

**SQDMLSLBT**

Signed saturating doubling multiply-subtract long from accumulator (bottom  $\bar{A}$ — top)

Multiply then double the corresponding even-numbered signed elements of the first and odd-numbered signed elements of the second source vector. Each intermediate value is saturated to the double-width N-bit value's signed integer range  $-2^{(N-1)}$  to  $(2^{(N-1)})-1$ . Then destructively subtract from the overlapping double-width elements of the addend and destination vector. Each destination 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	0	size	0	Zm					0	0	0	0	1	1	Zn					Zda										
																					S															

**SQDMLSLBT** **<Zda>.<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 da = UInt(Zda);
integer sel1 = 0;
integer sel2 = 1;
```

**Assembler Symbols**

**<Zda>** Is the name of the third source and destination scalable vector register, encoded in the "Zda" 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 PL = VL DIV 8;
constant integer elements = VL DIV esize;
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[m, VL];
bits(VL) result = Z[da, VL];

for e = 0 to elements-1
    integer element1 = SInt(Elem[operand1, 2 * e + sel1, esize DIV 2]);
    integer element2 = SInt(Elem[operand2, 2 * e + sel2, esize DIV 2]);
    integer element3 = SInt(Elem[result, e, esize]);
    integer product = SInt(SignedSat(2 * element1 * element2, esize));
    Elem[result, e, esize] = SignedSat(element3 - product, esize);

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

## Operational information

This instruction might be immediately preceded in program order by a MOVPRFX instruction. The MOVPRFX instruction must conform to all of the following requirements, otherwise the behavior of the MOVPRFX and this instruction is unpredictable:

- The MOVPRFX instruction must be unpredicated.
- The MOVPRFX instruction must specify the same destination register as this instruction.
- The destination register must not refer to architectural register state referenced by any other source operand register of this instruction.

---

[Base  
Instructions](#)

[SIMD&FP  
Instructions](#)

[SVE  
Instructions](#)

[SME  
Instructions](#)

[Index by  
Encoding](#)

[Sh  
Pseud](#)

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.