FAC<cc>

Floating-point absolute compare vectors

Compare active absolute values of floating-point elements in the first source vector with corresponding absolute values of elements in the second source vector, and place the boolean results of the specified comparison in the corresponding elements of the destination predicate. Inactive elements in the destination predicate register are set to zero. Does not set the condition flags.

<cc>	**Comparison**
GE	greater than or equal
GT	greater than
LE	less than or equal
LT	less than

This instruction is used by the pseudo-instructions FACLE, and FACLT.

It has encodings from 2 classes: Greater than and Greater than or equal

Greater than

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	1	size	0	Zm	1	1	1	Pg	Zn	1	Pd

FACGT <Pd>.<T>, <Pg>/Z, <Zn>.<T>, <Zm>.<T>

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
if size == '00' then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer g = UInt(Pg);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Pd);
SVECmp op = Cmp_GT;
```

Greater than or equal

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	1	size	0	Zm	1	1	0	Pg	Zn	1	Pd

FACGE <Pd>.<T>, <Pg>/Z, <Zn>.<T>, <Zm>.<T>

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
if size == '00' then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer g = UInt(Pg);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Pd);
SVECmp op = Cmp_GE;
```

Assembler Symbols

<Pd>	Is the name of the destination scalable predicate register, encoded in the "Pd" field.
<T>	Is the size specifier, encoded in “size”:

size	<T>
00	RESERVED
01	H
10	S
11	D

<Pg>	Is the name of the governing scalable predicate register P0-P7, encoded in the "Pg" 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(PL) mask = P[g, PL];
bits(VL) operand1 = if AnyActiveElement(mask, esize) then Z[n, VL] else
bits(VL) operand2 = if AnyActiveElement(mask, esize) then Z[m, VL] else
bits(PL) result;
constant integer psize = esize DIV 8;

for e = 0 to elements-1
    if ActivePredicateElement(mask, e, esize) then
        bits(esize) element1 = Elem[operand1, e, esize];
        bits(esize) element2 = Elem[operand2, e, esize];
        boolean res;
        case op of
            when Cmp_GE res = FPCompareGE(FPAbs(element1), FPAbs(elemen
            when Cmp_GT res = FPCompareGT(FPAbs(element1), FPAbs(elemen
        bit pbit = if res then '1' else '0';
        Elem[result, e, psize] = ZeroExtend(pbit, psize);
    else
        Elem[result, e, psize] = ZeroExtend('0', psize);

P[d, PL] = result;
```

Operational information

If FEAT_SME is implemented and the PE is in Streaming SVE mode, then any subsequent instruction which is dependent on the predicate register written by this instruction might be significantly delayed.

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