

## BFMLS (indexed)

BFloat16 floating-point fused multiply-subtract vectors by indexed elements

Multiply all BFloat16 elements within each 128-bit segment of the first source vector by the specified element in the corresponding second source vector segment. The products are then destructively subtracted without intermediate rounding from the corresponding elements of the addend and destination vector.

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 7.

This instruction follows SVE2.1 non-widening BFloat16 numerical behaviors. This instruction is unpredicated.

ID\_AA64ZFR0\_EL1.B16B16 indicates whether this instruction is implemented.

### SVE2

(FEAT\_SVE\_B16B16)

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	1	0	0	1	0	0	0	i3h	1	i3l	Zm	0	0	0	0	1	1	Zn				Zda								
																				op											

**BFMLS** <Zda>.H, <Zn>.H, <Zm>.H[<imm>]

```
if (!HaveSVE2() && !HaveSME2()) || !IsFeatureImplemented(FEAT_SVE_B16B16)
integer index = UInt(i3h:i3l);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer da = UInt(Zda);
boolean op1_neg = TRUE;
boolean op3_neg = FALSE;
```

### Assembler Symbols

<Zda>	Is the name of the third source and destination scalable vector register, encoded in the "Zda" field.
<Zn>	Is the name of the first source scalable vector register, encoded in the "Zn" field.
<Zm>	Is the name of the second source scalable vector register Z0-Z7, encoded in the "Zm" field.
<imm>	Is the immediate index, in the range 0 to 7, encoded in the "i3h:i3l" fields.

## Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV 16;
constant integer eltspersegment = 128 DIV 16;
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 segmentbase = e - (e MOD eltspersegment);
    integer s = segmentbase + index;
    bits(16) element1 = Elem[operand1, e, 16];
    bits(16) element2 = Elem[operand2, s, 16];
    bits(16) element3 = Elem[result, e, 16];
    if op1_neg then element1 = BFNeg(element1);
    if op3_neg then element3 = BFNeg(element3);
    Elem[result, e, 16] = BFMulAdd(element3, element1, element2, FPCR[...])

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.