FADD

Floating-point add multi-vector to ZA array vector accumulators

Destructively add all elements of the two or four source vectors to 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	1	sz	1	0	0	0	0	0	0	Rv		1	1	1	Zm				0	0	0	off3		
																												S			

```
FADD ZA.<T>[<Wv>, <offs>{, VGx2}], { <Zm1>.<T>-<Zm2>.<T> }
```

```
if !HaveSME2() then UNDEFINED;
if sz == '1' && !HaveSMEF64F64() then UNDEFINED;
integer v = UInt('010':Rv);
constant integer esize = 32 << UInt(sz);
integer m = UInt(Zm:'0');
integer offset = UInt(off3);
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	1	0	1	0	0	1	0	0	0	Rv		1	1	1	Zm				0	0	0	off3		
									sz																			S			

FADD ZA.H[<Wv>, <offs>{, VGx2}], { <Zm1>.H-<Zm2>.H }

```
if !HaveSME2() || !IsFeatureImplemented(FEAT_SME_F16F16) then UNDEFINED
integer v = UInt('010':Rv);
constant integer esize = 16;
integer m = UInt(Zm:'0');
integer offset = UInt(off3);
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	1	sz	1	0	0	0	0	1	0	Rv		1	1	1	Zm			0	0	0	0	off3		
																												S			

FADD ZA.<T>[<Wv>, <offs>{, VGx4}], { <Zm1>.<T>-<Zm4>.<T> }

```
if !HaveSME2() then UNDEFINED;
if sz == '1' && !HaveSMEF64F64() then UNDEFINED;
integer v = UInt('010':Rv);
constant integer esize = 32 << UInt(sz);
integer m = UInt(Zm:'00');
integer offset = UInt(off3);
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	1	0	1	0	0	1	0	1	0	Rv		1	1	1	Zm			0	0	0	0	off3		
									sz																			S			

FADD ZA.H[<Wv>, <offs>{, VGx4}], { <Zm1>.H-<Zm4>.H }

```
if !HaveSME2() || !IsFeatureImplemented(FEAT_SME_F16F16) then UNDEFINED
integer v = UInt('010':Rv);
constant integer esize = 16;
integer m = UInt(Zm:'00');
integer offset = UInt(off3);
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.

<Zm1>

For the two ZA single-vectors and two ZA single-vectors of half precision elements variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zm" times 2.

For the four ZA single-vectors and four ZA single-vectors of half precision elements variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zm" times 4.

<Zm4>

Is the name of the fourth scalable vector register of a multi-vector sequence, encoded as "Zm" times 4 plus 3.

<Zm2>

Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zm" times 2 plus 1.

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 = ZAvector[vec, VL];
    bits(VL) operand2 = Z[m+r, VL];
    for e = 0 to elements-1
        bits(esize) element1 = Elem[operand1, e, esize];
        bits(esize) element2 = Elem[operand2, e, esize];
        Elem[result, e, esize] = FPAdd_ZA(element1, element2, FPCR[]);
    ZAvector[vec, VL] = result;
    vec = vec + vstride;
```

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