

## SDOT (2-way, indexed)

Signed integer indexed dot product

The signed integer indexed dot product instruction computes the dot product of a group of two signed 16-bit integer values held in each 32-bit element of the first source vector multiplied by a group of two signed 16-bit integer values in an indexed 32-bit element of the second source vector, and then destructively adds the widened dot product to the corresponding 32-bit element of the destination vector.

The groups within the second source vector are specified using an immediate index which selects the same group position within each 128-bit vector segment. The index range is from 0 to 3. This instruction is unpredicated.

### SVE2

(FEAT\_SVE2p1)

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	0	1	0	0	i2		Zm		1	1	0	0	1	0		Zn						Zda			
																						U									

**SDOT <Zda>.S, <Zn>.H, <Zm>.H[<imm>]**

```

if !HaveSME2() && !HaveSVE2p1() then UNDEFINED;
constant integer esize = 32;
integer index = UInt(i2);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer da = UInt(Zda);

```

### 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> Is the name of the second source scalable vector register Z0-Z7, encoded in the "Zm" field.
- <imm> Is the immediate index of a group of two 16-bit elements within each 128-bit vector segment, in the range 0 to 3, encoded in the "i2" field.

### Operation

```

CheckSVEEnabled();
constant integer VL = CurrentVL;

```

```

constant integer PL = VL DIV 8;
constant integer elements = VL DIV esize;
constant integer eltspersegment = 128 DIV esize;
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[m, VL];
bits(VL) operand3 = Z[da, VL];
bits(VL) result;

for e = 0 to elements-1
    integer segmentbase = e - (e MOD eltspersegment);
    integer s = segmentbase + index;
    bits(esize) res = Elem[operand3, e, esize];
    for i = 0 to 1
        integer element1 = SInt(Elem[operand1, 2 * e + i, esize DIV 2])
        integer element2 = SInt(Elem[operand2, 2 * s + i, esize DIV 2])
        res = res + element1 * element2;
    Elem[result, e, esize] = res;

Z[da, VL] = result;

```

### Operational information

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.