## FMLALT (vectors)

Half-precision floating-point multiply-add long to single-precision (top)

This half-precision floating-point multiply-add long instruction widens the odd-numbered half-precision elements in the first source vector and the corresponding elements in the second source vector to single-precision format and then destructively multiplies and adds these values without intermediate rounding to the single-precision elements of the destination vector that overlap with the corresponding half-precision elements in the source vectors. This instruction is unpredicated.

| 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 | 1 | 0 | 1 | | | Zm | | | 1 | 0 | 0 | 0 | 0 | 1 | | | Zn | | | | | Zda | | |

o2 (bit 23 area), op (bits 12-11), T (bit 10)

    FMLALT **<Zda>**.S, **<Zn>**.H, **<Zm>**.H

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
constant integer esize = 32;
integer n = UInt(Zn);
integer m = UInt(Zm);
integer da = UInt(Zda);
boolean op1_neg = FALSE;
```

### 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

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
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
    bits(esize DIV 2) element1 = Elem[operand1, 2 * e + 1, esize DIV 2]
    bits(esize DIV 2) element2 = Elem[operand2, 2 * e + 1, esize DIV 2]
    bits(esize) element3 = Elem[operand3, e, esize];
    if op1_neg then element1 = FPNeg(element1);
```

```
      Elem[result, e, esize] = FPMulAddH(element3, element1, element2, FP

    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.