LASTB (scalar)

Extract last element to general-purpose register

If there is an active element then extract the last active element from the final source vector register. If there are no active elements, extract the highest-numbered element. Then zero-extend and place the extracted element in the destination general-purpose register.

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 0 0 0	1 1 0 1	Pg	Zn	Rd

B

LASTB **\<R\>\<d\>**, **\<Pg\>**, **\<Zn\>.\<T\>**

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
constant integer esize = 8 << UInt(size);
constant integer rsize = if esize < 64 then 32 else 64;
integer g = UInt(Pg);
integer n = UInt(Zn);
integer d = UInt(Rd);
boolean isBefore = TRUE;
```

Assembler Symbols

\<R\>

Is a width specifier, encoded in "size":

size	\<R\>
01	W
x0	W
11	X

\<d\>   Is the number [0-30] of the destination general-purpose register or the name ZR (31), encoded in the "Rd" field.

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

\<Zn\>   Is the name of the source scalable vector register, encoded in the "Zn" 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(VL) operand = Z[n, VL];
bits(rsize) result;
integer last = LastActiveElement(mask, esize);

if isBefore then
    if last < 0 then last = elements – 1;
else
    last = last + 1;
    if last >= elements then last = 0;
result = ZeroExtend(Elem[operand, last, esize], rsize);

X[d, rsize] = result;
```

Operational information

If FEAT_SME is implemented and the PE is in Streaming SVE mode, then any subsequent instruction which is dependent on the general-purpose register written by this instruction might be significantly delayed.