SME Index by
Instructions Encoding

Sh

Pseu

UDOT (4-way, multiple and single vector)

SIMD&FP

Instructions

Multi-vector unsigned integer dot-product by vector

The unsigned integer dot product instruction computes the dot product of four unsigned 8-bit or 16-bit integer values held in each 32-bit or 64-bit element of the two or four first source vectors and four unsigned 8-bit or 16-bit integer values in the corresponding 32-bit or 64-bit element of the second source vector. The widened dot product result is destructively added to the corresponding 32-bit or 64-bit element of the ZA single-vector groups. 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.

SVE

Instructions

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 is unpredicated.

ID_AA64SMFR0_EL1.I16I64 indicates whether the 16-bit integer variant is implemented.

It has encodings from 2 classes: <u>Two ZA single-vectors</u> and <u>Four ZA single-vectors</u>

Two ZA single-vectors (FEAT_SME2)

Base

Instructions

```
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 1 0 0 0 0 0 1 0 | sz | 1 0 | Zm | 0 | Rv | 1 0 1 | Zn | 1 0 | off3
```

```
UDOT ZA.<T>[<Wv>, <offs>{, VGx2}], { <Zn1>.<Tb>-<Zn2>.<Tb>}, <Zm>.<
```

```
if !HaveSME2() then UNDEFINED;
if sz == '1' && !HaveSMEI16I64() then UNDEFINED;
integer v = UInt('010':Rv);
constant integer esize = 32 << UInt(sz);
integer n = UInt(Zn);
integer m = UInt('0':Zm);
integer offset = UInt(off3);
constant integer nreg = 2;</pre>
```

Four ZA single-vectors (FEAT SME2)

31 30	29	28	27	26	25	24	23	22	21	20	19 18 17	16 1	5 1	4 13	12	11	10	9	8	7	6	5	4	3	2	1	0
1 1	0	0	0	0	0	1	0	sz	1	1	Zm	C	ıΤ	Rv	1	0	1			Zn			1	0		off3	3

U

```
UDOT ZA.<T>[<Wv>, <offs>{, VGx4}], { <Zn1>.<Tb>-<Zn4>.<Tb>}, <Zm>.<
```

```
if !HaveSME2() then UNDEFINED;
if sz == '1' && !HaveSMEI16I64() then UNDEFINED;
integer v = UInt('010':Rv);
constant integer esize = 32 << UInt(sz);
integer n = UInt(Zn);
integer m = UInt('0':Zm);
integer offset = UInt(off3);
constant integer nreg = 4;</pre>
```

Assembler Symbols

<T>

Is the size specifier, encoded in "sz":

SZ	<t></t>
0	S
1	D

<Wv> Is the 32-bit name of the vector select register W8-W11, encoded in the "Ry" field.

<offs> Is the vector select offset, in the range 0 to 7, encoded in
the "off3" field.

<Zn1> Is the name of the first scalable vector register of a multivector sequence, encoded as "Zn".

SZ	<tb></tb>						
0	В						
1	Н						

<Zn4> Is the name of the fourth scalable vector register of a multivector sequence, encoded as "Zn" plus 3 modulo 32.

<Zn2> Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zn" plus 1 modulo 32.

<Zm> Is the name of the second source scalable vector register Z0-Z15, encoded in the "Zm" field.

Operation

```
CheckStreamingSVEAndZAEnabled();
constant integer VL = CurrentVL;
constant integer elements = VL DIV esize;
integer vectors = VL DIV 8;
integer vstride = vectors DIV nreg;
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) MOD 32, VL];
    bits(VL) operand2 = Z[m, VL];
    bits(VL) operand3 = ZAvector[vec, VL];
    for e = 0 to elements-1
        bits(esize) sum = Elem[operand3, e, esize];
        for i = 0 to 3
            integer element1 = UInt(Elem[operand1, 4 * e + i, esize DIV integer element2 = UInt(Elem[operand2, 4 * e + i, esize DIV sum = sum + element1 * element2;
        Elem[result, e, esize] = sum;
        ZAvector[vec, VL] = result;
        vec = vec + vstride;
```

<u>Instructions</u> <u>Instructions</u> <u>Instructions</u> <u>Encoding</u>

SVE

SME

Index by

Sh

Pseu

SIMD&FP

Base

 $Internal\ version\ only: is a\ v33.64,\ AdvSIMD\ v29.12,\ pseudocode\ no_diffs_2023_09_RC2,\ sve\ v2023-06_rel\ ;\ Build\ timestamp:\ 2023-09-18T17:56$

Copyright © 2010-2023 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.