<u>Base</u>	SIMD&FP	<u>SVE</u>	<u>SME</u>	<u>Index by</u>
<u>Instructions</u>	<u>Instructions</u>	<u>Instructions</u>	<u>Instructions</u>	Encoding

Pseu

FMIN (vector)

Floating-point minimum (vector). This instruction compares corresponding elements in the vectors in the two source SIMD&FP registers, places the smaller of each of the two floating-point values into a vector, and writes the vector to the destination SIMD&FP register.

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 can generate a floating-point exception. Depending on the settings in *FPCR*, the exception results in either a flag being set in *FPSR*, or a synchronous exception being generated. For more information, see *Floating-point exception traps*.

Depending on the settings in the *CPACR_EL1*, *CPTR_EL2*, and *CPTR_EL3* registers, and the current Security state and Exception level, an attempt to execute the instruction might be trapped.

It has encodings from 2 classes: <u>Half-precision</u> and <u>Single-precision</u> and <u>double-precision</u>

Half-precision (FEAT_FP16)

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 Q 0 0 1	1 1 0 1 1 0	Rm 0 0	1 1 0 1	Rn	Rd
U	01				•

FMIN <Vd>.<T>, <Vn>.<T>, <Vm>.<T>

```
if !IsFeatureImplemented(FEAT_FP16) then UNDEFINED;
integer d = UInt(Rd);
integer n = UInt(Rn);
integer m = UInt(Rm);
constant integer esize = 16;
constant integer datasize = 64 << UInt(Q);
integer elements = datasize DIV esize;

boolean pair = (U == '1');
boolean minimum = (o1 == '1');</pre>
```

Single-precision and double-precision

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 Q 0 0 1	1 1 0 1 sz 1	Rm	1 1	1 1	0 1		Rn				Rd	
U	01											

```
FMIN <Vd>. <T>, <Vn>. <T>, <Vm>. <T>
```

```
integer d = UInt(Rd);
integer n = UInt(Rn);
integer m = UInt(Rm);
if sz:Q == '10' then UNDEFINED;
constant integer esize = 32 << UInt(sz);
constant integer datasize = 64 << UInt(Q);
integer elements = datasize DIV esize;

boolean pair = (U == '1');
boolean minimum = (o1 == '1');</pre>
```

Assembler Symbols

<Vd>

Is the name of the SIMD&FP destination register, encoded in the "Rd" field.

<T>

For the half-precision variant: is an arrangement specifier, encoded in "Q":

Q	<t></t>
0	4 H
1	8H

For the single-precision and double-precision variant: is an arrangement specifier, encoded in "sz:Q":

SZ	Q	<t></t>
0	0	2S
0	1	4S
1	0	RESERVED
1	1	2D

<Vn>

Is the name of the first SIMD&FP source register, encoded in the "Rn" field.

<Vm>

Is the name of the second SIMD&FP source register, encoded in the "Rm" field.

Operation

```
CheckFPAdvSIMDEnabled64();
bits(datasize) operand1 = V[n, datasize];
bits(datasize) operand2 = V[m, datasize];
bits(datasize) result;
bits(2*datasize) concat = operand2:operand1;
bits(esize) element1;
bits(esize) element2;
for e = 0 to elements-1
    if pair then
        element1 = <u>Elem</u>[concat, 2*e, esize];
        element2 = Elem[concat, (2*e)+1, esize];
        element1 = <u>Elem[operand1, e, esize];</u>
        element2 = <u>Elem</u>[operand2, e, esize];
    if minimum then
        Elem[result, e, esize] = FPMin(element1, element2, FPCR[]);
    else
        Elem[result, e, esize] = FPMax(element1, element2, FPCR[]);
V[d, datasize] = result;
```

BaseSIMD&FPSVESMEIndex byInstructionsInstructionsInstructionsInstructions

 $Internal\ version\ only: is a\ v33.64,\ AdvSIMD\ v29.12,\ pseudocode\ no_diffs_2023_09_RC2,\ sve\ v2023-06_rel\ ;\ Build\ timestamp:\ 2023-09-18T17:56$

Sh

Pseu

Copyright \hat{A} © 2010-2023 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.