Base
Instructions

SIMD&FP
Instructions

SVE
Instructions

SME
Instructions

Index by
Encoding

Sh
Pseu

BFDOT (indexed)

BFloat16 floating-point indexed dot product

This instruction delimits the source vectors into pairs of BFloat16 elements. The BFloat16 pairs within the second source vector are specified using an immediate index which selects the same BFloat16 pair position within each 128-bit vector segment. The index range is from 0 to 3.

If FEAT_EBF16 is not implemented or FPCR.EBF is 0, this instruction:

- Performs an unfused sum-of-products of each pair of adjacent BFloat16 elements in the first source vector with the specified pair of elements in the second vector. The intermediate single-precision products are rounded before they are summed, and the intermediate sum is rounded before accumulation into the single-precision destination element that overlaps with the corresponding pair of BFloat16 elements in the first source vector.
- Uses the non-IEEE 754 Round-to-Odd rounding mode, which forces bit 0 of an inexact result to 1, and rounds an overflow to an appropriately signed Infinity.
- Flushes denormalized inputs and results to zero, as if FPCR.{FZ, FIZ} is {1, 1}.
- Disables alternative floating point behaviors, as if FPCR.AH is 0.

If FEAT_EBF16 is implemented and FPCR.EBF is 1, then this instruction:

- Performs a fused sum-of-products of each pair of adjacent BFloat16 elements in the first source vector with the specified pair of elements in the second vector. The intermediate single-precision products are not rounded before they are summed, but the intermediate sum is rounded before accumulation into the single-precision destination element that overlaps with the corresponding pair of BFloat16 elements in the first source vector.
- Follows all other floating-point behaviors that apply to single-precision arithmetic, as governed by FPCR.RMode, FPCR.FZ, FPCR.AH, and FPCR.FIZ.

Irrespective of FEAT_EBF16 and FPCR.EBF, this instruction:

- Does not modify the cumulative FPSR exception bits (IDC, IXC, UFC, OFC, DZC, and IOC).
- Disables trapped floating-point exceptions, as if the FPCR trap enable bits (IDE, IXE, UFE, OFE, DZE, and IOE) are all zero.
- Generates only the default NaN, as if FPCR.DN is 1.

This instruction is unpredicated.
ID_AA64ZFR0_EL1.BF16 indicates whether this instruction is implemented.

SVE
(FEAT_BF16)

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	1	1	i2		Zm			0	1	0	0	0	0		Zn				Zda				

```
        BFDOT <Zda>.S, <Zn>.H, <Zm>.H[<imm>]

    if (!HaveSVE() && !HaveSME()) || !HaveBF16Ext() then UNDEFINED;
    integer n = UInt(Zn);
    integer m = UInt(Zm);
    integer da = UInt(Zda);
    integer index = UInt(i2);
```

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 of a group of two 16-bit elements
            within each 128-bit vector segment, in the range 0 to 3,
            encoded in the "i2" field.

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 = segmentbase + index;
        bits(16) elt1_a = Elem[operand1, 2 * e + 0, 16];
        bits(16) elt1_b = Elem[operand1, 2 * e + 1, 16];
        bits(16) elt2_a = Elem[operand2, 2 * s + 0, 16];
        bits(16) elt2_b = Elem[operand2, 2 * s + 1, 16];
        bits(32) sum = Elem[operand3, e, 32];

        sum = BFDotAdd(sum, elt1_a, elt1_b, elt2_a, elt2_b, FPCR[]);
        Elem[result, e, 32] = sum;

    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.