

SUB (array results, multiple vectors)

Subtract multi-vector from multi-vector with ZA array vector results

Subtract all corresponding elements of the two or four second source vectors from first source vectors and place the results in the corresponding elements 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.

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 64-bit integer variant is implemented.

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	1	sz	1		Zm		0	0	Rv	1	1	0		Zn		0	1	1		off3				
																S															

SUB ZA.<T>[<Wv>, <offs>{, VGx2}], { <Zn1>.<T>-<Zn2>.<T> }, { <Zm1>.<T>-<Zm2>.<T> }

```

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:'0');
integer m = UInt(Zm:'0');
integer offset = UInt(off3);
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	1	sz	1		Zm		0	1	0	Rv		1	1	0		Zn		0	0	1	1		off3	
																S															

SUB ZA.<T>[<Wv>, <offs>{, VGx4}], { <Zn1>.<T>-<Zn4>.<T> }, { <Zm1>.<T>-<Zm4>.<T> }

```

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:'00');
integer m = UInt(Zm:'00');
integer offset = UInt(off3);
constant integer nreg = 4;

```

Assembler Symbols

<T>

Is the size specifier, encoded in "sz":

sz	<T>
0	S
1	D

<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.

<Zm1>

For the two ZA single-vectors variant: is the name of the first scalable vector register of a multi-vector sequence, encoded as "Zm" 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 "Zm" times 4.

<Zm4>

Is the name of the fourth scalable vector register of a multi-vector sequence, encoded as "Zm" times 4 plus 3.

<Zm2>

Is the name of the second scalable vector register of a multi-vector sequence, encoded as "Zm" times 2 plus 1.

Operation

```

CheckStreamingSVEAndZAAEnabled();
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, VL];
  bits(VL) operand2 = Z[m+r, VL];
  for e = 0 to elements-1
    bits(esize) element1 = Elem[operand1, e, esize];
    bits(esize) element2 = Elem[operand2, e, esize];
    Elem[result, e, esize] = element1 - element2;
  ZAvector[vec, VL] = result;
  vec = vec + vstride;

```

Operational information

If PSTATE.DIT is 1:

- The execution time of this instruction is independent of:
 - The values of the data supplied in any of its registers.
 - The values of the NZCV flags.
- The response of this instruction to asynchronous exceptions does not vary based on:
 - The values of the data supplied in any of its registers.
 - The values of the NZCV flags.

[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.