SQDMULLB (vectors)

Signed saturating doubling multiply long (bottom)

Multiply the corresponding even-numbered signed elements of the first and second source vectors, double and place the results in the overlapping double-width elements of the destination vector. Each result element is saturated to the double-width N-bit element's signed integer range $-2^{(N-1)}$ to $(2^{(N-1)})-1$. This instruction is unpredicated.

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 1	size	0	Zm	0 1 1	0	0	0	Zn	Zd
						U	T		

SQDMULLB **<Zd>.<T>**, **<Zn>.<Tb>**, **<Zm>.<Tb>**

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
if size == '00' then UNDEFINED;
constant integer esize = 8 << UInt(size);
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.

<T>

Is the size specifier, encoded in "size":

size	<T>
00	RESERVED
01	H
10	S
11	D

<Zn>   Is the name of the first source scalable vector register, encoded in the "Zn" field.

<Tb>

Is the size specifier, encoded in "size":

size	<Tb>
00	RESERVED
01	B
10	H
11	S

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

Operation

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

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

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