SDOT (4-way, indexed)

Signed integer indexed dot product

The signed integer indexed dot product instruction computes the dot product of a group of four signed 8-bit or 16-bit integer values held in each 32-bit or 64-bit element of the first source vector multiplied by a group of four signed 8-bit or 16-bit integer values in an indexed 32-bit or 64-bit element of the second source vector, and then destructively adds the widened dot product to the corresponding 32-bit or 64-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 one less than the number of groups per 128-bit segment, encoded in 1 to 2 bits depending on the size of the group. This instruction is unpredicated.

It has encodings from 2 classes: [32-bit](#) and [64-bit](#)

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
0 1 0 0 0 1 0 0	1	0	1	i2	Zm	0 0 0 0 0 0	Zn	Zda
	size<1>	size<0>				U		

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

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

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
0 1 0 0 0 1 0 0	1	1	1	i1	Zm	0 0 0 0 0 0	Zn	Zda
	size<1>	size<0>				U		

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

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
constant integer esize = 64;
integer index = UInt(i1);
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>　　　　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 immediate index of a 32-bit group of four 8-bit values within each 128-bit vector segment, in the range 0 to 3, encoded in the "i2" field.

　　　　　　　For the 64-bit variant: is the immediate index of a 64-bit group of four 16-bit values within each 128-bit vector segment, in the range 0 to 1, encoded in the "i1" 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 3
        integer element1 = SInt(Elem[operand1, 4 * e + i, esize DIV 4])
        integer element2 = SInt(Elem[operand2, 4 * s + i, esize DIV 4])
        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.