BFMLSLT (indexed)

BFloat16 floating-point multiply-subtract long from single-precision (top, indexed)

This BFloat16 floating-point multiply-subtract long instruction widens the odd-numbered BFloat16 elements in the first source vector and the indexed element from the corresponding 128-bit segment in the second source vector to single-precision format and then destructively multiplies and subtracts these values without intermediate rounding from the single-precision elements of the destination vector that overlap with the corresponding BFloat16 elements in the first source vector. This instruction is unpredicated.

SVE2
(FEAT_SVE2p1)

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

o2   op   T

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

```
if !HaveSME2() && !HaveSVE2p1() then UNDEFINED;
integer n = UInt(Zn);
integer m = UInt(Zm);
integer da = UInt(Zda);
integer index = UInt(i3h:i3l);
boolean op1_neg = TRUE;
```

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 32;
constant integer eltspersegment = 128 DIV 32;
```

```
    bits(VL) operand1 = Z[n, VL];
    bits(VL) operand2 = Z[m, VL];
    bits(VL) operand3 = Z[da, VL];
    bits(VL) result;

    for e = 0 to elements-1
        integer segmentbase = e - (e MOD eltspersegment);
        integer s = 2 * segmentbase + index;
        bits(16) element1 = Elem[operand1, 2 * e + 1, 16];
        bits(16) element2 = Elem[operand2, s, 16];
        bits(32) element3 = Elem[operand3, e, 32];
        if op1_neg then element1 = BFNeg(element1);
        Elem[result, e, 32] = BFMulAddH(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.