

FMAXV

Floating-point Maximum across Vector. This instruction compares all the vector elements in the source SIMD&FP register, and writes the largest of the values as a scalar to the destination SIMD&FP register. All the values in this instruction are floating-point values.

When [FPCR.AH](#) is 0, the behavior is as follows:

- Negative zero compares less than positive zero.
- When [FPCR.DN](#) is 0, if either value is a NaN, the result is a quiet NaN.
- When [FPCR.DN](#) is 1, if either value is a NaN, the result is Default NaN.

When [FPCR.AH](#) is 1, the behavior is as follows:

- If both values are zeros, regardless of the sign of either zero, the result is the second value.
- If either value is a NaN, regardless of the value of [FPCR.DN](#), the result is the second value.

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: [Half-precision](#) and [Single-precision and double-precision](#)

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	0	0	1	1	0	0	0	0	1	1	1	1	1	0	Rn			Rd						
o1																															

o1

FMAXV [<V>](#)[<d>](#), [<Vn>](#).[<T>](#)

```
if !IsFeatureImplemented(FEAT_FP16) then UNDEFINED;
```

```
integer d = UInt(Rd);
integer n = UInt(Rn);
```

```
constant integer esize = 16;
constant integer datasize = 64 << UInt(Q);
```

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	1	0	1	1	1	0	0	sz	1	1	0	0	0	0	1	1	1	1	1	0	Rn					Rd				
01																															

FMAXV <V><d>, <Vn>.<T>

```
integer d = UInt(Rd);
integer n = UInt(Rn);

if sz:Q != '01' then UNDEFINED;

constant integer esize = 32 << UInt(sz);
constant integer datasize = 64 << UInt(Q);
```

Assembler Symbols

<V> For the half-precision variant: is the destination width specifier, H.

For the single-precision and double-precision variant: is the destination width specifier, encoded in "sz":

sz	<V>
0	S
1	RESERVED

<d> Is the number of the SIMD&FP destination register, encoded in the "Rd" field.

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

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

Q	<T>
0	4H
1	8H

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

Q	sz	<T>
0	x	RESERVED
1	0	4S
1	1	RESERVED

Operation

```
CheckFPAdvSIMDEnabled64();
bits(datasize) operand = V[n, datasize];

V[d, esize] = Reduce(ReduceOp_FMAX, operand, esize);
```

[Base
Instructions](#)

[SIMD&FP
Instructions](#)

[SVE
Instructions](#)

[SME
Instructions](#)

[Index by
Encoding](#)

[Sh
Pseu](#)

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
Copyright Â© 2010-2023 Arm Limited or its affiliates. All rights reserved. This
document is Non-Confidential.