FMIN (multiple vectors)

Multi-vector floating-point minimum

Determine the mininum of floating-point elements of the two or four second source vectors and the corresponding floating-point elements of the two or four first source vectors and destructively place the results in the corresponding elements of the two or four first source vectors.
When FPCR.AH is 0, the behavior is as follows:

- Negative zero compares less than positive zero.
- When FPCR.DN is 0, if either element is a NaN, the result is a quiet NaN.
- When FPCR.DN is 1, if either element is a NaN, the result is Default NaN.

When FPCR.AH is 1, the behavior is as follows:

- If both elements are zeros, regardless of the sign of either zero, the result is the second element.
- If either element is a NaN, regardless of the value of FPCR.DN, the result is the second element.

This instruction follows SME2 floating-point numerical behaviors corresponding to instructions that place their results in one or more SVE Z vectors.
This instruction is unpredicated.
It has encodings from 2 classes: Two registers and Four registers

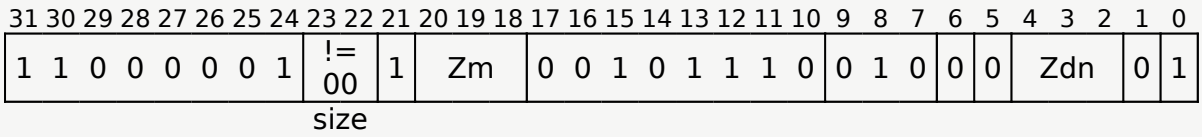
Two registers
(FEAT_SME2)

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	!= 00		1	Zm				0	1	0	1	1	0	0	0	1	0	0	0	Zdn				1
								size																							

FMIN { <Zdn1>.<T>-<Zdn2>.<T> }, { <Zdn1>.<T>-<Zdn2>.<T> }, { <Zm1>.

```
if !HaveSME2() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer dn = UInt(Zdn:'0');
integer m = UInt(Zm:'0');
constant integer nreg = 2;
```

Four registers
(FEAT_SME2)




```
FMIN { <Zdn1>.<T>-<Zdn4>.<T> }, { <Zdn1>.<T>-<Zdn4>.<T> }, { <Zm1>.<
```

```
if !HaveSME2() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer dn = UInt(Zdn:'00');
integer m = UInt(Zm:'00');
constant integer nreg = 4;
```

Assembler Symbols

<Zdn1> For the two registers variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zdn" times 2.

For the four registers variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zdn" times 4.

<T> Is the size specifier, encoded in “size”:

size	<T>
01	H
10	S
11	D

<Zdn4> Is the name of the fourth scalable vector register of a multi-vector sequence, encoded as "Zdn" times 4 plus 3.

<Zdn2> Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zdn" times 2 plus 1.

<Zm1> For the two registers variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zm" times 2.

For the four registers 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

```
CheckStreamingSVEEnabled();
constant integer VL = CurrentVL;
constant integer elements = VL DIV esize;
array [0..3] of bits(VL) results;

for r = 0 to nreg-1
    bits(VL) operand1 = Z[dn+r, 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[results[r], e, esize] = FPMin(element1, element2, FPCR[]);

for r = 0 to nreg-1
    Z[dn+r, VL] = results[r];
```

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