ZIP (four registers)

Interleave elements from four vectors

Place the four-way interleaved elements from the four source vectors in the corresponding elements of the four destination vectors.
This instruction is unpredicated.
It has encodings from 2 classes: 8-bit to 64-bit elements and 128-bit element

8-bit to 64-bit elements
(FEAT_SME2)

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
1	1	0	0	0	0	0	1	size		1	1	0	1	1	0	1	1	1	0	0	0	Zn			0	0	Zd			0	0

```
ZIP { <Zd1>.<T>-<Zd4>.<T> }, { <Zn1>.<T>-<Zn4>.<T> }
```

```
if !HaveSME2() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer n = UInt(Zn:'00');
integer d = UInt(Zd:'00');
```

128-bit element
(FEAT_SME2)

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
1	1	0	0	0	0	0	1	0	0	1	1	0	1	1	1	1	1	1	0	0	0	Zn			0	0	Zd			0	0

```
ZIP { <Zd1>.Q-<Zd4>.Q }, { <Zn1>.Q-<Zn4>.Q }
```

```
if !HaveSME2() then UNDEFINED;
constant integer esize = 128;
integer n = UInt(Zn:'00');
integer d = UInt(Zd:'00');
```

Assembler Symbols

<Zd1> Is the name of the first destination scalable vector register of a multi-vector sequence, encoded as "Zd" times 4.

<T> Is the size specifier, encoded in “size”:

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

<Zd4>	Is the name of the fourth destination scalable vector register of a multi-vector sequence, encoded as "Zd" times 4 plus 3.
<Zn1>	Is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zn" times 4.
<Zn4>	Is the name of the fourth scalable vector register of a multi-vector sequence, encoded as "Zn" times 4 plus 3.

Operation

```
CheckStreamingSVEEnabled();
constant integer VL = CurrentVL;
if VL < esize * 4 then UNDEFINED;
constant integer quads = VL DIV (esize * 4);
bits(VL) operand0 = Z[n, VL];
bits(VL) operand1 = Z[n+1, VL];
bits(VL) operand2 = Z[n+2, VL];
bits(VL) operand3 = Z[n+3, VL];
bits(VL) result;

for r = 0 to 3
    integer base = r * quads;
    for q = 0 to quads-1
        Elem[result, 4*q+0, esize] = Elem[operand0, base+q, esize];
        Elem[result, 4*q+1, esize] = Elem[operand1, base+q, esize];
        Elem[result, 4*q+2, esize] = Elem[operand2, base+q, esize];
        Elem[result, 4*q+3, esize] = Elem[operand3, base+q, esize];
    Z[d+r, VL] = result;
```

Operational information

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