

## ORN (predicates)

Bitwise inclusive OR inverted predicate

Bitwise inclusive OR inverted active elements of the second source predicate with corresponding elements of the first source predicate and place the results 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.

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	1	0	0	1	0	1	1	0	0	0		Pm			0	1		Pg			0		Pn		1			Pd		
S																															

**ORN** **<Pd>.B, <Pg>/Z, <Pn>.B, <Pm>.B**

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
constant integer esize = 8;
integer g = UInt(Pg);
integer n = UInt(Pn);
integer m = UInt(Pm);
integer d = UInt(Pd);
boolean setflags = FALSE;
```

## Assembler Symbols

- <Pd>** Is the name of the destination scalable predicate register, encoded in the "Pd" field.
- <Pg>** Is the name of the governing scalable predicate register, encoded in the "Pg" field.
- <Pn>** Is the name of the first source scalable predicate register, encoded in the "Pn" field.
- <Pm>** Is the name of the second source scalable predicate register, encoded in the "Pm" 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(PL) operand1 = P[n, PL];
bits(PL) operand2 = P[m, PL];
bits(PL) result;
constant integer psize = esize DIV 8;

for e = 0 to elements-1
    bit element1 = PredicateElement(operand1, e, esize);
```

```

bit element2 = PredicateElement(operand2, e, esize);
if ActivePredicateElement(mask, e, esize) then
    Elem[result, e, psize] = ZeroExtend(element1 OR (NOT element2),
else
    Elem[result, e, psize] = ZeroExtend('0', psize);

if setflags then
    PSTATE.<N,Z,C,V> = PredTest(mask, result, esize);
P[d, PL] = result;

```

## Operational information

If FEAT\_SVE2 is implemented or FEAT\_SME is implemented, then if PSTATE.DIT is 1:

- The execution time of this instruction is independent of:
  - The values of the data supplied in any of its operand registers when its governing predicate register contains the same value for each execution.
  - The values of the NZCV flags.
- The response of this instruction to asynchronous exceptions does not vary based on:
  - The values of the data supplied in any of its operand registers when its governing predicate register contains the same value for each execution.
  - The values of the NZCV flags.

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