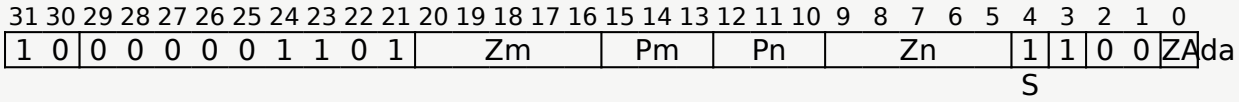
BFMOPS (non-widening)

BFloat16 floating-point outer product and subtract

This instruction works with a 16-bit element ZA tile.
These instructions generate an outer product of the first source vector and the second source vector. The first source is SVL_HÃ—1 vector and the second source is 1Ã—SVL_H vector.

Each source vector is independently predicated by a corresponding governing predicate. When either source vector element is Inactive the corresponding destination tile element remains unmodified.
The resulting outer product, SVL_HÃ—SVL_H, is then destructively subtracted from the destination tile. This is equivalent to performing a single multiply-subtract from each of the destination tile elements.
This instruction follows SME2.1 ZA-targeting non-widening BFloat16 numerical behaviors.
ID_AA64SMFR0_EL1.B16B16 indicates whether this instruction is implemented.

SME2
(FEAT_SVE_B16B16)



BFMOPS <ZAda>.H, <Pn>/M, <Pm>/M, <Zn>.H, <Zm>.H

```
if !HaveSME2() || !IsFeatureImplemented(FEAT_SVE_B16B16) then UNDEFINED
integer a = UInt(Pn);
integer b = UInt(Pm);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer da = UInt(ZAda);
boolean sub_op = TRUE;
```

Assembler Symbols

<ZAda>	Is the name of the ZA tile ZA0-ZA1, encoded in the "ZAda" field.
<Pn>	Is the name of the first governing scalable predicate register P0-P7, encoded in the "Pn" field.
<Pm>	Is the name of the second governing scalable predicate register P0-P7, encoded in the "Pm" 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

```
CheckStreamingSVEAndZAEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer dim = VL DIV 16;
bits(PL) mask1 = P[a, PL];
bits(PL) mask2 = P[b, PL];
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[m, VL];
bits(dim*dim*16) operand3 = ZAtile[da, 16, dim*dim*16];
bits(dim*dim*16) result;

for row = 0 to dim-1
    for col = 0 to dim-1
        bits(16) element1 = Elem[operand1, row, 16];
        bits(16) element2 = Elem[operand2, col, 16];
        bits(16) element3 = Elem[operand3, row*dim+col, 16];

        if ActivePredicateElement(mask1, row, 16) && ActivePredicateEle
            if sub_op then element1 = BFNeg(element1);
            Elem[result, row*dim+col, 16] = BFMulAdd_ZA(element3, eleme
        else
            Elem[result, row*dim+col, 16] = element3;

ZAtile[da, 16, dim*dim*16] = result;
```


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