Base
Instructions

SIMD&FP
Instructions

SVE
Instructions

SME
Instructions

Index by
Encoding

Sh
Pseud

INDEX (scalar, immediate)

Create index starting from general-purpose register and incremented by immediate

Populates the destination vector by setting the first element to the first signed scalar integer operand and monotonically incrementing the value by the second signed immediate integer operand for each subsequent element. The scalar source operand is a general-purpose register in which only the least significant bits corresponding to the vector element size are used and any remaining bits are ignored. This instruction is unpredicated.

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	0	size	1	imm5	0	1	0	0	0	1	Rn	Zd

INDEX **<Zd>.<T>**, **<R><n>**, **#<imm>**

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer n = UInt(Rn);
integer d = UInt(Zd);
integer imm = SInt(imm5);
```

Assembler Symbols

<Zd>   Is the name of the destination scalable vector register, encoded in the "Zd" field.

<T>
          Is the size specifier, encoded in "size":

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

<R>
          Is a width specifier, encoded in "size":

size	<R>
01	W
x0	W
11	X

<n>   Is the number [0-30] of the source general-purpose register or the name ZR (31), encoded in the "Rn" field.

<imm>   Is the signed immediate operand, in the range -16 to 15, encoded in the "imm5" field.

Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
bits(esize) operand1 = X[n, esize];
integer element1 = SInt(operand1);
bits(VL) result;

for e = 0 to elements-1
    integer index = element1 + e * imm;
    Elem[result, e, esize] = index<esize-1:0>;

Z[d, VL] = 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 registers.
  ◦ 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 registers.
  ◦ The values of the NZCV flags.