MOV (vector to tile, two registers)

Move two vector registers to two ZA tile slices

The instruction operates on two 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 2 in the range 0 to the number of elements in a 128-bit vector segment minus 2.

This instruction is unpredicated.

This is an alias of MOVA (vector to tile, two registers). This means:

- The encodings in this description are named to match the encodings of MOVA (vector to tile, two registers).
- The description of MOVA (vector to tile, two registers) gives the operational pseudocode, any constrained unpredictable behavior, and any operational information for this instruction.

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

8-bit

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 0	0	0	0 0 0 1 0 0	V	Rs	0 0 0	Zn	0 0	0	off3
		size<1>	size<0>								

MOV ZA0<HV>.B[<Ws>, <offs1>:<offs2>], { <Zn1>.B-<Zn2>.B }

is equivalent to

MOVA ZA0<HV>.B[<Ws>, <offs1>:<offs2>], { <Zn1>.B-<Zn2>.B }

and is always the preferred disassembly.

16-bit

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 0	0	1	0 0 0 1 0 0	V	Rs	0 0 0	Zn	0 0	0	ZAd	off2
		size<1>	size<0>									

```
MOV <ZAd><HV>.H[<Ws>, <offs1>:<offs2>], { <Zn1>.H-<Zn2>.H }
```

is equivalent to

```
MOVA <ZAd><HV>.H[<Ws>, <offs1>:<offs2>], { <Zn1>.H-<Zn2>.H }
```

and is always the preferred disassembly.

32-bit

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 0	1	0	0 0 0 1 0 0	V	Rs	0 0 0	Zn	0 0	0	ZAd	o1
		size<1>	size<0>									

```
MOV <ZAd><HV>.S[<Ws>, <offs1>:<offs2>], { <Zn1>.S-<Zn2>.S }
```

is equivalent to

```
MOVA <ZAd><HV>.S[<Ws>, <offs1>:<offs2>], { <Zn1>.S-<Zn2>.S }
```

and is always the preferred disassembly.

64-bit

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 0	1	1	0 0 0 1 0 0	V	Rs	0 0 0	Zn	0 0	0	ZAd
		size<1>	size<0>								

```
MOV <ZAd><HV>.D[<Ws>, <offs1>:<offs2>], { <Zn1>.D-<Zn2>.D }
```

is equivalent to

```
MOVA <ZAd><HV>.D[<Ws>, <offs1>:<offs2>], { <Zn1>.D-<Zn2>.D }
```

and is always the preferred disassembly.

Assembler Symbols

<ZAd>	For the 16-bit variant: is the name of the ZA tile ZA0-ZA1 to be accessed, encoded in the "ZAd" field. For the 32-bit variant: is the name of the ZA tile ZA0-ZA3 to be accessed, encoded in the "ZAd" field. For the 64-bit variant: is the name of the ZA tile ZA0-ZA7 to be accessed, encoded in the "ZAd" 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 two consecutive slices, encoded as "off3" field times 2. For the 16-bit variant: is the slice index offset, pointing to first of two consecutive slices, encoded as "off2" field times 2. For the 32-bit variant: is the slice index offset, pointing to first of two consecutive slices, encoded as "o1" field times 2. For the 64-bit variant: is the slice index offset, pointing to first of two consecutive slices, with implicit value 0.
<offs2>	For the 8-bit variant: is the slice index offset, pointing to last of two consecutive slices, encoded as "off3" field times 2 plus 1. For the 16-bit variant: is the slice index offset, pointing to last of two consecutive slices, encoded as "off2" field times 2 plus 1. For the 32-bit variant: is the slice index offset, pointing to last of two consecutive slices, encoded as "o1" field times 2 plus 1. For the 64-bit variant: is the slice index offset, pointing to last of two consecutive slices, with implicit value 1.
<Zn1>	Is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zn" times 2.
<Zn2>	Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zn" times 2 plus 1.

Operation

The description of MOVA (vector to tile, two registers) gives the operational pseudocode for this instruction.

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