TBL

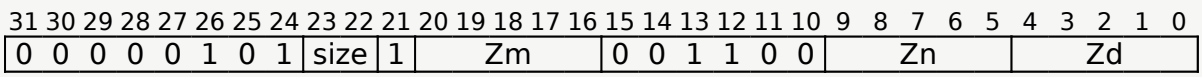
Programmable table lookup in one or two vector table (zeroing)

Reads each element of the second source (index) vector and uses its value to select an indexed element from a table of elements consisting of one or two consecutive vector registers, where the first or only vector holds the lower numbered elements, and places the indexed table element in the destination vector element corresponding to the index vector element. If an index value is greater than or equal to the number of vector elements then it places zero in the corresponding destination vector element.

Since the index values can select any element in a vector this operation is not naturally vector length agnostic.

It has encodings from 2 classes: SVE and SVE2

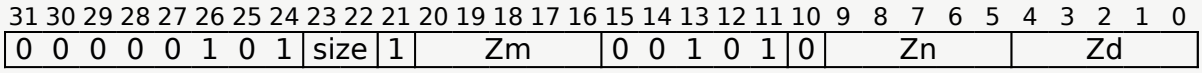
SVE



TBL <Zd>.<T>, { <Zn>.<T> }, <Zm>.<T>

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Zd);
boolean double_table = FALSE;
```

SVE2



TBL <Zd>.<T>, { <Zn1>.<T>, <Zn2>.<T> }, <Zm>.<T>

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Zd);
boolean double_table = TRUE;
```

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

<Zn> Is the name of the first source scalable vector register, encoded in the "Zn" field.

<Zn1> Is the name of the first scalable vector register of a multi-vector sequence, encoded in the "Zn" field.

<Zn2> Is the name of the second scalable vector register of a multi-vector sequence, encoded in the "Zn" 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(VL) indexes = Z[m, VL];
bits(VL) result;
constant integer table_size = if double_table then VL*2 else VL;
constant integer table_elems = table_size DIV esize;
bits(table_size) table;

if double_table then
    bits(VL) top = Z[(n + 1) MOD 32, VL];
    bits(VL) bottom = Z[n, VL];
    table = (top:bottom)<table_size-1:0>;
else
    table = Z[n, table_size];

for e = 0 to elements-1
    integer idx = UInt(Elem[indexes, e, esize]);
    Elem[result, e, esize] = if idx < table_elems then Elem[table, idx,

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.

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