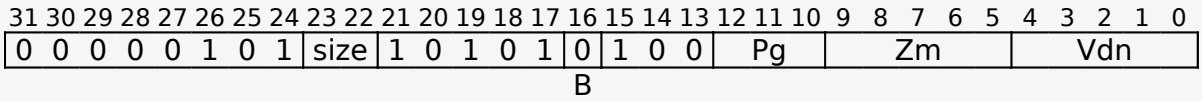
CLASTA (SIMD&FP scalar)

Conditionally extract element after last to SIMD&FP scalar register

From the 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 zero-extend that element to destructively place in the destination and first source SIMD & floating-point scalar register. If there are no active elements then destructively zero-extend the least significant element-size bits of the destination and first source SIMD & floating-point scalar register.



CLASTA <V><dn>, <Pg>, <V><dn>, <Zm>.<T>

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

Assembler Symbols

<V>

Is a width specifier, encoded in “size”:

size	<V>
00	B
01	H
10	S
11	D

<dn>

Is the number [0-31] of the source and destination SIMD&FP register, encoded in the "Vdn" field.

<Pg>

Is the name of the governing scalable predicate register P0-P7, encoded in the "Pg" field.

<Zm>

Is the name of the source scalable vector register, encoded in the "Zm" field.

<T>

Is the size specifier, encoded in “size”:

size	<T>
00	B
01	H
10	S
11	D

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(esize) operand1 = V[dn, esize];
bits(VL) operand2 = Z[m, VL];
bits(esize) result;
integer last = LastActiveElement(mask, esize);

if last < 0 then
    result = ZeroExtend(operand1, esize);
else
    if !isBefore then
        last = last + 1;
        if last >= elements then last = 0;
    result = Elem[operand2, last, esize];

V[dn, esize] = result;
```


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