BFMMLA

BFloat16 floating-point matrix multiply-accumulate into 2Ã—2 matrices

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

- Performs two unfused sums-of-products within each two pairs of adjacent BFloat16 elements while multiplying the 2Ã—4 matrix of BFloat16 values held in each 128-bit segment of the first source vector by the 4Ã—2 matrix of BFloat16 values in the corresponding segment of the second source vector. The intermediate single-precision products are rounded before they are summed and the intermediate sum is rounded before accumulation into the 2Ã—2 single-precision matrix in the corresponding segment of the destination vector. This is equivalent to accumulating two 2-way unfused dot products per destination element.
- 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 two fused sums-of-products within each two pairs of adjacent BFloat16 elements while multiplying the 2Ã—4 matrix of BFloat16 values held in each 128-bit segment of the first source vector by the 4Ã—2 matrix of BFloat16 values in the corresponding segment of the second source vector. The intermediate single-precision products are not rounded before they are summed, but the intermediate sum is rounded before accumulation into the 2Ã—2 single-precision matrix in the corresponding segment of the destination vector. This is equivalent to accumulating two 2-way fused dot products per destination element.
- Follows all other floating-point behaviors that apply to single-precision arithmetic, as governed by FPCR.RMode, FPCR.FZ, FPCR.AH, and FPCR.FIZ.

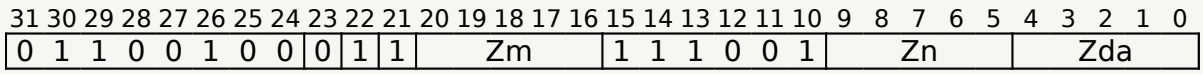
Irrespective of FEAT_EBF16 and FPCR.EBF, this instruction:

- 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 and vector length agnostic.
ID_AA64ZFR0_EL1.BF16 indicates whether this instruction is implemented.

This instruction is illegal when executed in Streaming SVE mode, unless FEAT_SME_FA64 is implemented and enabled.

SVE
(FEAT_BF16)



BFMMLA <Zda>.S, <Zn>.H, <Zm>.H

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

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, encoded in the "Zm" field.

Operation

```
CheckNonStreamingSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer segments =  VL DIV 128;
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[m, VL];
bits(VL) operand3 = Z[da, VL];
bits(VL) result;
bits(128) op1, op2;
bits(128) res, addend;

for s = 0 to segments-1
    op1    = Elem[operand1, s, 128];
    op2    = Elem[operand2, s, 128];
    addend = Elem[operand3, s, 128];
    res    = BFMatMulAdd(addend, op1, op2);
    Elem[result, s, 128] = res;

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.

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