MOVA (tile to vector, four registers)

Move four ZA tile slices to four vector registers

The instruction operates on four consecutive horizontal or vertical slices within a named ZA tile of the specified element size.

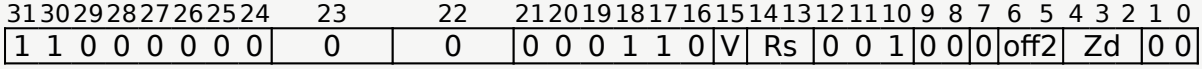
The consecutive slice numbers within the tile are selected starting from the sum of the slice index register and immediate offset, modulo the number of such elements in a vector. The immediate offset is a multiple of 4 in the range 0 to the number of elements in a 128-bit vector segment minus 4.

This instruction is unpredicated.

This instruction is used by the alias MOV (tile to vector, four registers).

It has encodings from 4 classes: 8-bit , 16-bit , 32-bit and 64-bit

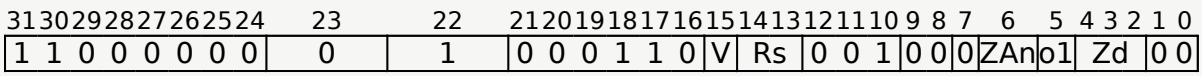
8-bit
(FEAT_SME2)



MOVA { <Zd1>.B-<Zd4>.B }, ZA0<HV>.B[<Ws>, <offs1>:<offs4>]

```
if !HaveSME2() then UNDEFINED;
integer s = UInt('011':Rs);
constant integer nreg = 4;
constant integer esize = 8;
integer d = UInt(Zd:'00');
integer n = 0;
integer offset = UInt(off2:'00');
boolean vertical = V == '1';
```

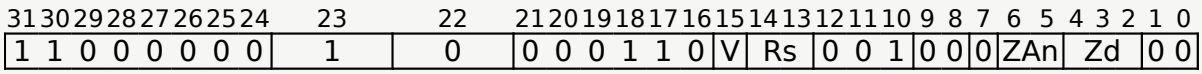
16-bit
(FEAT_SME2)



MOVA { <Zd1>.H-<Zd4>.H }, <ZAn><HV>.H[<Ws>, <offs1>:<offs4>]

```
if !HaveSME2() then UNDEFINED;
integer s = UInt('011':Rs);
constant integer nreg = 4;
constant integer esize = 16;
integer d = UInt(Zd:'00');
integer n = UInt(ZAn);
integer offset = UInt(o1:'00');
boolean vertical = V == '1';
```

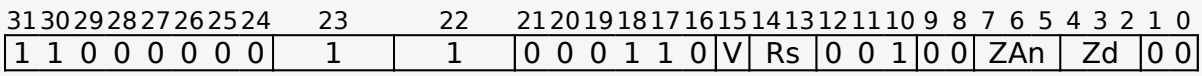
32-bit
(FEAT_SME2)



MOVA { <Zd1>.S-<Zd4>.S }, <ZAn><HV>.S[<Ws>, <offs1>:<offs4>]

```
if !HaveSME2() then UNDEFINED;
integer s = UInt('011':Rs);
constant integer nreg = 4;
constant integer esize = 32;
integer d = UInt(Zd:'00');
integer n = UInt(ZAn);
integer offset = 0;
boolean vertical = V == '1';
```

64-bit
(FEAT_SME2)



MOVA { <Zd1>.D-<Zd4>.D }, <ZAn><HV>.D[<Ws>, <offs1>:<offs4>]

```
if !HaveSME2() then UNDEFINED;
integer s = UInt('011':Rs);
constant integer nreg = 4;
constant integer esize = 64;
integer d = UInt(Zd:'00');
integer n = UInt(ZAn);
integer offset = 0;
boolean vertical = V == '1';
```

Assembler Symbols

<Zd1>	Is the name of the first destination scalable vector register of a multi-vector sequence, encoded as "Zd" times 4.
<Zd4>	Is the name of the fourth destination scalable vector register of a multi-vector sequence, encoded as "Zd" times 4 plus 3.
<ZAn>	For the 16-bit variant: is the name of the ZA tile ZA0-ZA1 to be accessed, encoded in the "ZAn" field. For the 32-bit variant: is the name of the ZA tile ZA0-ZA3 to be accessed, encoded in the "ZAn" field. For the 64-bit variant: is the name of the ZA tile ZA0-ZA7 to be accessed, encoded in the "ZAn" field.

<HV>

Is the horizontal or vertical slice indicator, encoded in “V”:

V	<HV>
0	H
1	V

<Ws>

Is the 32-bit name of the slice index register W12-W15, encoded in the "Rs" field.

<offs1>

For the 8-bit variant: is the slice index offset, pointing to first of four consecutive slices, encoded as "off2" field times 4.

For the 16-bit variant: is the slice index offset, pointing to first of four consecutive slices, encoded as "o1" field times 4.

For the 32-bit and 64-bit variant: is the slice index offset, pointing to first of four consecutive slices, with implicit value 0.

<offs4>

For the 8-bit variant: is the slice index offset, pointing to last of four consecutive slices, encoded as "off2" field times 4 plus 3.

For the 16-bit variant: is the slice index offset, pointing to last of four consecutive slices, encoded as "o1" field times 4 plus 3.

For the 32-bit and 64-bit variant: is the slice index offset, pointing to last of four consecutive slices, with implicit value 3.

Operation

```
CheckStreamingSVEAndZAEnabled();
constant integer VL = CurrentVL;
if nreg == 4 && esize == 64 && VL == 128 then UNDEFINED;
integer slices = VL DIV esize;
bits(32) index = X[s, 32];
integer slice = ((UInt(index) - (UInt(index) MOD nreg)) + offset) MOD sl

for r = 0 to nreg-1
    bits(VL) result = ZAslice[n, esize, vertical, slice + r, VL];
    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