

## SQRDMULH (by element)

Signed saturating Rounding Doubling Multiply returning High half (by element). This instruction multiplies each vector element in the first source SIMD&FP register by the specified vector element of the second source SIMD&FP register, doubles the results, places the most significant half of the final results into a vector, and writes the vector to the destination SIMD&FP register.

The results are rounded. For truncated results, see [SQDMULH](#).

If any of the results overflows, they are saturated. If saturation occurs, the cumulative saturation bit [FPSR.QC](#) is set.

Depending on the settings in the [CPACR\\_EL1](#), [CPTR\\_EL2](#), and [CPTR\\_EL3](#) registers, and the current Security state and Exception level, an attempt to execute the instruction might be trapped.

It has encodings from 2 classes: [Scalar](#) and [Vector](#)

### Scalar

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	1	1	1	size	L	M			Rm		1	1	0	1	H	0				Rn					Rd		

op

**SQRDMULH** [<V><d>](#), [<V><n>](#), [<Vm>.<Ts>\[<index>\]](#)

```

constant integer idxdsize = 64 << UInt(H);
integer index;
bit Rmhi;
case size of
    when '01' index = UInt(H:L:M); Rmhi = '0';
    when '10' index = UInt(H:L); Rmhi = M;
    otherwise UNDEFINED;

integer d = UInt(Rd);
integer n = UInt(Rn);
integer m = UInt(Rmhi:Rm);

constant integer esize = 8 << UInt(size);
constant integer datasize = esize;
integer elements = 1;

boolean round = (op == '1');
```

### Vector

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Q	0	0	1	1	1	1	size	L	M			Rm			1	1	0	1	H	0			Rn					Rd		
op																															

**SQRDMULH** <Vd>.<T>, <Vn>.<T>, <Vm>.<Ts> [<index>]

```
constant integer idxdsize = 64 << UInt(H);
integer index;
bit Rmhi;
case size of
    when '01' index = UInt(H:L:M); Rmhi = '0';
    when '10' index = UInt(H:L); Rmhi = M;
    otherwise UNDEFINED;

integer d = UInt(Rd);
integer n = UInt(Rn);
integer m = UInt(Rmhi:Rm);

constant integer esize = 8 << UInt(size);
constant integer datasize = 64 << UInt(Q);
integer elements = datasize DIV esize;

boolean round = (op == '1');
```

## Assembler Symbols

<V>

Is a width specifier, encoded in “size”:

size	<V>
00	RESERVED
01	H
10	S
11	RESERVED

<d>

Is the number of the SIMD&FP destination register, encoded in the "Rd" field.

<n>

Is the number of the first SIMD&FP source register, encoded in the "Rn" field.

<Vd>

Is the name of the SIMD&FP destination register, encoded in the "Rd" field.

<T>

Is an arrangement specifier, encoded in “size:Q”:

size	Q	<T>
00	x	RESERVED
01	0	4H
01	1	8H
10	0	2S
10	1	4S
11	x	RESERVED

<Vn>

Is the name of the first SIMD&FP source register, encoded in the "Rn" field.

<Vm>

Is the name of the second SIMD&FP source register, encoded in “size:M:Rm”:

size	<Vm>
00	RESERVED
01	0:Rm
10	M:Rm
11	RESERVED

Restricted to V0-V15 when element size <Ts> is H.

<Ts>

Is an element size specifier, encoded in “size”:

size	<Ts>
00	RESERVED
01	H
10	S
11	RESERVED

<index>

Is the element index, encoded in “size:L:H:M”:

size	<index>
00	RESERVED
01	H:L:M
10	H:L
11	RESERVED

## Operation

```
CheckFPAdvSIMDEnabled64();
bits(datasize) operand1 = V[n, datasize];
bits(idxdsize) operand2 = V[m, idxdsize];
bits(datasize) result;
integer element1;
integer element2;
integer product;
boolean sat;

element2 = SInt(Elem[operand2, index, esize]);
for e = 0 to elements-1
    element1 = SInt(Elem[operand1, e, esize]);
    product = 2 * element1 * element2;
    product = RShr(product, esize, round);
    // The following only saturates if element1 and element2 equal  $-(2^{esize-1})$ 
    (Elem[result, e, esize], sat) = SignedSatQ(product, esize);
    if sat then FPSR.QC = '1';

V[d, datasize] = result;
```

**Operational information**

If PSTATE.DIT is 1:

- The execution time of this instruction is independent of:
  - The values of the data supplied in any of its registers.
  - The values of the NZCV flags.
- The response of this instruction to asynchronous exceptions does not vary based on:
  - The values of the data supplied in any of its registers.
  - The values of the NZCV flags.

---

<a href="#">Base Instructions</a>	<a href="#">SIMD&amp;FP Instructions</a>	<a href="#">SVE Instructions</a>	<a href="#">SME Instructions</a>	<a href="#">Index by Encoding</a>
-----------------------------------	--	----------------------------------	----------------------------------	-----------------------------------

[Sh](#)  
[Pseu](#)

Internal version only: isa v33.64, AdvSIMD v29.12, pseudocode no\_diffs\_2023\_09\_RC2, sve v2023-06\_rel ; Build timestamp: 2023-09-18T17:56

Copyright Â© 2010-2023 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.