

## CLASTA (vectors)

Conditionally extract element after last to vector register

From the second source vector register extract the element after the last active element, or if the last active element is the final element extract element zero, and then replicate that element to destructively fill the destination and first source vector.

If there are no active elements then leave the destination and source vector 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	0	0	0	0	1	0	1	size	1	0	1	0	0	0	1	0	0	Pg							Zm					Zdn	
B																															

**CLASTA** <Zdn>.<T>, <Pg>, <Zdn>.<T>, <Zm>.<T>

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

## 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	B
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 = Z[m, VL];
bits(VL) result;
integer last = LastActiveElement(mask, esize);

if last < 0 then
    result = operand1;
else
    if !isBefore then
        last = last + 1;
        if last >= elements then last = 0;
    for e = 0 to elements-1
        Elem[result, e, esize] = Elem[operand2, last, esize];

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.
- 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.