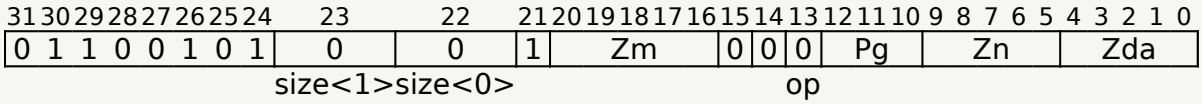
BFMLA (vectors)

BFloat16 floating-point fused multiply-add vectors

Multiply the corresponding active BFloat16 elements of the first and second source vectors and add to elements of the third source (addend) vector without intermediate rounding. Destructively place the results in the destination and third source (addend) vector. Inactive elements in the destination vector register remain unmodified.

This instruction follows SVE2.1 non-widening BFloat16 numerical behaviors. ID_AA64ZFR0_EL1.B16B16 indicates whether this instruction is implemented.

SVE2
(FEAT_SVE_B16B16)



BFMLA <Zda>.H, <Pg>/M, <Zn>.H, <Zm>.H

```
if (!HaveSVE2() && !HaveSME2()) || !IsFeatureImplemented(FEAT_SVE_B16B1
integer g = UInt(Pg);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer da = UInt(Zda);
boolean op1_neg = FALSE;
boolean op3_neg = FALSE;
```

Assembler Symbols

<Zda>	Is the name of the third source and destination scalable vector register, encoded in the "Zda" field.
<Pg>	Is the name of the governing scalable predicate register P0-P7, encoded in the "Pg" 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, encoded in the "Zm" field.

Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV 16;
```

```
bits(PL) mask = P[g, PL];
bits(VL) operand1 = if AnyActiveElement(mask, 16) then Z[n, VL] else Ze
bits(VL) operand2 = if AnyActiveElement(mask, 16) then Z[m, VL] else Ze
bits(VL) operand3 = Z[da, VL];
bits(VL) result;

for e = 0 to elements-1
    if ActivePredicateElement(mask, e, 16) then
        bits(16) element1 = Elem[operand1, e, 16];
        bits(16) element2 = Elem[operand2, e, 16];
        bits(16) element3 = Elem[operand3, e, 16];

        if op1_neg then element1 = BFNeg(element1);
        if op3_neg then element3 = BFNeg(element3);
        Elem[result, e, 16] = BFMulAdd(element3, element1, element2, FP
    else
        Elem[result, e, 16] = Elem[operand3, e, 16];

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, or be predicated using the same governing predicate register and source element size as this instruction.
- 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.

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