FMLS (multiple and single vector)

Multi-vector floating-point fused multiply-subtract by vector

Multiply the corresponding floating-point elements of the two or four first source vector with corresponding elements of the second source vector and destructively subtract without intermediate rounding from the corresponding elements of the ZA single-vector groups. The vector numbers forming the single-vector group within each half of or each quarter of the ZA array are selected by the sum of the vector select register and immediate offset, modulo half or quarter the number of ZA array vectors.

The vector group symbol, VGx2 or VGx4, indicates that the ZA operand consists of two or four ZA single-vector groups respectively. The vector group symbol is preferred for disassembly, but optional in assembler source code.

This instruction follows SME ZA-targeting floating-point behaviors.

This instruction is unpredicated.

ID_AA64SMFR0_EL1.F64F64 indicates whether the double-precision variant is implemented, and ID_AA64SMFR0_EL1.F16F16 indicates whether the half-precision variant is implemented.

It has encodings from 4 classes: Two ZA single-vectors , Two ZA single-vectors of half precision elements , Four ZA single-vectors and Four ZA single-vectors of half precision elements

Two ZA single-vectors

(FEAT_SME2)

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
1	1	0	0	0	0	0	1	0	sz	1	0	Zm				0	Rv		1	1	0	Zn					0	1	off3		
																												S			

```
FMLS ZA.<T>[<Wv>, <offs>{, VGx2}], { <Zn1>.<T>-<Zn2>.<T> }, <Zm>.<T>
```

```
if !HaveSME2() then UNDEFINED;
if sz == '1' && !HaveSMEF64F64() then UNDEFINED;
integer v = UInt('010':Rv);
constant integer esize = 32 << UInt(sz);
integer n = UInt(Zn);
integer m = UInt('0':Zm);
integer offset = UInt(off3);
boolean sub_op = TRUE;
constant integer nreg = 2;
```

Two ZA single-vectors of half precision elements

(FEAT_SME_F16F16)

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
1	1	0	0	0	0	0	1	0	0	1	0	Zm				0	Rv		1	1	1	Zn					0	1	off3		
									sz																			S			

```
FMLS ZA.H[<Wv>, <offs>{, VGx2}], { <Zn1>.H-<Zn2>.H }, <Zm>.H
```

```
if !HaveSME2() || !IsFeatureImplemented(FEAT_SME_F16F16) then UNDEFINED
integer v = UInt('010':Rv);
constant integer esize = 16;
integer n = UInt(Zn);
integer m = UInt('0':Zm);
integer offset = UInt(off3);
boolean sub_op = TRUE;
constant integer nreg = 2;
```

Four ZA single-vectors

(FEAT_SME2)

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
1	1	0	0	0	0	0	1	0	sz	1	1	Zm				0	Rv		1	1	0	Zn					0	1	off3		
																												S			

```
FMLS ZA.<T>[<Wv>, <offs>{, VGx4}], { <Zn1>.<T>-<Zn4>.<T> }, <Zm>.<T>
```

```
if !HaveSME2() then UNDEFINED;
if sz == '1' && !HaveSMEF64F64() then UNDEFINED;
integer v = UInt('010':Rv);
constant integer esize = 32 << UInt(sz);
integer n = UInt(Zn);
integer m = UInt('0':Zm);
integer offset = UInt(off3);
boolean sub_op = TRUE;
constant integer nreg = 4;
```

Four ZA single-vectors of half precision elements

(FEAT_SME_F16F16)

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
1	1	0	0	0	0	0	1	0	0	1	1	Zm				0	Rv		1	1	1	Zn					0	1	off3		
									sz																			S			

```
FMLS ZA.H[<Wv>, <offs>{, VGx4}], { <Zn1>.H-<Zn4>.H }, <Zm>.H
```

```
if !HaveSME2() || !IsFeatureImplemented(FEAT_SME_F16F16) then UNDEFINED
integer v = UInt('010':Rv);
constant integer esize = 16;
integer n = UInt(Zn);
integer m = UInt('0':Zm);
integer offset = UInt(off3);
boolean sub_op = TRUE;
constant integer nreg = 4;
```

Assembler Symbols

<T>

Is the size specifier, encoded in “sz”:

sz	<T>
0	S
1	D

<Wv> Is the 32-bit name of the vector select register W8-W11, encoded in the "Rv" field.

<offs> Is the vector select offset, in the range 0 to 7, encoded in the "off3" field.

<Zn1> Is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zn".

<Zn4> Is the name of the fourth scalable vector register of a multi-vector sequence, encoded as "Zn" plus 3 modulo 32.

<Zn2> Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zn" plus 1 modulo 32.

<Zm> Is the name of the second source scalable vector register Z0-Z15, encoded in the "Zm" field.

Operation

```
CheckStreamingSVEAndZAEnabled();
constant integer VL = CurrentVL;
constant integer elements = VL DIV esize;
integer vectors = VL DIV 8;
integer vstride = vectors DIV nreg;
bits(32) vbase = X[v, 32];
integer vec = (UInt(vbase) + offset) MOD vstride;
bits(VL) result;

for r = 0 to nreg-1
    bits(VL) operand1 = Z[(n+r) MOD 32, VL];
    bits(VL) operand2 = Z[m, VL];
    bits(VL) operand3 = ZAvector[vec, VL];
    for e = 0 to elements-1
        bits(esize) element1 = Elem[operand1, e, esize];
        bits(esize) element2 = Elem[operand2, e, esize];
        bits(esize) element3 = Elem[operand3, e, esize];
        if sub_op then element1 = FPNeg(element1);
        Elem[result, e, esize] = FPMulAdd_ZA(element3, element1, elemen
    ZAvector[vec, VL] = result;
    vec = vec + vstride;
```