## SQDMULH (multiple and single vector)

Multi-vector signed saturating doubling multiply high by vector

Multiply then double the corresponding signed elements of the two or four first source vectors and the signed elements of the second source vector, and destructively place the most significant half of the result in the corresponding elements of the two or four first source vectors. Each result element is saturated to the N-bit element's signed integer range $-2^{(N-1)}$ to $(2^{(N-1)})-1$.

This instruction is unpredicated.

It has encodings from 2 classes: Two registers and Four registers

### Two registers
**(FEAT_SME2)**

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | size | | 1 | 0 | Zm | | | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Zdn | | | | 0 |

```
SQDMULH { <Zdn1>.<T>-<Zdn2>.<T> }, { <Zdn1>.<T>-<Zdn2>.<T> }, <Zm>.<
```

```
if !HaveSME2() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer dn = UInt(Zdn:'0');
integer m = UInt('0':Zm);
constant integer nreg = 2;
```

### Four registers
**(FEAT_SME2)**

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | size | | 1 | 0 | Zm | | | | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | Zdn | | | 0 | 0 |

```
SQDMULH { <Zdn1>.<T>-<Zdn4>.<T> }, { <Zdn1>.<T>-<Zdn4>.<T> }, <Zm>.<
```

```
if !HaveSME2() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer dn = UInt(Zdn:'00');
integer m = UInt('0':Zm);
constant integer nreg = 4;
```

### Assembler Symbols

| | |
|---|---|
| <Zdn1> | For the two registers variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zdn" times 2. |

For the four registers variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zdn" times 4.

<T>

Is the size specifier, encoded in “size”:

| size | <T> |
|---|---|
| 00 | B |
| 01 | H |
| 10 | S |
| 11 | D |

<Zdn4> Is the name of the fourth scalable vector register of a multi-vector sequence, encoded as "Zdn" times 4 plus 3.

<Zdn2> Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zdn" times 2 plus 1.

<Zm> Is the name of the second source scalable vector register Z0-Z15, encoded in the "Zm" field.

**Operation**

```
CheckStreamingSVEEnabled();
constant integer VL = CurrentVL;
constant integer elements = VL DIV esize;
array [0..3] of bits(VL) results;

for r = 0 to nreg-1
    bits(VL) operand1 = Z[dn+r, VL];
    bits(VL) operand2 = Z[m, VL];
    for e = 0 to elements-1
        integer element1 = SInt(Elem[operand1, e, esize]);
        integer element2 = SInt(Elem[operand2, e, esize]);
        integer res = 2 * element1 * element2;
        Elem[results[r], e, esize] = SignedSat(res >> esize, esize);

for r = 0 to nreg-1
    Z[dn+r, VL] = results[r];
```



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.