

FDOT (multiple and indexed vector)

Multi-vector half-precision floating-point dot-product by indexed element

The instruction computes the fused sum-of-products of a pair of half-precision floating-point values held in the corresponding 32-bit elements of the two or four first source vectors and the indexed 32-bit element of the second source vector, without intermediate rounding. The single-precision sum-of-products results are destructively added to the corresponding single-precision elements of the ZA single-vector groups.

The half-precision floating-point pairs within the second source vector are specified using an immediate index which selects the same pair position within each 128-bit vector segment. The element index range is from 0 to 3. The vector numbers forming the single-vector group within each half of or each quarter of the ZA array are selected by the sum of the vector select register and immediate offset, modulo half or quarter the number of ZA array vectors.

The vector group symbol, VGx2 or VGx4, indicates that the ZA operand consists of two or four ZA single-vector groups respectively. The vector group symbol is preferred for disassembly, but optional in assembler source code.

This instruction follows SME ZA-targeting floating-point behaviors.

This instruction is unpredicated.

It has encodings from 2 classes: [Two ZA single-vectors](#) and [Four ZA single-vectors](#)

Two ZA single-vectors (FEAT_SME2)

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
1	1	0	0	0	0	0	1	0	1	0	1		Zm		0	Rv	1	i2		Zn		0	0	1		off3					

FDOT ZA.S[<Wv>, <offs>{, VGx2}], { <Zn1>.H-<Zn2>.H }, <Zm>.H[<index>]

```
if !HaveSME2() then UNDEFINED;
integer v = UInt('010':Rv);
integer n = UInt(Zn:'0');
integer m = UInt('0':Zm);
integer offset = UInt(off3);
integer index = UInt(i2);
constant integer nreg = 2;
```

Four ZA single-vectors (FEAT_SME2)

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
1	1	0	0	0	0	0	1	0	1	0	1		Zm		1	Rv	1	i2		Zn		0	0	0	1		off3				

```
FDOT ZA.S[<Wv>, <offs>{, VGx4}], { <Zn1>.H- <Zn4>.H }, <Zm>.H[<index>]
```

```
if !HaveSME2() then UNDEFINED;
integer v = UInt('010':Rv);
integer n = UInt(Zn:'00');
integer m = UInt('0':Zm);
integer offset = UInt(off3);
integer index = UInt(i2);
constant integer nreg = 4;
```

Assembler Symbols

<Wv>	Is the 32-bit name of the vector select register W8-W11, encoded in the "Rv" field.
<offs>	Is the vector select offset, in the range 0 to 7, encoded in the "off3" field.
<Zn1>	For the two ZA single-vectors variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zn" times 2. For the four ZA single-vectors variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zn" times 4.
<Zn4>	Is the name of the fourth scalable vector register of a multi-vector sequence, encoded as "Zn" times 4 plus 3.
<Zn2>	Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zn" times 2 plus 1.
<Zm>	Is the name of the second source scalable vector register Z0-Z15, encoded in the "Zm" field.
<index>	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

```
CheckStreamingSVEAndZAAEnabled();
constant integer VL = CurrentVL;
constant integer elements = VL DIV 32;
integer vectors = VL DIV 8;
integer vstride = vectors DIV nreg;
integer eltspersegment = 128 DIV 32;
bits(32) vbase = X[v, 32];
integer vec = (UInt(vbase) + offset) MOD vstride;
bits(VL) result;

for r = 0 to nreg-1
    bits(VL) operand1 = Z[n+r, VL];
    bits(VL) operand2 = Z[m, VL];
    bits(VL) operand3 = ZAvector[vec, VL];
    for e = 0 to elements-1
```

```

bits(16) elt1_a = Elem[operand1, 2 * e + 0, 16];
bits(16) elt1_b = Elem[operand1, 2 * e + 1, 16];
integer segmentbase = e - (e MOD eltspersegment);
integer s = segmentbase + index;
bits(16) elt2_a = Elem[operand2, 2 * s + 0, 16];
bits(16) elt2_b = Elem[operand2, 2 * s + 1, 16];
bits(32) sum = Elem[operand3, e, 32];
sum = FPDotAdd\_ZA(sum, elt1_a, elt1_b, elt2_a, elt2_b, FPCR[]);
Elem[result, e, 32] = sum;
ZAvector[vec, VL] = result;
vec = vec + vstride;

```

[Base
Instructions](#)

[SIMD&FP
Instructions](#)

[SVE
Instructions](#)

[SME
Instructions](#)

[Index by
Encoding](#)

[Sh
Pseudocode](#)

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.