SUDOT (by element)

Dot product index form with signed and unsigned integers. This instruction performs the dot product of the four signed 8-bit integer values in each 32-bit element of the first source register with the four unsigned 8-bit integer values in an indexed 32-bit element of the second source register, accumulating the result into the corresponding 32-bit element of the destination vector.

From Armv8.2 to Armv8.5, this is an optional instruction. From Armv8.6 it is mandatory for implementations that include Advanced SIMD to support it. *ID_AA64ISAR1_EL1*.I8MM indicates whether this instruction is supported.

Vector
(FEAT_I8MM)

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	Q	0	0	1	1	1	1	0	0	L	M	Rm	1	1	1	1	H	0	Rn	Rd
								US												

```
SUDOT <Vd>.<Ta>, <Vn>.<Tb>, <Vm>.4B[<index>]
```

```
if !IsFeatureImplemented(FEAT_I8MM) then UNDEFINED;
boolean op1_unsigned = (US == '1');
boolean op2_unsigned = (US == '0');
integer n = UInt(Rn);
integer m = UInt(M:Rm);
integer d = UInt(Rd);
integer i = UInt(H:L);
constant integer datasize = 64 << UInt(Q);
integer elements = datasize DIV 32;
```

Assembler Symbols

<Vd>
Is the name of the SIMD&FP third source and destination register, encoded in the "Rd" field.

<Ta>
Is an arrangement specifier, encoded in “Q”:

Q	<Ta>
0	2S
1	4S

<Vn>
Is the name of the first SIMD&FP source register, encoded in the "Rn" field.

<Tb>

Is an arrangement specifier, encoded in “Q”:

Q	<Tb>
0	8B
1	16B

<Vm>

Is the name of the second SIMD&FP source register, encoded in the "M:Rm" fields.

<index>

Is the immediate index of a 32-bit group of four 8-bit values in the range 0 to 3, encoded in the "H:L" fields.

Operation

```
CheckFPAdvSIMDEnabled64();
bits(datasize) operand1 = V[n, datasize];
bits(128) operand2 = V[m, 128];
bits(datasize) operand3 = V[d, datasize];
bits(datasize) result;

for e = 0 to elements-1
    bits(32) res = Elem[operand3, e, 32];
    for b = 0 to 3
        integer element1 = Int(Elem[operand1, 4*e+b, 8], op1_unsigned);
        integer element2 = Int(Elem[operand2, 4*i+b, 8], op2_unsigned);
        res = res + element1 * element2;
    Elem[result, e, 32] = res;
V[d, datasize] = result;
```

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