BFMLA (multiple vectors)

Multi-vector BFloat16 floating-point fused multiply-add

Multiply the corresponding BFloat16 floating-point elements of the two or four first and second source vectors and destructively add without intermediate rounding to the corresponding elements of the ZA single-vector groups. The vector numbers forming the single-vector group within each half of or each quarter of the ZA array are selected by the sum of the vector select register and immediate offset, modulo half or quarter the number of ZA array vectors.

The vector group symbol, VGx2 or VGx4, indicates that the ZA operand consists of two or four ZA single-vector groups respectively. The vector group symbol is preferred for disassembly, but optional in assembler source code.

This instruction follows SME2.1 ZA-targeting non-widening BFloat16 numerical behaviors.

This instruction is unpredicated.

ID_AA64SMFR0_EL1.B16B16 indicates whether this instruction is implemented.

It has encodings from 2 classes: Two ZA single-vectors and Four ZA single-vectors

Two ZA single-vectors
(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
1	1	0	0	0	0	0	1	1	1	1	Zm				0	0	Rv		1	0	0	Zn				0	0	1	off3		
									sz																		S				

```
BFMLA  ZA.H[<Wv>, <offs>{, VGx2}], { <Zn1>.H-<Zn2>.H }, { <Zm1>.H-<Zm
```

```
if !HaveSME2() || !IsFeatureImplemented(FEAT_SVE_B16B16) then UNDEFINED
integer v = UInt('010':Rv);
integer n = UInt(Zn:'0');
integer m = UInt(Zm:'0');
integer offset = UInt(off3);
boolean sub_op = FALSE;
constant integer nreg = 2;
```

Four ZA single-vectors
(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
1	1	0	0	0	0	0	1	1	1	1	Zm			0	1	0	Rv		1	0	0	Zn			0	0	0	1	off3		
									sz																		S				

```
BFMLA ZA.H[<Wv>, <offs>{, VGx4}], { <Zn1>.H-<Zn4>.H }, { <Zm1>.H-<Zm
```

```
if !HaveSME2() || !IsFeatureImplemented(FEAT_SVE_B16B16) then UNDEFINED
integer v = UInt('010':Rv);
integer n = UInt(Zn:'00');
integer m = UInt(Zm:'00');
integer offset = UInt(off3);
boolean sub_op = FALSE;
constant integer nreg = 4;
```

Assembler Symbols

<Wv>
Is the 32-bit name of the vector select register W8-W11, encoded in the "Rv" field.

<offs>
Is the vector select offset, in the range 0 to 7, encoded in the "off3" field.

<Zn1>
For the two ZA single-vectors variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zn" times 2.

For the four ZA single-vectors variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zn" times 4.

<Zn4>
Is the name of the fourth scalable vector register of a multi-vector sequence, encoded as "Zn" times 4 plus 3.

<Zn2>
Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zn" times 2 plus 1.

<Zm1>
For the two ZA single-vectors variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zm" times 2.

For the four ZA single-vectors variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zm" times 4.

<Zm4>
Is the name of the fourth scalable vector register of a multi-vector sequence, encoded as "Zm" times 4 plus 3.

<Zm2>
Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zm" times 2 plus 1.

Operation

```
CheckStreamingSVEAndZAEnabled();
constant integer VL = CurrentVL;
constant integer elements = VL DIV 16;
integer vectors = VL DIV 8;
integer vstride = vectors DIV nreg;
bits(32) vbase = X[v, 32];
integer vec = (UInt(vbase) + offset) MOD vstride;
bits(VL) result;
```

```
for r = 0 to nreg-1
    bits(VL) operand1 = Z[n+r, VL];
    bits(VL) operand2 = Z[m+r, VL];
    bits(VL) operand3 = ZAvector[vec, VL];
    for e = 0 to elements-1
        bits(16) element1 = Elem[operand1, e, 16];
        bits(16) element2 = Elem[operand2, e, 16];
        bits(16) element3 = Elem[operand3, e, 16];
        if sub_op then element1 = BFNeg(element1);
        Elem[result, e, 16] = BFMulAdd_ZA(element3, element1, element2,
    ZAvector[vec, VL] = result;
    vec = vec + vstride;
```

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