a)	a -> Qm	VADDLV.U32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vaddlvq_p[_s32](int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VADDLVT.S32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [__arm_]vaddlvq_p[_u32](uint32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VADDLVT.U32 RdaLo,RdaHi,Qm	[RdaHi,RdaLo] -> result	MVE
int32_t [__arm_]vaddvq[_s8](int32_t a, int8x16_t b)	a -> Rda b -> Qm	VADDVA.S8 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vaddvq[_s16](int32_t a, int16x8_t b)	a -> Rda b -> Qm	VADDVA.S16 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vaddvq[_s32](int32_t a, int32x4_t b)	a -> Rda b -> Qm	VADDVA.S32 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq[_u8](uint32_t a, uint8x16_t b)	a -> Rda b -> Qm	VADDVA.U8 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq[_u16](uint32_t a, uint16x8_t b)	a -> Rda b -> Qm	VADDVA.U16 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq[_u32](uint32_t a, uint32x4_t b)	a -> Rda b -> Qm	VADDVA.U32 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vaddvq_p[_s8](int32_t a, int8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VADDVAT.S8 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vaddvq_p[_s16](int32_t a, int16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VADDVAT.S16 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vaddvq_p[_s32](int32_t a, int32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VADDVAT.S32 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq_p[_u8](uint32_t a, uint8x16_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VADDVAT.U8 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq_p[_u16](uint32_t a, uint16x8_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VADDVAT.U16 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq_p[_u32](uint32_t a, uint32x4_t b, mve_pred16_t p)	a -> Rda b -> Qm p -> Rp	VMSR P0,Rp VPST VADDVAT.U32 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vaddvq[_s8](int8x16_t a)	a -> Qm	VADDV.S8 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vaddvq[_s16](int16x8_t a)	a -> Qm	VADDV.S16 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vaddvq[_s32](int32x4_t a)	a -> Qm	VADDV.S32 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq[_u8](uint8x16_t a)	a -> Qm	VADDV.U8 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq[_u16](uint16x8_t a)	a -> Qm	VADDV.U16 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq[_u32](uint32x4_t a)	a -> Qm	VADDV.U32 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vaddvq_p[_s8](int8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VADDVT.S8 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vaddvq_p[_s16](int16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VADDVT.S16 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vaddvq_p[_s32](int32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VADDVT.S32 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq_p[_u8](uint8x16_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VADDVT.U8 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq_p[_u16](uint16x8_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VADDVT.U16 Rda,Qm	Rda -> result	MVE
uint32_t [__arm_]vaddvq_p[_u32](uint32x4_t a, mve_pred16_t p)	a -> Qm p -> Rp	VMSR P0,Rp VPST VADDVT.U32 Rda,Qm	Rda -> result	MVE
int32_t [__arm_]vmladavaq[_s8](int32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVA.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmladavaq[_s16](int32_t a, int16x8_t b, int16x8_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVA.S16 Rda,Qn,Qm	Rda -> result	MVE

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

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint32_t [__arm_]vmladavaxq[_u8](uint32_t a, uint8x16_t b, uint8x16_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVAX.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [__arm_]vmladavaxq[_u16](uint32_t a, uint16x8_t b, uint16x8_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVAX.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [__arm_]vmladavaxq[_u32](uint32_t a, uint32x4_t b, uint32x4_t c)	a -> Rda b -> Qn c -> Qm	VMLADAVAX.U32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmladavaxq_p[_s8](int32_t a, int8x16_t b, int8x16_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAXT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmladavaxq_p[_s16](int32_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAXT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmladavaxq_p[_s32](int32_t a, int32x4_t b, int32x4_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVAXT.S32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [__arm_]vmladavaxq_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 VMLADAVAXT.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [__arm_]vmladavaxq_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 VMLADAVAXT.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [__arm_]vmladavaxq_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 VMLADAVAXT.U32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmladavxq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMLADAVX.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmladavxq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMLADAVX.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmladavxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMLADAVX.S32 Rda,Qn,Qm	Rda -> result	MVE
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, uint16x8_t b)	a -> Qn b -> Qm	VMLADAVX.U16 Rda,Qn,Qm	Rda -> result	MVE
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 b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVXT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmladavxq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVXT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmladavxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVXT.S32 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [__arm_]vmladavxq_p[_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVXT.U8 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [__arm_]vmladavxq_p[_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVXT.U16 Rda,Qn,Qm	Rda -> result	MVE
uint32_t [__arm_]vmladavxq_p[_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLADAVXT.U32 Rda,Qn,Qm	Rda -> result	MVE
int64_t [__arm_]vmlaldavaq[_s16](int64_t a, int16x8_t b, int16x8_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVA.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmlaldavaq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVA.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
uint64_t [__arm_]vmlaldavaq[_u16](uint64_t a, uint16x8_t b, uint16x8_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVA.U16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE

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
int64_t [__arm_]vmlaldavaxq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMLALDAVAX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
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_]vmlaldavxq_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_]vmlaldavxq_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 b, int8_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [__arm_]vmlasq[_n_s16](int16x8_t a, int16x8_t b, int16_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [__arm_]vmlasq[_n_s32](int32x4_t a, int32x4_t b, int32_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vmlasq[_n_u8](uint8x16_t a, uint8x16_t b, uint8_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [__arm_]vmlasq[_n_u16](uint16x8_t a, uint16x8_t b, uint16_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [__arm_]vmlasq[_n_u32](uint32x4_t a, uint32x4_t b, uint32_t c)	a -> Qda b -> Qn c -> Rm	VMLAS.U32 Qda,Qn,Rm	Qda -> result	MVE
int8x16_t [__arm_]vmlasq_m[_n_s8](int8x16_t a, int8x16_t b, int8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.S8 Qda,Qn,Rm	Qda -> result	MVE
int16x8_t [__arm_]vmlasq_m[_n_s16](int16x8_t a, int16x8_t b, int16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.S16 Qda,Qn,Rm	Qda -> result	MVE
int32x4_t [__arm_]vmlasq_m[_n_s32](int32x4_t a, int32x4_t b, int32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.S32 Qda,Qn,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vmlasq_m[_n_u8](uint8x16_t a, uint8x16_t b, uint8_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.U8 Qda,Qn,Rm	Qda -> result	MVE
uint16x8_t [__arm_]vmlasq_m[_n_u16](uint16x8_t a, uint16x8_t b, uint16_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.U16 Qda,Qn,Rm	Qda -> result	MVE
uint32x4_t [__arm_]vmlasq_m[_n_u32](uint32x4_t a, uint32x4_t b, uint32_t c, mve_pred16_t p)	a -> Qda b -> Qn c -> Rm p -> Rp	VMSR P0,Rp VPST VMLAST.U32 Qda,Qn,Rm	Qda -> result	MVE
int32_t [__arm_]vmlsdavaq[_s8](int32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn c -> Qm	VMLSDAVA.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmlsdavaq[_s16](int32_t a, int16x8_t b, int16x8_t c)	a -> Rda b -> Qn c -> Qm	VMLSDAVA.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmlsdavaq[_s32](int32_t a, int32x4_t b, int32x4_t c)	a -> Rda b -> Qn c -> Qm	VMLSDAVA.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmlsdavaq_p[_s8](int32_t a, int8x16_t b, int8x16_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVAT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmlsdavaq_p[_s16](int32_t a, int16x8_t b, int16x8_t c, mve_pred16_t p)	a -> Rda b -> Qn c -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVAT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmlsdavaq_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 VMLSDAVAT.S32 Rda,Qn,Qm	Rda -> result	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 b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmlsdavq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMLSDAVT.S16 Rda,Qn,Qm	Rda -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int32_t [__arm_]vmsldavq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMSLDAVT.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq[_s8](int32_t a, int8x16_t b, int8x16_t c)	a -> Rda b -> Qn c -> Qm	VMSLDAVAX.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq[_s16](int32_t a, int16x8_t b, int16x8_t c)	a -> Rda b -> Qn c -> Qm	VMSLDAVAX.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq[_s32](int32_t a, int32x4_t b, int32x4_t c)	a -> Rda b -> Qn c -> Qm	VMSLDAVAX.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq_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 VMSLDAVAXT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq_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 VMSLDAVAXT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq_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 VMSLDAVAXT.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VMSLDAVX.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMSLDAVX.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMSLDAVX.S32 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq_p[_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMSLDAVXT.S8 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMSLDAVXT.S16 Rda,Qn,Qm	Rda -> result	MVE
int32_t [__arm_]vmsldavxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMSLDAVXT.S32 Rda,Qn,Qm	Rda -> result	MVE
int64_t [__arm_]vmsldavq[_s16](int64_t a, int16x8_t b, int16x8_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMSLDAVA.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMSLDAVA.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavq_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 VMSLDAVAT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavq_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 VMSLDAVAT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMSLDAV.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMSLDAV.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMSLDAVT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMSLDAVT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavxq[_s16](int64_t a, int16x8_t b, int16x8_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMSLDAVAX.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavxq[_s32](int64_t a, int32x4_t b, int32x4_t c)	a -> [RdaHi,RdaLo] b -> Qn c -> Qm	VMSLDAVAX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int64_t [__arm_]vmsldavxq_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 VMSLDAVXT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavxq_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 VMSLDAVXT.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavxq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VMSLDAVX.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VMSLDAVX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavxq_p[_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMSLDAVXT.S16 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vmsldavxq_p[_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VMSLDAVXT.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 p -> Rp	VMSR P0,Rp VPST VHADDT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vhaddq_m[_n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHADDT.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vhaddq_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 VHADDT.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [__arm_]vhaddq_m[_n_u8](uint8x16_t inactive, uint8x16_t a, uint8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHADDT.U8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [__arm_]vhaddq_m[_n_u16](uint16x8_t inactive, uint16x8_t a, uint16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHADDT.U16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [__arm_]vhaddq_m[_n_u32](uint32x4_t inactive, uint32x4_t a, uint32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHADDT.U32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [__arm_]vhaddq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHADDT.S8 Qd,Qn,Qm	Qd -> result	MVE

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 VHADD.T.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 VHADD.T.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 VHADD.T.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 VHADD.T.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 VHADD.T.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 -> Qm	VHCADD.S32 Qd,Qn,Qm,#90	Qd -> result	MVE
int8x16_t [__arm_]vhcaddq_rot90_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VHCADD.T.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 VHCADD.T.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 VHCADD.T.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 VHCADD.T.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 VHCADD.T.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 VHCADD.T.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_]vhsuq[u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VHSUB.U16 Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vhsuq[u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VHSUB.U32 Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vhsuq_m[n_s8](int8x16_t inactive, int8x16_t a, int8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHSUBT.S8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vhsuq_m[n_s16](int16x8_t inactive, int16x8_t a, int16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHSUBT.S16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vhsuq_m[n_s32](int32x4_t inactive, int32x4_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VHSUBT.S32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [__arm_]vhsuq_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_]vhsuq_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_]vhsuq_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_]vhsuq_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_]vhsuq_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_]vhsuq_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_]vhsuq_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_]vhsuq_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_]vhsuq_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 VRHADD.T.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 VRHADD.T.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 VRHADD.T.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 VFMA.T.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 VFMA.T.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 VFMA.T.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 VFMA.T.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	VFMA.S.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	VFMA.S.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 VFMA.S.T.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 VFMA.S.T.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	VFMA.S.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	VFMA.S.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 VFMA.S.T.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 VFMA.S.T.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_]vrmlslavhaq[_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_]vrmlslavhaq_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_]vrmlslavhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VRMLSLDAVH.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vrmlslavhq_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_]vrmlslavhaxq[_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_]vrmlslavhaxq_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_]vrmlslavhxq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VRMLSLDAVHX.S32 RdaLo,RdaHi,Qn,Qm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]vrmlslavhxq_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_]vrmluhq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VRMULH.S8 Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vrmluhq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VRMULH.S16 Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vrmluhq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VRMULH.S32 Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vrmluhq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VRMULH.U8 Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vrmluhq[_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_]vcvtq_s16_f16(float16x8_t a)	a -> Qm	VCVTA.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vcvtq_s32_f32(float32x4_t a)	a -> Qm	VCVTA.S32.F32 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vcvtq_u16_f16(float16x8_t a)	a -> Qm	VCVTA.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vcvtq_u32_f32(float32x4_t a)	a -> Qm	VCVTA.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vcvtq_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_]vcvtq_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_]vcvtq_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_]vcvtq_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_]vcvtq_s16_f16(float16x8_t a)	a -> Qm	VCVTN.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vcvtq_s32_f32(float32x4_t a)	a -> Qm	VCVTN.S32.F32 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vcvtq_u16_f16(float16x8_t a)	a -> Qm	VCVTN.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vcvtq_u32_f32(float32x4_t a)	a -> Qm	VCVTN.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vcvtq_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_]vcvtq_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_]vcvtq_m[_u16_f16](uint16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTNT.U16.F16 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 VCVTNT.U32.F32 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vcvtq_s16_f16(float16x8_t a)	a -> Qm	VCVTP.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vcvtq_s32_f32(float32x4_t a)	a -> Qm	VCVTP.S32.F32 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vcvtq_u16_f16(float16x8_t a)	a -> Qm	VCVTP.U16.F16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vcvtq_u32_f32(float32x4_t a)	a -> Qm	VCVTP.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vcvtq_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_]vcvtq_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_]vcvtq_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_]vcvtq_m[_u32_f32](uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTPT.U32.F32 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vcvtm_s16_f16(float16x8_t a)	a -> Qm	VCVTM.S16.F16 Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vcvtm_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	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vcvtmq_u32_f32(float32x4_t a)	a -> Qm	VCVTM.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vcvtmq_m_s16_f16(int16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTMT.S16.F16 Qd,Qm	Qd -> result	MVE
int32x4_t [__arm_]vcvtmq_m_s32_f32(int32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTMT.S32.F32 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vcvtmq_m_u16_f16(uint16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTMT.U16.F16 Qd,Qm	Qd -> result	MVE
uint32x4_t [__arm_]vcvtmq_m_u32_f32(uint32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTMT.U32.F32 Qd,Qm	Qd -> result	MVE
float16x8_t [__arm_]vcvtbq_f16_f32(float16x8_t a, float32x4_t b)	a -> Qd b -> Qm	VCVTB.F16.F32 Qd,Qm	Qd -> result	MVE
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, float32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VCVTBT.F16.F32 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vcvtbq_m_f32_f16(float32x4_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTBT.F32.F16 Qd,Qm	Qd -> result	MVE
float16x8_t [__arm_]vcvttq_f16_f32(float16x8_t a, float32x4_t b)	a -> Qd b -> Qm	VCVTT.F16.F32 Qd,Qm	Qd -> result	MVE
float32x4_t [__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, float32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VCVTTT.F16.F32 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vcvttq_m_f32_f16(float32x4_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTTT.F32.F16 Qd,Qm	Qd -> result	MVE
float16x8_t [__arm_]vcvtq_f16_s16(int16x8_t a)	a -> Qm	VCVT.F16.S16 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vcvtq_f16_u16(uint16x8_t a)	a -> Qm	VCVT.F16.U16 Qd,Qm	Qd -> result	MVE/NEON
float32x4_t [__arm_]vcvtq_f32_s32(int32x4_t a)	a -> Qm	VCVT.F32.S32 Qd,Qm	Qd -> result	MVE/NEON
float32x4_t [__arm_]vcvtq_f32_u32(uint32x4_t a)	a -> Qm	VCVT.F32.U32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vcvtq_m_f16_s16(float16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.F16.S16 Qd,Qm	Qd -> result	MVE
float16x8_t [__arm_]vcvtq_m_f16_u16(float16x8_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.F16.U16 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vcvtq_m_f32_s32(float32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.F32.S32 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vcvtq_m_f32_u32(float32x4_t inactive, uint32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.F32.U32 Qd,Qm	Qd -> result	MVE
float16x8_t [__arm_]vcvtq_n_f16_s16(int16x8_t a, const int imm6)	a -> Qm 1 <= imm6 <= 16	VCVT.F16.S16 Qd,Qm,imm6	Qd -> result	MVE/NEON
float16x8_t [__arm_]vcvtq_n_f16_u16(uint16x8_t a, const int imm6)	a -> Qm 1 <= imm6 <= 16	VCVT.F16.U16 Qd,Qm,imm6	Qd -> result	MVE/NEON
float32x4_t [__arm_]vcvtq_n_f32_s32(int32x4_t a, const int imm6)	a -> Qm 1 <= imm6 <= 32	VCVT.F32.S32 Qd,Qm,imm6	Qd -> result	MVE/NEON
float32x4_t [__arm_]vcvtq_n_f32_u32(uint32x4_t a, const int imm6)	a -> Qm 1 <= imm6 <= 32	VCVT.F32.U32 Qd,Qm,imm6	Qd -> result	MVE/NEON
float16x8_t [__arm_]vcvtq_m_n_f16_s16(float16x8_t inactive, int16x8_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm6 <= 16 p -> Rp	VMSR P0,Rp VPST VCVTT.F16.S16 Qd,Qm,imm6	Qd -> result	MVE
float16x8_t [__arm_]vcvtq_m_n_f16_u16(float16x8_t inactive, uint16x8_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm6 <= 16 p -> Rp	VMSR P0,Rp VPST VCVTT.F16.U16 Qd,Qm,imm6	Qd -> result	MVE
float32x4_t [__arm_]vcvtq_m_n_f32_s32(float32x4_t inactive, int32x4_t a, const int imm6, mve_pred16_t p)	inactive -> Qd a -> Qm 1 <= imm6 <= 32 p -> Rp	VMSR P0,Rp VPST VCVTT.F32.S32 Qd,Qm,imm6	Qd -> result	MVE

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)	a -> Qm	VCVT.U32.F32 Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vcvtq_m[_s16_f16](int16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VCVTT.S16.F16 Qd,Qm	Qd -> result	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)	a -> Qm	VRINTZ.F32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vrndq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTZT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vrndq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTZT.F32 Qd,Qm	Qd -> result	MVE
float16x8_t [__arm_]vrndq[_f16](float16x8_t a)	a -> Qm	VRINTN.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vrndq[_f32](float32x4_t a)	a -> Qm	VRINTN.F32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vrndq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTNT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vrndq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST 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, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTMT.F32 Qd,Qm	Qd -> result	MVE
float16x8_t [__arm_]vrndpq[_f16](float16x8_t a)	a -> Qm	VRINTP.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vrndpq[_f32](float32x4_t a)	a -> Qm	VRINTP.F32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vrndpq_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTPT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vrndpq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTPT.F32 Qd,Qm	Qd -> result	MVE
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 -> Qm p -> Rp	VMSR P0,Rp VPST VRINTAT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vrndaq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTAT.F32 Qd,Qm	Qd -> result	MVE
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, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTXT.F16 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vrndxq_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VRINTXT.F32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vandq[_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vandq[_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vandq[_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vandq[_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vandq[_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vandq[_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vandq[_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
float32x4_t [__arm_]vandq[_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VAND Qd,Qn,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vandq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vandq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vandq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vandq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vandq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [__arm_]vandq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [__arm_]vandq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [__arm_]vandq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VANDT Qd,Qn,Qm	Qd -> result	MVE

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 AdvSIMDExpandImm	VBIC.I16 Qda,#imm	Qda -> result	MVE
int32x4_t [__arm_]vbicq[_n_s32](int32x4_t a, const int imm)	a -> Qda imm in AdvSIMDExpandImm	VBIC.I32 Qda,#imm	Qda -> result	MVE
uint16x8_t [__arm_]vbicq[_n_u16](uint16x8_t a, const int imm)	a -> Qda imm in AdvSIMDExpandImm	VBIC.I16 Qda,#imm	Qda -> result	MVE
uint32x4_t [__arm_]vbicq[_n_u32](uint32x4_t a, const int imm)	a -> Qda imm in AdvSIMDExpandImm	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 AdvSIMDExpandImm 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 AdvSIMDExpandImm 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 AdvSIMDExpandImm p -> Rp	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)	a -> Qda imm in AdvSIMDExpandImm p -> Rp	VMSR P0,Rp VPST VBICT.I32 Qda,#imm	Qda -> result	MVE
int8x16_t [__arm_]vbrsq[_n_s8](int8x16_t a, int32_t b)	a -> Qn b -> Rm	VBR SR.8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vbrsq[_n_s16](int16x8_t a, int32_t b)	a -> Qn b -> Rm	VBR SR.16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vbrsq[_n_s32](int32x4_t a, int32_t b)	a -> Qn b -> Rm	VBR SR.32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [__arm_]vbrsq[_n_u8](uint8x16_t a, int32_t b)	a -> Qn b -> Rm	VBR SR.8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [__arm_]vbrsq[_n_u16](uint16x8_t a, int32_t b)	a -> Qn b -> Rm	VBR SR.16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [__arm_]vbrsq[_n_u32](uint32x4_t a, int32_t b)	a -> Qn b -> Rm	VBR SR.32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [__arm_]vbrsq[_n_f16](float16x8_t a, int32_t b)	a -> Qn b -> Rm	VBR SR.16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [__arm_]vbrsq[_n_f32](float32x4_t a, int32_t b)	a -> Qn b -> Rm	VBR SR.32 Qd,Qn,Rm	Qd -> result	MVE
int8x16_t [__arm_]vbrsq_m[_n_s8](int8x16_t inactive, int8x16_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBR SRT.8 Qd,Qn,Rm	Qd -> result	MVE
int16x8_t [__arm_]vbrsq_m[_n_s16](int16x8_t inactive, int16x8_t a, int32_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Rm p -> Rp	VMSR P0,Rp VPST VBR SRT.16 Qd,Qn,Rm	Qd -> result	MVE
int32x4_t [__arm_]vbrsq_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 VBR SRT.32 Qd,Qn,Rm	Qd -> result	MVE
uint8x16_t [__arm_]vbrsq_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 VBR SRT.8 Qd,Qn,Rm	Qd -> result	MVE
uint16x8_t [__arm_]vbrsq_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 VBR SRT.16 Qd,Qn,Rm	Qd -> result	MVE
uint32x4_t [__arm_]vbrsq_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 VBR SRT.32 Qd,Qn,Rm	Qd -> result	MVE
float16x8_t [__arm_]vbrsq_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 VBR SRT.16 Qd,Qn,Rm	Qd -> result	MVE
float32x4_t [__arm_]vbrsq_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 VBR SRT.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, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]veorq_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 VEORT Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]veorq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [__arm_]veorq_m[_u8](uint8x16_t inactive, uint8x16_t a, uint8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [__arm_]veorq_m[_u16](uint16x8_t inactive, uint16x8_t a, uint16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [__arm_]veorq_m[_u32](uint32x4_t inactive, uint32x4_t a, uint32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [__arm_]veorq_m[_f16](float16x8_t inactive, float16x8_t a, float16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [__arm_]veorq_m[_f32](float32x4_t inactive, float32x4_t a, float32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPST VEORT Qd,Qn,Qm	Qd -> result	MVE
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, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.S8 Qd,Qm	Qd -> result	MVE
int32x4_t [__arm_]vmovlbq_m[_s16](int32x4_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vmovlbq_m[_u8](uint16x8_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.U8 Qd,Qm	Qd -> result	MVE
uint32x4_t [__arm_]vmovlbq_m[_u16](uint32x4_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLBT.U16 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vmovltq[_s8](int8x16_t a)	a -> Qm	VMOVLT.S8 Qd,Qm	Qd -> result	MVE
int32x4_t [__arm_]vmovltq[_s16](int16x8_t a)	a -> Qm	VMOVLT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vmovltq[_u8](uint8x16_t a)	a -> Qm	VMOVLT.U8 Qd,Qm	Qd -> result	MVE
uint32x4_t [__arm_]vmovltq[_u16](uint16x8_t a)	a -> Qm	VMOVLT.U16 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vmovltq_m[_s8](int16x8_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.S8 Qd,Qm	Qd -> result	MVE
int32x4_t [__arm_]vmovltq_m[_s16](int32x4_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vmovltq_m[_u8](uint16x8_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.U8 Qd,Qm	Qd -> result	MVE
uint32x4_t [__arm_]vmovltq_m[_u16](uint32x4_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMOVLTT.U16 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vmovnbq[_s16](int8x16_t a, int16x8_t b)	a -> Qd b -> Qm	VMOVNB.I16 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vmovnbq[_s32](int16x8_t a, int32x4_t b)	a -> Qd b -> Qm	VMOVNB.I32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vmovnbq[_u16](uint8x16_t a, uint16x8_t b)	a -> Qd b -> Qm	VMOVNB.I16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vmovnbq[_u32](uint16x8_t a, uint32x4_t b)	a -> Qd b -> Qm	VMOVNB.I32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vmovnbq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNB.I16 Qd,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [__arm_]vmoynbq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNBT.I32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vmoynbq_m[_u16](uint8x16_t a, uint16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNBT.I16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vmoynbq_m[_u32](uint16x8_t a, uint32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNBT.I32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vmovntq[_s16](int8x16_t a, int16x8_t b)	a -> Qd b -> Qm	VMOVNT.I16 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vmovntq[_s32](int16x8_t a, int32x4_t b)	a -> Qd b -> Qm	VMOVNT.I32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vmovntq[_u16](uint8x16_t a, uint16x8_t b)	a -> Qd b -> Qm	VMOVNT.I16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vmovntq[_u32](uint16x8_t a, uint32x4_t b)	a -> Qd b -> Qm	VMOVNT.I32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vmovntq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNTT.I16 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vmovntq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNTT.I32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vmovntq_m[_u16](uint8x16_t a, uint16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNTT.I16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vmovntq_m[_u32](uint16x8_t a, uint32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VMOVNTT.I32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vmvntq[_s8](int8x16_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vmvntq[_s16](int16x8_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vmvntq[_s32](int32x4_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vmvntq[_u8](uint8x16_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vmvntq[_u16](uint16x8_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vmvntq[_u32](uint32x4_t a)	a -> Qm	VMVN Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vmvntq_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vmvntq_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int32x4_t [__arm_]vmvntq_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vmvntq_m[_u8](uint8x16_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vmvntq_m[_u16](uint16x8_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
uint32x4_t [__arm_]vmvntq_m[_u32](uint32x4_t inactive, uint32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VMVNT Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vmvntq_n_s16(const int16_t imm)	imm in AdvSIMDExpandImm	VMVN.I16 Qd,#imm	Qd -> result	MVE
int32x4_t [__arm_]vmvntq_n_s32(const int32_t imm)	imm in AdvSIMDExpandImm	VMVN.I32 Qd,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vmvntq_n_u16(const uint16_t imm)	imm in AdvSIMDExpandImm	VMVN.I16 Qd,#imm	Qd -> result	MVE
uint32x4_t [__arm_]vmvntq_n_u32(const uint32_t imm)	imm in AdvSIMDExpandImm	VMVN.I32 Qd,#imm	Qd -> result	MVE
int16x8_t [__arm_]vmvntq_m[_n_s16](int16x8_t inactive, const int16_t imm, mve_pred16_t p)	inactive -> Qd imm in AdvSIMDExpandImm p -> Rp	VMSR P0,Rp VPST VMVNT.I16 Qd,#imm	Qd -> result	MVE
int32x4_t [__arm_]vmvntq_m[_n_s32](int32x4_t inactive, const int32_t imm, mve_pred16_t p)	inactive -> Qd imm in AdvSIMDExpandImm p -> Rp	VMSR P0,Rp VPST VMVNT.I32 Qd,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [__arm_]vmvnm_m[<i>n_u16</i>](uint16x8_t inactive, const uint16_t imm, mve_pred16_t p)	inactive -> Qd imm in AdvSIMDExpandImm p -> Rp	VMSR P0,Rp VPST VMVNT.I16 Qd,#imm	Qd -> result	MVE
uint32x4_t [__arm_]vmvnm_m[<i>n_u32</i>](uint32x4_t inactive, const uint32_t imm, mve_pred16_t p)	inactive -> Qd imm in AdvSIMDExpandImm p -> Rp	VMSR P0,Rp VPST VMVNT.I32 Qd,#imm	Qd -> result	MVE
mve_pred16_t [__arm_]vpnot(mve_pred16_t a)	a -> Rp	VMSR P0,Rp VPNOT VMRS Rt,P0	Rt -> result	MVE
int8x16_t [__arm_]vpseql_s8](int8x16_t a, int8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vpseql_s16](int16x8_t a, int16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qd -> result	MVE
int32x4_t [__arm_]vpseql_s32](int32x4_t a, int32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qd -> result	MVE
int64x2_t [__arm_]vpseql_s64](int64x2_t a, int64x2_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vpseql_u8](uint8x16_t a, uint8x16_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vpseql_u16](uint16x8_t a, uint16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qd -> result	MVE
uint32x4_t [__arm_]vpseql_u32](uint32x4_t a, uint32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qd -> result	MVE
uint64x2_t [__arm_]vpseql_u64](uint64x2_t a, uint64x2_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [__arm_]vpseql_f16](float16x8_t a, float16x8_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [__arm_]vpseql_f32](float32x4_t a, float32x4_t b, mve_pred16_t p)	a -> Qn b -> Qm p -> Rp	VMSR P0,Rp VPSEL Qd,Qn,Qm	Qd -> result	MVE
float16x8_t [__arm_]vornq_f16](float16x8_t a, float16x8_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE
float32x4_t [__arm_]vornq_f32](float32x4_t a, float32x4_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [__arm_]vornq_s8](int8x16_t a, int8x16_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
int16x8_t [__arm_]vornq_s16](int16x8_t a, int16x8_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
int32x4_t [__arm_]vornq_s32](int32x4_t a, int32x4_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vornq_u8](uint8x16_t a, uint8x16_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vornq_u16](uint16x8_t a, uint16x8_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vornq_u32](uint32x4_t a, uint32x4_t b)	a -> Qn b -> Qm	VORN Qd,Qn,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vornq_m_f16](float16x8_t inactive, float16x8_t a, float16x8_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
float32x4_t [__arm_]vornq_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 VORNT Qd,Qn,Qm	Qd -> result	MVE
int8x16_t [__arm_]vornq_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 VORNT Qd,Qn,Qm	Qd -> result	MVE
int16x8_t [__arm_]vornq_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 VORNT Qd,Qn,Qm	Qd -> result	MVE

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 AdvSIMDExpandImm	VORR.I16 Qda,#imm	Qda -> result	MVE
int32x4_t [__arm_]vorrq[_n_s32](int32x4_t a, const int imm)	a -> Qda imm in AdvSIMDExpandImm	VORR.I32 Qda,#imm	Qda -> result	MVE
uint16x8_t [__arm_]vorrq[_n_u16](uint16x8_t a, const int imm)	a -> Qda imm in AdvSIMDExpandImm	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 imm)	a -> Qda imm in AdvSIMDExpandImm	VORR.I32 Qda,#imm	Qda -> result	MVE
int16x8_t [__arm_]vorrq_m_n[_s16](int16x8_t a, const int imm, mve_pred16_t p)	a -> Qda imm in AdvSIMDExpandImm p -> Rp	VMSR P0,Rp VPST VORRT.I16 Qda,#imm	Qda -> result	MVE
int32x4_t [__arm_]vorrq_m_n[_s32](int32x4_t a, const int imm, mve_pred16_t p)	a -> Qda imm in AdvSIMDExpandImm p -> Rp	VMSR P0,Rp VPST VORRT.I32 Qda,#imm	Qda -> result	MVE
uint16x8_t [__arm_]vorrq_m_n[_u16](uint16x8_t a, const int imm, mve_pred16_t p)	a -> Qda imm in AdvSIMDExpandImm p -> Rp	VMSR P0,Rp VPST VORRT.I16 Qda,#imm	Qda -> result	MVE
uint32x4_t [__arm_]vorrq_m_n[_u32](uint32x4_t a, const int imm, mve_pred16_t p)	a -> Qda imm in AdvSIMDExpandImm p -> Rp	VMSR P0,Rp VPST VORRT.I32 Qda,#imm	Qda -> result	MVE
int8x16_t [__arm_]vqmovnbq[_s16](int8x16_t a, int16x8_t b)	a -> Qd b -> Qm	VQMOVNB.S16 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqmovnbq[_s32](int16x8_t a, int32x4_t b)	a -> Qd b -> Qm	VQMOVNB.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vqmovnbq[_u16](uint8x16_t a, uint16x8_t b)	a -> Qd b -> Qm	VQMOVNB.U16 Qd,Qm	Qd -> result	MVE
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, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVNBT.S16 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqmovnbq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVNBT.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vqmovnbq_m[_u16](uint8x16_t a, uint16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVNBT.U16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vqmovnbq_m[_u32](uint16x8_t a, uint32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVNBT.U32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vqmovntq[_s16](int8x16_t a, int16x8_t b)	a -> Qd b -> Qm	VQMOVNT.S16 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqmovntq[_s32](int16x8_t a, int32x4_t b)	a -> Qd b -> Qm	VQMOVNT.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vqmovntq[_u16](uint8x16_t a, uint16x8_t b)	a -> Qd b -> Qm	VQMOVNT.U16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vqmovntq[_u32](uint16x8_t a, uint32x4_t b)	a -> Qd b -> Qm	VQMOVNT.U32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vqmovntq_m[_s16](int8x16_t a, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVNTT.S16 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vqmovntq_m[_s32](int16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVNTT.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vqmovntq_m[_u16](uint8x16_t a, uint16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVNTT.U16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vqmovntq_m[_u32](uint16x8_t a, uint32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVNTT.U32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vqmovunbq[_s16](uint8x16_t a, int16x8_t b)	a -> Qd b -> Qm	VQMOVUNB.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vqmovunbq[_s32](uint16x8_t a, int32x4_t b)	a -> Qd b -> Qm	VQMOVUNB.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vqmovunbq_m[_s16](uint8x16_t a, int16x8_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVUNBT.S16 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vqmovunbq_m[_s32](uint16x8_t a, int32x4_t b, mve_pred16_t p)	a -> Qd b -> Qm p -> Rp	VMSR P0,Rp VPST VQMOVUNBT.S32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vqmovuntq[_s16](uint8x16_t a, int16x8_t b)	a -> Qd b -> Qm	VQMOVUNT.S16 Qd,Qm	Qd -> result	MVE

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 p -> Rp	VMSR P0,Rp VPST VQMOVUNTT.S32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vqrshlq[_n_s8](int8x16_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [__arm_]vqrshlq[_n_s16](int16x8_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [__arm_]vqrshlq[_n_s32](int32x4_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vqrshlq[_n_u8](uint8x16_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [__arm_]vqrshlq[_n_u16](uint16x8_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [__arm_]vqrshlq[_n_u32](uint32x4_t a, int32_t b)	a -> Qda b -> Rm	VQRSHL.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [__arm_]vqrshlq_m[_n_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [__arm_]vqrshlq_m[_n_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [__arm_]vqrshlq_m[_n_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VQRSHLT.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vqrshlq_m[_n_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm 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_]vqsrhnbq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VQRSHRNB.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vqsrhnbq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VQRSHRNB.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqsrhnbq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VQRSHRNB.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqsrhnbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VQRSHRNB.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vqsrhnbq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMRSR P0,Rp VPST VQRSHRNB.T.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vqsrhnbq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMRSR P0,Rp VPST VQRSHRNB.T.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqsrhnbq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMRSR P0,Rp VPST VQRSHRNB.T.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqsrhnbq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMRSR P0,Rp VPST VQRSHRNB.T.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vqsrhntq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VQRSHRNT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vqsrhntq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VQRSHRNT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqsrhntq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VQRSHRNT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqsrhntq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VQRSHRNT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vqsrhntq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMRSR P0,Rp VPST VQRSHRNTT.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vqsrhntq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMRSR P0,Rp VPST VQRSHRNTT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqsrhntq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMRSR P0,Rp VPST VQRSHRNTT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqsrhntq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMRSR P0,Rp VPST VQRSHRNTT.U32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqsrhnbq[_n_s16](uint8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VQRSHRUNB.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqsrhnbq[_n_s32](uint16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VQRSHRUNB.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqsrhnbq_m[_n_s16](uint8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMRSR P0,Rp VPST VQRSHRUNBT.S16 Qd,Qm,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [__arm_]vqsrhbrunbq_m[_n_s32](uint16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQRSHRUNBT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqsrhbrunq[_n_s16](uint8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VQRSHRUNT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqsrhbrunq[_n_s32](uint16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VQRSHRUNT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqsrhbrunq_m[_n_s16](uint8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQRSHRUNTT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqsrhbrunq_m[_n_s32](uint16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= 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

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_]vqshrnbg[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VQSHRNB.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vqshrnbg[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VQSHRNB.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqshrnbg[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VQSHRNB.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqshrnbg[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VQSHRNB.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vqshrnbg_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQSHRNB.T.S16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vqshrnbg_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQSHRNB.T.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqshrnbg_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQSHRNB.T.U16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqshrnbg_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQSHRNB.T.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 l <= 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 l <= 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 l <= 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 l <= 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 l <= 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 l <= 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 l <= 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 l <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQSHRNTT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vqshrunbg[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VQSHRUNB.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqshrunbg[_n_s32](uint16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VQSHRUNB.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqshrunbg_m[_n_s16](uint8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQSHRUNB.T.S16 Qd,Qm,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
uint16x8_t [__arm_]vqshrunbq_m[_n_s32](uint16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQSHRUNBT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vqshrunbq_m[_n_s16](uint8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VQSHRUNT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqshrunbq_m[_n_s32](uint16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VQSHRUNT.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 l <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VQSHRUNTT.S16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vqshrunbq_m[_n_s32](uint16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VQSHRUNTT.S32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vrev16q[_s8](int8x16_t a)	a -> Qm	VREV16.8 Qd,Qm	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vrev16q[_u8](uint8x16_t a)	a -> Qm	VREV16.8 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vrev16q_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV16T.8 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vrev16q_m[_u8](uint8x16_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV16T.8 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]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, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV32T.8 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vrev32q_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV32T.16 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vrev32q_m[_u8](uint8x16_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV32T.8 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vrev32q_m[_u16](uint16x8_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV32T.16 Qd,Qm	Qd -> result	MVE
float16x8_t [__arm_]vrev32q_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV32T.16 Qd,Qm	Qd -> result	MVE
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)	a -> Qm	VREV64.8 Qd,Qm	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vrev64q[_u16](uint16x8_t a)	a -> Qm	VREV64.16 Qd,Qm	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vrev64q[_u32](uint32x4_t a)	a -> Qm	VREV64.32 Qd,Qm	Qd -> result	MVE/NEON
float16x8_t [__arm_]vrev64q[_f16](float16x8_t a)	a -> Qm	VREV64.16 Qd,Qm	Qd -> result	MVE/NEON
float32x4_t [__arm_]vrev64q[_f32](float32x4_t a)	a -> Qm	VREV64.32 Qd,Qm	Qd -> result	MVE/NEON
int8x16_t [__arm_]vrev64q_m[_s8](int8x16_t inactive, int8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV64T.8 Qd,Qm	Qd -> result	MVE
int16x8_t [__arm_]vrev64q_m[_s16](int16x8_t inactive, int16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV64T.16 Qd,Qm	Qd -> result	MVE
int32x4_t [__arm_]vrev64q_m[_s32](int32x4_t inactive, int32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV64T.32 Qd,Qm	Qd -> result	MVE
uint8x16_t [__arm_]vrev64q_m[_u8](uint8x16_t inactive, uint8x16_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV64T.8 Qd,Qm	Qd -> result	MVE
uint16x8_t [__arm_]vrev64q_m[_u16](uint16x8_t inactive, uint16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV64T.16 Qd,Qm	Qd -> result	MVE
uint32x4_t [__arm_]vrev64q_m[_u32](uint32x4_t inactive, uint32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV64T.32 Qd,Qm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
float16x8_t [__arm_]vrev64q_m[_f16](float16x8_t inactive, float16x8_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV64T.16 Qd,Qm	Qd -> result	MVE
float32x4_t [__arm_]vrev64q_m[_f32](float32x4_t inactive, float32x4_t a, mve_pred16_t p)	inactive -> Qd a -> Qm p -> Rp	VMSR P0,Rp VPST VREV64T.32 Qd,Qm	Qd -> result	MVE
int8x16_t [__arm_]vrshlq[_n_s8](int8x16_t a, int32_t b)	a -> Qda b -> Rm	VRSHL.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [__arm_]vrshlq[_n_s16](int16x8_t a, int32_t b)	a -> Qda b -> Rm	VRSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [__arm_]vrshlq[_n_s32](int32x4_t a, int32_t b)	a -> Qda b -> Rm	VRSHL.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vrshlq[_n_u8](uint8x16_t a, int32_t b)	a -> Qda b -> Rm	VRSHL.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [__arm_]vrshlq[_n_u16](uint16x8_t a, int32_t b)	a -> Qda b -> Rm	VRSHL.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [__arm_]vrshlq[_n_u32](uint32x4_t a, int32_t b)	a -> Qda b -> Rm	VRSHL.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [__arm_]vrshlq_m_n[_s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VRSHLT.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [__arm_]vrshlq_m_n[_s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VRSHLT.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [__arm_]vrshlq_m_n[_s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VRSHLT.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vrshlq_m_n[_u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VRSHLT.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [__arm_]vrshlq_m_n[_u16](uint16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VRSHLT.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [__arm_]vrshlq_m_n[_u32](uint32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VRSHLT.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [__arm_]vrshlq[_s8](int8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VRSHL.S8 Qd,Qm,Qn	Qd -> result	MVE/NEON
int16x8_t [__arm_]vrshlq[_s16](int16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VRSHL.S16 Qd,Qm,Qn	Qd -> result	MVE/NEON
int32x4_t [__arm_]vrshlq[_s32](int32x4_t a, int32x4_t b)	a -> Qm b -> Qn	VRSHL.S32 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vrshlq[_u8](uint8x16_t a, int8x16_t b)	a -> Qm b -> Qn	VRSHL.U8 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vrshlq[_u16](uint16x8_t a, int16x8_t b)	a -> Qm b -> Qn	VRSHL.U16 Qd,Qm,Qn	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vrshlq[_u32](uint32x4_t a, int32x4_t b)	a -> Qm b -> Qn	VRSHL.U32 Qd,Qm,Qn	Qd -> result	MVE/NEON
int8x16_t [__arm_]vrshlq_m[_s8](int8x16_t inactive, int8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.S8 Qd,Qm,Qn	Qd -> result	MVE
int16x8_t [__arm_]vrshlq_m[_s16](int16x8_t inactive, int16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.S16 Qd,Qm,Qn	Qd -> result	MVE
int32x4_t [__arm_]vrshlq_m[_s32](int32x4_t inactive, int32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.S32 Qd,Qm,Qn	Qd -> result	MVE
uint8x16_t [__arm_]vrshlq_m[_u8](uint8x16_t inactive, uint8x16_t a, int8x16_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.U8 Qd,Qm,Qn	Qd -> result	MVE
uint16x8_t [__arm_]vrshlq_m[_u16](uint16x8_t inactive, uint16x8_t a, int16x8_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.U16 Qd,Qm,Qn	Qd -> result	MVE
uint32x4_t [__arm_]vrshlq_m[_u32](uint32x4_t inactive, uint32x4_t a, int32x4_t b, mve_pred16_t p)	inactive -> Qd a -> Qm b -> Qn p -> Rp	VMSR P0,Rp VPST VRSHLT.U32 Qd,Qm,Qn	Qd -> result	MVE
int8x16_t [__arm_]vshlcq[_s8](int8x16_t a, uint32_t *b, const int imm)	a -> Qda *b -> Rdm 1 <= imm <= 32	VSHLC Qda,Rdm,#imm	Qda -> result Rdm -> *b	MVE

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 p -> Rp	VMSR P0,Rp VPST VSHLT.S8 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vshlq_m_n[s16](int16x8_t inactive, int16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VSHLT.S16 Qd,Qm,#imm	Qd -> result	MVE
int32x4_t [__arm_]vshlq_m_n[s32](int32x4_t inactive, int32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VSHLT.S32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vshlq_m_n[u8](uint8x16_t inactive, uint8x16_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 7 p -> Rp	VMSR P0,Rp VPST VSHLT.U8 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vshlq_m_n[u16](uint16x8_t inactive, uint16x8_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 15 p -> Rp	VMSR P0,Rp VPST VSHLT.U16 Qd,Qm,#imm	Qd -> result	MVE
uint32x4_t [__arm_]vshlq_m_n[u32](uint32x4_t inactive, uint32x4_t a, const int imm, mve_pred16_t p)	inactive -> Qd a -> Qm 0 <= imm <= 31 p -> Rp	VMSR P0,Rp VPST VSHLT.U32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vshlq_r[s8](int8x16_t a, int32_t b)	a -> Qda b -> Rm	VSHL.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [__arm_]vshlq_r[s16](int16x8_t a, int32_t b)	a -> Qda b -> Rm	VSHL.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [__arm_]vshlq_r[s32](int32x4_t a, int32_t b)	a -> Qda b -> Rm	VSHL.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vshlq_r[u8](uint8x16_t a, int32_t b)	a -> Qda b -> Rm	VSHL.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [__arm_]vshlq_r[u16](uint16x8_t a, int32_t b)	a -> Qda b -> Rm	VSHL.U16 Qda,Rm	Qda -> result	MVE
uint32x4_t [__arm_]vshlq_r[u32](uint32x4_t a, int32_t b)	a -> Qda b -> Rm	VSHL.U32 Qda,Rm	Qda -> result	MVE
int8x16_t [__arm_]vshlq_m_r[s8](int8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.S8 Qda,Rm	Qda -> result	MVE
int16x8_t [__arm_]vshlq_m_r[s16](int16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.S16 Qda,Rm	Qda -> result	MVE
int32x4_t [__arm_]vshlq_m_r[s32](int32x4_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.S32 Qda,Rm	Qda -> result	MVE
uint8x16_t [__arm_]vshlq_m_r[u8](uint8x16_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.U8 Qda,Rm	Qda -> result	MVE
uint16x8_t [__arm_]vshlq_m_r[u16](uint16x8_t a, int32_t b, mve_pred16_t p)	a -> Qda b -> Rm p -> Rp	VMSR P0,Rp VPST VSHLT.U16 Qda,Rm	Qda -> result	MVE
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_]vrshrbq[n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm 1 <= imm <= 8	VRSHRNB.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vrshrbq[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_]vrshrbq[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_]vrshrbq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VRSHRNB.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vrshrbq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNB.T.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vrshrbq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNB.T.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vrshrbq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNB.T.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vrshrbq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNB.T.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vrshrtq[_n_s16](int8x16_t a, int16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VRSHRNT.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vrshrtq[_n_s32](int16x8_t a, int32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VRSHRNT.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vrshrtq[_n_u16](uint8x16_t a, uint16x8_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 8	VRSHRNT.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vrshrtq[_n_u32](uint16x8_t a, uint32x4_t b, const int imm)	a -> Qd b -> Qm l <= imm <= 16	VRSHRNT.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vrshrtq_m[_n_s16](int8x16_t a, int16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNT.T.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vrshrtq_m[_n_s32](int16x8_t a, int32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNT.T.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vrshrtq_m[_n_u16](uint8x16_t a, uint16x8_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 8 p -> Rp	VMSR P0,Rp VPST VRSHRNT.T.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vrshrtq_m[_n_u32](uint16x8_t a, uint32x4_t b, const int imm, mve_pred16_t p)	a -> Qd b -> Qm l <= imm <= 16 p -> Rp	VMSR P0,Rp VPST VRSHRNT.T.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vrshrq[_n_s8](int8x16_t a, const int imm)	a -> Qm l <= 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 l <= 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 l <= 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 l <= 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 l <= 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 l <= 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 l <= 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_]vshrbq[_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_]vshrbq[_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_]vshrbq[_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_]vshrbq[_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_]vshrbq_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 VSHRNB.T.I16 Qd,Qm,#imm	Qd -> result	MVE
int16x8_t [__arm_]vshrbq_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 VSHRNB.T.I32 Qd,Qm,#imm	Qd -> result	MVE
uint8x16_t [__arm_]vshrbq_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 VSHRNB.T.I16 Qd,Qm,#imm	Qd -> result	MVE
uint16x8_t [__arm_]vshrbq_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 VSHRNB.T.I32 Qd,Qm,#imm	Qd -> result	MVE
int8x16_t [__arm_]vshrtq[_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_]vshrtq[_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_]vshrtq[_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_]vshrtq[_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_]vshrtq_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 VSHRNT.T.I16 Qd,Qm,#imm	Qd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int16x8_t [__arm_]vshrtq_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_]vshrtq_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_]vshrtq_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_]vslq[_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_]vslq[_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_]vslq[_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_]vslq[_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_]vslq[_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_]vslq[_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_]vslq_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_]vslq_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_]vslq_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_]vslq_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_]vslq_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_]vslq_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_f16(float16x8_t a, const int idx)	a -> 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 -> Qn 0 <= idx <= 3	VMOV.32 Rt,Qn[idx]	Rt -> result	MVE/NEON
int8_t [__arm_]vgetq_lane_s8(int8x16_t a, const int idx)	a -> Qn 0 <= idx <= 15	VMOV.S8 Rt,Qn[idx]	Rt -> result	MVE/NEON
int16_t [__arm_]vgetq_lane_s16(int16x8_t a, const int idx)	a -> Qn 0 <= idx <= 7	VMOV.S16 Rt,Qn[idx]	Rt -> result	MVE/NEON
int32_t [__arm_]vgetq_lane_s32(int32x4_t a, const int idx)	a -> Qn 0 <= idx <= 3	VMOV.32 Rt,Qn[idx]	Rt -> result	MVE/NEON
int64_t [__arm_]vgetq_lane_s64(int64x2_t a, const int idx)	a -> Qn 0 <= idx <= 1	VMOV Rt1,Rt2,D(2*n+idx)	[Rt1,Rt2] -> result	MVE/NEON
uint8_t [__arm_]vgetq_lane_u8(uint8x16_t a, const int idx)	a -> Qn 0 <= idx <= 15	VMOV.U8 Rt,Qn[idx]	Rt -> result	MVE/NEON
uint16_t [__arm_]vgetq_lane_u16(uint16x8_t a, const int idx)	a -> Qn 0 <= idx <= 7	VMOV.U16 Rt,Qn[idx]	Rt -> result	MVE/NEON
uint32_t [__arm_]vgetq_lane_u32(uint32x4_t a, const int idx)	a -> Qn 0 <= idx <= 3	VMOV.32 Rt,Qn[idx]	Rt -> result	MVE/NEON
uint64_t [__arm_]vgetq_lane_u64(uint64x2_t a, const int idx)	a -> Qn 0 <= idx <= 1	VMOV Rt1,Rt2,D(2*n+idx)	[Rt1,Rt2] -> result	MVE/NEON
float16x8_t [__arm_]vsetq_lane_f16(float16_t a, float16x8_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 7	VMOV.16 Qd[idx],Rt	Qd -> result	MVE/NEON
float32x4_t [__arm_]vsetq_lane_f32(float32_t a, float32x4_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 3	VMOV.32 Qd[idx],Rt	Qd -> result	MVE/NEON
int8x16_t [__arm_]vsetq_lane_s8(int8_t a, int8x16_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 15	VMOV.8 Qd[idx],Rt	Qd -> result	MVE/NEON
int16x8_t [__arm_]vsetq_lane_s16(int16_t a, int16x8_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 7	VMOV.16 Qd[idx],Rt	Qd -> result	MVE/NEON
int32x4_t [__arm_]vsetq_lane_s32(int32_t a, int32x4_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 3	VMOV.32 Qd[idx],Rt	Qd -> result	MVE/NEON
int64x2_t [__arm_]vsetq_lane_s64(int64_t a, int64x2_t b, const int idx)	a -> [Rt1,Rt2] b -> Qd 0 <= idx <= 1	VMOV D(2*d+idx),Rt1,Rt2	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vsetq_lane_u8(uint8_t a, uint8x16_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 15	VMOV.8 Qd[idx],Rt	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vsetq_lane_u16(uint16_t a, uint16x8_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 7	VMOV.16 Qd[idx],Rt	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vsetq_lane_u32(uint32_t a, uint32x4_t b, const int idx)	a -> Rt b -> Qd 0 <= idx <= 3	VMOV.32 Qd[idx],Rt	Qd -> result	MVE/NEON
uint64x2_t [__arm_]vsetq_lane_u64(uint64_t a, uint64x2_t b, const int idx)	a -> [Rt1,Rt2] b -> Qd 0 <= idx <= 1	VMOV D(2*d+idx),Rt1,Rt2	Qd -> result	MVE/NEON
mve_pred16_t [__arm_]vctp8q(uint32_t a)	a -> Rn	VCTP.8 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [__arm_]vctp16q(uint32_t a)	a -> Rn	VCTP.16 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [__arm_]vctp32q(uint32_t a)	a -> Rn	VCTP.32 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [__arm_]vctp64q(uint32_t a)	a -> Rn	VCTP.64 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [__arm_]vctp8q_m(uint32_t a, mve_pred16_t p)	a -> Rn p -> Rp	VMSR P0,Rp VPST VCTPT.8 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [__arm_]vctp16q_m(uint32_t a, mve_pred16_t p)	a -> Rn p -> Rp	VMSR P0,Rp VPST VCTPT.16 Rn VMRS Rd,P0	Rd -> result	MVE
mve_pred16_t [__arm_]vctp32q_m(uint32_t a, mve_pred16_t p)	a -> Rn p -> Rp	VMSR P0,Rp VPST VCTPT.32 Rn VMRS Rd,P0	Rd -> result	MVE

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
mve_pred16_t [__arm_]vctp64q_m(uint32_t a, mve_pred16_t p)	a -> Rn p -> Rp	VMSR P0,Rp VPST VCPT.64 Rn VMRS Rd,P0	Rd -> result	MVE
int8x16_t [__arm_]vuninitializedq_s8(void)			Qd -> result	MVE
int16x8_t [__arm_]vuninitializedq_s16(void)			Qd -> result	MVE
int32x4_t [__arm_]vuninitializedq_s32(void)			Qd -> result	MVE
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)			Qd -> result	MVE
int16x8_t [__arm_]vreinterpretq_s16_s8(int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [__arm_]vreinterpretq_s32_s8(int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [__arm_]vreinterpretq_f32_s8(int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vreinterpretq_u8_s8(int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vreinterpretq_u16_s8(int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vreinterpretq_u32_s8(int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [__arm_]vreinterpretq_u64_s8(int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [__arm_]vreinterpretq_s64_s8(int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [__arm_]vreinterpretq_f16_s8(int8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [__arm_]vreinterpretq_s8_s16(int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [__arm_]vreinterpretq_s32_s16(int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [__arm_]vreinterpretq_f32_s16(int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vreinterpretq_u8_s16(int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vreinterpretq_u16_s16(int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vreinterpretq_u32_s16(int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [__arm_]vreinterpretq_u64_s16(int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [__arm_]vreinterpretq_s64_s16(int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [__arm_]vreinterpretq_f16_s16(int16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [__arm_]vreinterpretq_s8_s32(int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [__arm_]vreinterpretq_s16_s32(int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [__arm_]vreinterpretq_f32_s32(int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vreinterpretq_u8_s32(int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vreinterpretq_u16_s32(int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vreinterpretq_u32_s32(int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [__arm_]vreinterpretq_u64_s32(int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [__arm_]vreinterpretq_s64_s32(int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [__arm_]vreinterpretq_f16_s32(int32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [__arm_]vreinterpretq_s8_f32(float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [__arm_]vreinterpretq_s16_f32(float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [__arm_]vreinterpretq_s32_f32(float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vreinterpretq_u8_f32(float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vreinterpretq_u16_f32(float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vreinterpretq_u32_f32(float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [__arm_]vreinterpretq_u64_f32(float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [__arm_]vreinterpretq_s64_f32(float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [__arm_]vreinterpretq_f16_f32(float32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [__arm_]vreinterpretq_s8_u8(uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [__arm_]vreinterpretq_s16_u8(uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [__arm_]vreinterpretq_s32_u8(uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [__arm_]vreinterpretq_f32_u8(uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vreinterpretq_u16_u8(uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vreinterpretq_u32_u8(uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [__arm_]vreinterpretq_u64_u8(uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [__arm_]vreinterpretq_s64_u8(uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [__arm_]vreinterpretq_f16_u8(uint8x16_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [__arm_]vreinterpretq_s8_u16(uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [__arm_]vreinterpretq_s16_u16(uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [__arm_]vreinterpretq_s32_u16(uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [__arm_]vreinterpretq_f32_u16(uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vreinterpretq_u8_u16(uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vreinterpretq_u32_u16(uint16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [__arm_]vreinterpretq_u64_u16(uint16x8_t a)	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)	a -> Qd	NOP	Qd -> result	MVE/NEON

Intrinsic	Argument Preparation	Instruction	Result	Supported Architectures
int8x16_t [__arm_]vreinterpretq_s8[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [__arm_]vreinterpretq_s16[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [__arm_]vreinterpretq_s32[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [__arm_]vreinterpretq_f32[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vreinterpretq_u8[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vreinterpretq_u16[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [__arm_]vreinterpretq_u64[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [__arm_]vreinterpretq_s64[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float16x8_t [__arm_]vreinterpretq_f16[_u32](uint32x4_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int8x16_t [__arm_]vreinterpretq_s8[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int16x8_t [__arm_]vreinterpretq_s16[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int32x4_t [__arm_]vreinterpretq_s32[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
float32x4_t [__arm_]vreinterpretq_f32[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vreinterpretq_u8[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vreinterpretq_u16[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vreinterpretq_u32[_u64](uint64x2_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
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)	a -> Qd	NOP	Qd -> result	MVE/NEON
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)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint8x16_t [__arm_]vreinterpretq_u8[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint16x8_t [__arm_]vreinterpretq_u16[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint32x4_t [__arm_]vreinterpretq_u32[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64x2_t [__arm_]vreinterpretq_u64[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
int64x2_t [__arm_]vreinterpretq_s64[_f16](float16x8_t a)	a -> Qd	NOP	Qd -> result	MVE/NEON
uint64_t [__arm_]lsl(uint64_t value, int32_t shift)	value -> [RdaHi,RdaLo] shift -> Rm	LSLL RdaLo,RdaHi,Rm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]asrl(int64_t value, int32_t shift)	value -> [RdaHi,RdaLo] shift -> Rm	ASRL RdaLo,RdaHi,Rm	[RdaHi,RdaLo] -> result	MVE
uint64_t [__arm_]uqrshll(uint64_t value, int32_t shift)	value -> [RdaHi,RdaLo] shift -> Rm	UQRSHLL RdaLo,RdaHi,Rm	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]sqsrshll(int64_t value, int32_t shift)	value -> [RdaHi,RdaLo] shift -> Rm	SQSRSHLL RdaLo,RdaHi,Rm	[RdaHi,RdaLo] -> result	MVE
uint64_t [__arm_]uqshll(uint64_t value, const int shift)	value -> [RdaHi,RdaLo] 1 <= shift <= 32	UQSHLL RdaLo,RdaHi,#shift	[RdaHi,RdaLo] -> result	MVE
uint64_t [__arm_]urshrl(uint64_t value, const int shift)	value -> [RdaHi,RdaLo] 1 <= shift <= 32	URSHRL RdaLo,RdaHi,#shift	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]srshrl(int64_t value, const int shift)	value -> [RdaHi,RdaLo] 1 <= shift <= 32	SRSHRL RdaLo,RdaHi,#shift	[RdaHi,RdaLo] -> result	MVE
int64_t [__arm_]sqshll(int64_t value, const int shift)	value -> [RdaHi,RdaLo] 1 <= shift <= 32	SQSHLL RdaLo,RdaHi,#shift	[RdaHi,RdaLo] -> result	MVE
uint32_t [__arm_]uqrshl(uint32_t value, int32_t shift)	value -> Rda shift -> Rm	UQRSHL Rda,Rm	Rda -> result	MVE

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