

MOVA (tile to vector, single)

Move ZA tile slice to vector register

The instruction operates on individual horizontal or vertical slices within a named ZA tile of the specified element size. The slice number within the tile is selected by the sum of the slice index register and immediate offset, modulo the number of such elements in a vector. The immediate offset is in the range 0 to the number of elements in a 128-bit vector segment minus 1. Inactive elements in the destination vector remain unmodified.

This instruction is used by the alias [MOV \(tile to vector, single\)](#).

It has encodings from 5 classes: [8-bit](#) , [16-bit](#) , [32-bit](#) , [64-bit](#) and [128-bit](#)

8-bit

(FEAT_SME)

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	0	1	0	V	Rs	Pg	0	off4	Zd										
size<1>size<0>								Q																							

MOVA <Zd> .B, <Pg>/M, ZA0<HV> .B[<Ws>, <offs>]

```

if !HaveSME() then UNDEFINED;
integer g = UInt(Pg);
integer s = UInt('011':Rs);
integer n = 0;
integer offset = UInt(off4);
constant integer esize = 8;
integer d = UInt(Zd);
boolean vertical = V == '1';

```

16-bit

(FEAT_SME)

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	0	1	0	V	Rs	Pg	0	ZAn	off3	Zd									
size<1>size<0>								Q																							

MOVA <Zd> .H, <Pg>/M, <ZAn><HV> .H[<Ws>, <offs>]

```

if !HaveSME() then UNDEFINED;
integer g = UInt(Pg);
integer s = UInt('011':Rs);
integer n = UInt(ZAn);
integer offset = UInt(off3);
constant integer esize = 16;
integer d = UInt(Zd);
boolean vertical = V == '1';

```

32-bit

(FEAT_SME)

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	V	Rs	Pg	0	ZAn	off2											Zd	
								size<1>size<0>		Q																						

MOVA <Zd>.S, <Pg>/M, <ZAn><HV>.S[<Ws>, <offs>]

```
if !HaveSME() then UNDEFINED;
integer g = UInt(Pg);
integer s = UInt('011':Rs);
integer n = UInt(ZAn);
integer offset = UInt(off2);
constant integer esize = 32;
integer d = UInt(Zd);
boolean vertical = V == '1';
```

64-bit

(FEAT_SME)

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	0	1	0	V	Rs	Pg	0	ZAn	o1										Zd	
								size<1>size<0>		Q																						

MOVA <Zd>.D, <Pg>/M, <ZAn><HV>.D[<Ws>, <offs>]

```
if !HaveSME() then UNDEFINED;
integer g = UInt(Pg);
integer s = UInt('011':Rs);
integer n = UInt(ZAn);
integer offset = UInt(o1);
constant integer esize = 64;
integer d = UInt(Zd);
boolean vertical = V == '1';
```

128-bit

(FEAT_SME)

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	0	1	1	V	Rs	Pg	0	ZAn											Zd	
								size<1>size<0>		Q																						

MOVA <Zd>.Q, <Pg>/M, <ZAn><HV>.Q[<Ws>, <offs>]

```
if !HaveSME() then UNDEFINED;
integer g = UInt(Pg);
integer s = UInt('011':Rs);
integer n = UInt(ZAn);
integer offset = 0;
constant integer esize = 128;
integer d = UInt(Zd);
boolean vertical = V == '1';
```

Assembler Symbols

- <Zd> Is the name of the destination scalable vector register, encoded in the "Zd" field.
- <Pg> Is the name of the governing scalable predicate register P0-P7, encoded in the "Pg" field.
- <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.
- For the 128-bit variant: is the name of the ZA tile ZA0-ZA15 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.
- <offs> For the 8-bit variant: is the slice index offset, in the range 0 to 15, encoded in the "off4" field.
- For the 16-bit variant: is the slice index offset, in the range 0 to 7, encoded in the "off3" field.
- For the 32-bit variant: is the slice index offset, in the range 0 to 3, encoded in the "off2" field.
- For the 64-bit variant: is the slice index offset, in the range 0 to 1, encoded in the "off1" field.
- For the 128-bit variant: is the slice index offset 0.

Operation

```
CheckStreamingSVEAndZAAEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer dim = VL DIV esize;
bits(PL) mask = P[g, PL];
bits(32) index = X[s, 32];
integer slice = (UInt(index) + offset) MOD dim;
bits(VL) operand = ZAslice[n, esize, vertical, slice, VL];
bits(VL) result = Z[d, VL];
```

```

for e = 0 to dim-1
    bits(esize) element = Elem[operand, e, esize];
    if ActivePredicateElement(mask, e, esize) then
        Elem[result, e, esize] = element;

Z[d, 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 operand registers when its governing predicate register contains the same value for each execution.
 - 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 operand registers when its governing predicate register contains the same value for each execution.
 - The values of the NZCV flags.

[Base
Instructions](#)

[SIMD&FP
Instructions](#)

[SVE
Instructions](#)

[SME
Instructions](#)

[Index by
Encoding](#)

[Sh
Pseudocode](#)

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

Copyright Â© 2010-2023 Arm Limited or its affiliates. All rights reserved. This
document is Non-Confidential.