## FCMPE

Floating-point signaling Compare (scalar). This instruction compares the two SIMD&FP source register values, or the first SIMD&FP source register value and zero. It writes the result to the *PSTATE*.{N, Z, C, V} flags.

This instruction raises an Invalid Operation floating-point exception if either or both of the operands is any type of NaN.

A floating-point exception can be generated by this instruction. 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.

| 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 | 0 | 0 | 1 1 1 1 0 | ftype | 1 | Rm | 0 0 | 1 0 0 0 | Rn | 1 x | 0 0 0 |
| | | | | | | | | | | opc | |

**Half-precision (ftype == 11 && opc == 10)**
**(FEAT_FP16)**

```
FCMPE <Hn>, <Hm>
```

**Half-precision, zero (ftype == 11 && Rm == (00000) && opc == 11)**
**(FEAT_FP16)**

```
FCMPE <Hn>, #0.0
```

**Single-precision (ftype == 00 && opc == 10)**

```
FCMPE <Sn>, <Sm>
```

**Single-precision, zero (ftype == 00 && Rm == (00000) && opc == 11)**

```
FCMPE <Sn>, #0.0
```

**Double-precision (ftype == 01 && opc == 10)**

```
FCMPE <Dn>, <Dm>
```

**Double-precision, zero (ftype == 01 && Rm == (00000) && opc == 11)**

```
FCMPE <Dn>, #0.0
```

```
if ftype == '10' || (ftype == '11' && !IsFeatureImplemented(FEAT_FP16))

integer n = UInt(Rn);
integer m = UInt(Rm);     // ignored when opc<0> == '1'

constant integer datasize = 8 << UInt(ftype EOR '10');
boolean signal_all_nans = (opc<1> == '1');
boolean cmp_with_zero = (opc<0> == '1');
```

**Assembler Symbols**

| | |
|---|---|
| <Dn> | For the double-precision variant: is the 64-bit name of the first SIMD&FP source register, encoded in the "Rn" field.<br><br>For the double-precision, zero variant: is the 64-bit name of the SIMD&FP source register, encoded in the "Rn" field. |
| <Dm> | Is the 64-bit name of the second SIMD&FP source register, encoded in the "Rm" field. |
| <Hn> | For the half-precision variant: is the 16-bit name of the first SIMD&FP source register, encoded in the "Rn" field.<br><br>For the half-precision, zero variant: is the 16-bit name of the SIMD&FP source register, encoded in the "Rn" field. |
| <Hm> | Is the 16-bit name of the second SIMD&FP source register, encoded in the "Rm" field. |
| <Sn> | For the single-precision variant: is the 32-bit name of the first SIMD&FP source register, encoded in the "Rn" field.<br><br>For the single-precision, zero variant: is the 32-bit name of the SIMD&FP source register, encoded in the "Rn" field. |
| <Sm> | Is the 32-bit name of the second SIMD&FP source register, encoded in the "Rm" field. |

**Operation**

```
CheckFPEnabled64();

bits(datasize) operand1 = V[n, datasize];
bits(datasize) operand2;

operand2 = if cmp_with_zero then FPZero('0', datasize) else V[m, datasi

PSTATE.<N,Z,C,V> = FPCompare(operand1, operand2, signal_all_nans, FPCR[
```

**Operational information**

The IEEE 754 standard specifies that the result of a comparison is precisely one of <, ==, > or unordered. If either or both of the operands is a NaN, they are unordered, and all three of (Operand1 < Operand2), (Operand1 == Operand2) and (Operand1 > Operand2) are false. An unordered comparison sets the *PSTATE* condition flags to N=0, Z=0, C=1, and V=1.

If FEAT_SME is implemented and the PE is in Streaming SVE mode, then any subsequent instruction which is dependent on the NZCV condition flags 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