

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
constant integer esize = 32;
integer index = UInt(i2h:i2l);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer da = UInt(Zda);
integer sel = 0;
```

## Assembler Symbols

<Zda>	Is the name of the third source and destination scalable vector register, encoded in the "Zda" field.
<Zn>	Is the name of the first source scalable vector register, encoded in the "Zn" field.
<Zm>	For the 32-bit variant: is the name of the second source scalable vector register Z0-Z7, encoded in the "Zm" field.  For the 64-bit variant: is the name of the second source scalable vector register Z0-Z15, encoded in the "Zm" field.
<imm>	For the 32-bit variant: is the element index, in the range 0 to 7, encoded in the "i3h:i3l" fields.  For the 64-bit variant: is the element index, in the range 0 to 3, encoded in the "i2h:i2l" fields.

## Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer elements = VL DIV (2 * esize);
constant integer eltspersegment = 128 DIV (2 * esize);
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[m, VL];
bits(VL) result = Z[da, VL];

for e = 0 to elements-1
    integer s = e - (e MOD eltspersegment);
    integer element1 = UInt(Elem[operand1, 2 * e + sel, esize]);
    integer element2 = UInt(Elem[operand2, 2 * s + index, esize]);
    bits(2*esize) product = (element1 * element2) <2*esize-1:0>;
    Elem[result, e, 2*esize] = Elem[result, e, 2*esize] - product;

Z[da, 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  
Pseu](#)

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.