

FDIVR

Floating-point reversed divide by vector (predicated)

Reversed divide active floating-point elements of the second source vector by corresponding floating-point elements of the first source vector and destructively place the quotient in the corresponding elements of the first source vector. Inactive elements in the destination vector register remain unmodified.

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	0	1	1	0	0	1	0	0	Pg	Zm				Zdn								

FDIVR <Zdn>.<T>, <Pg>/M, <Zdn>.<T>, <Zm>.<T>

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
if size == '00' then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer g = UInt(Pg);
integer dn = UInt(Zdn);
integer m = UInt(Zm);
```

Assembler Symbols

<Zdn> Is the name of the first source and destination scalable vector register, encoded in the "Zdn" 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.

<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 = Z[dn, VL];
bits(VL) operand2 = if AnyActiveElement(mask, esize) then Z[m, VL] else
```

```

bits(VL) result;

for e = 0 to elements-1
    bits(esize) element1 = Elem[operand1, e, esize];
    if ActivePredicateElement(mask, e, esize) then
        bits(esize) element2 = Elem[operand2, e, esize];
        Elem[result, e, esize] = FPDiv(element2, element1, FPCR[]);
    else
        Elem[result, e, esize] = element1;

Z[dn, VL] = result;

```

Operational information

This instruction might be immediately preceded in program order by a MOVPRFX instruction. The MOVPRFX instruction must conform to all of the following requirements, otherwise the behavior of the MOVPRFX and this instruction is unpredictable:

- The MOVPRFX instruction must be unpredicated, or be predicated using the same governing predicate register and source element size as this instruction.
- The MOVPRFX instruction must specify the same destination register as this instruction.
- The destination register must not refer to architectural register state referenced by any other source operand register of this instruction.

[Base
Instructions](#)

[SIMD&FP
Instructions](#)

[SVE
Instructions](#)

[SME
Instructions](#)

[Index by
Encoding](#)

[Sh
Pseud](#)

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