SQDMULH (indexed)

Signed saturating doubling multiply high (indexed)

Multiply all signed elements within each 128-bit segment of the first source vector by the specified signed element in the corresponding second source vector segment, double and place the most significant half of the result in the corresponding elements of the destination vector register. Each result element is saturated to the N-bit element's signed integer range $-2^{(N-1)}$ to $(2^{(N-1)})-1$.

The elements within the second source vector are specified using an immediate index which selects the same element position within each 128-bit vector segment. The index range is from 0 to one less than the number of elements per 128-bit segment, encoded in 1 to 3 bits depending on the size of the element.

It has encodings from 3 classes: 16-bit , 32-bit and 64-bit

16-bit

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 0 0 1 0 0	0	i3h	1	i3l	Zm	1 1 1 1 0	0	Zn	Zd
							R		

```
SQDMULH <Zd>.H, <Zn>.H, <Zm>.H[<imm>]
```

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
constant integer esize = 16;
integer index = UInt(i3h:i3l);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Zd);
```

32-bit

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 0 0 1 0 0	1	0	1	i2	Zm	1 1 1 1 0	0	Zn	Zd
	size<1>	size<0>					R		

```
SQDMULH <Zd>.S, <Zn>.S, <Zm>.S[<imm>]
```

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
constant integer esize = 32;
integer index = UInt(i2);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Zd);
```

64-bit

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 0 0 1 0 0	1	1	1	i1	Zm	1 1 1 1 0	0	Zn	Zd
	size<1>	size<0>					R		

```
SQDMULH <Zd>.D, <Zn>.D, <Zm>.D[<imm>]
```

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
constant integer esize = 64;
integer index = UInt(i1);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Zd);
```

Assembler Symbols

<Zd>	Is the name of the destination scalable vector register, encoded in the "Zd" field.
<Zn>	Is the name of the first source scalable vector register, encoded in the "Zn" field.
<Zm>	For the 16-bit and 32-bit variant: is the name of the second source scalable vector register Z0-Z7, encoded in the "Zm" field.
	For the 64-bit variant: is the name of the second source scalable vector register Z0-Z15, encoded in the "Zm" field.
<imm>	For the 16-bit variant: is the element index, in the range 0 to 7, encoded in the "i3h:i3l" fields.
	For the 32-bit variant: is the element index, in the range 0 to 3, encoded in the "i2" field.
	For the 64-bit variant: is the element index, in the range 0 to 1, encoded in the "i1" field.

Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer elements = VL DIV esize;
constant integer eltspersegment = 128 DIV esize;
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[m, VL];
bits(VL) result;

for e = 0 to elements-1
    integer segmentbase = e - (e MOD eltspersegment);
    integer s = segmentbase + index;
    integer element1 = SInt(Elem[operand1, e, esize]);
    integer element2 = SInt(Elem[operand2, s, esize]);
    integer res = 2 * element1 * element2;
    Elem[result, e, esize] = SignedSat(res >> esize, esize);

Z[d, VL] = result;
```

Operational information

If FEAT_SVE2 is implemented or FEAT_SME is implemented, then 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.