

## EORTB

Interleaving exclusive OR (top, bottom)

Interleaving exclusive OR between the odd-numbered elements of the first source vector register and the even-numbered elements of the second source vector register, placing the result in the odd-numbered elements of the destination vector, leaving the even-numbered elements unchanged. This instruction is unpredicated.

|    |    |    |    |    |    |    |    |      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |
|----|----|----|----|----|----|----|----|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|
| 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 |
| 0  | 1  | 0  | 0  | 0  | 1  | 0  | 1  | size | 0  | Zm |    |    | 1  | 0  | 0  | 1  | 0  | 1  | Zn |    |    | Zd |   |   |   |   |   |   |   |   |   |
|    |    |    |    |    |    |    |    |      |    |    |    |    |    |    |    |    | tb |    |    |    |    |    |   |   |   |   |   |   |   |   |   |

**EORTB** <Zd>.<T>, <Zn>.<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);
integer sel1 = 1;
integer sel2 = 0;
```

## 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.

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

## Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer elements = VL DIV (2 * esize);
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[m, VL];
```

```

bits(VL) result = Z[d, VL];

for e = 0 to elements-1
    bits(esize) element1 = Elem[operand1, 2*e + sel1, esize];
    bits(esize) element2 = Elem[operand2, 2*e + sel2, esize];
    Elem[result, 2*e + sel1, esize] = element1 EOR element2;

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.

This instruction might be immediately preceded in program order by a MOVPRFX instruction. The MOVPRFX instruction must conform to all of the following requirements, otherwise the behavior of the MOVPRFX and this instruction is unpredictable:

- The MOVPRFX instruction must be unpredicated.
- The MOVPRFX instruction must specify the same destination register as this instruction.
- The destination register must not refer to architectural register state referenced by any other source operand register of this instruction.

[Base  
Instructions](#)

[SIMD&FP  
Instructions](#)

[SVE  
Instructions](#)

[SME  
Instructions](#)

[Index by  
Encoding](#)

[Sh  
Pseud](#)

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.