

SBCLT

Subtract with carry long (top)

Subtract the odd-numbered elements of the first source vector and the inverted 1-bit carry from the least-significant bit of the odd-numbered elements of the second source vector from the even-numbered elements of the destination and accumulator vector. The 1-bit carry output is placed in the corresponding odd-numbered element of the destination vector.

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	1	1	sz	0					Zm		1	1	0	1	0	1				Zn				Zda	
																						T									

SBCLT <Zda>.<T>, <Zn>.<T>, <Zm>.<T>

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
constant integer esize = 32 << UInt(sz);
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.

<T> Is the size specifier, encoded in "sz":

sz	<T>
0	S
1	D

<Zn> Is the name of the first source scalable vector register, encoded in the "Zn" field.

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

Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer pairs = VL DIV (esize * 2);
bits(VL) operand = Z[n, VL];
bits(VL) carries = Z[m, VL];
bits(VL) result = Z[da, VL];

for p = 0 to pairs-1
    bits(esize) element1 = Elem[result, 2*p + 0, esize];
```

```

bits(esize) element2 = Elem[operand, 2*p + 1, esize];
bit carry_in = Elem[carries, 2*p + 1, esize]<0>;

(res, nzcw) = AddWithCarry(element1, NOT(element2), carry_in);
carry_out = nzcw<1>;

Elem[result, 2*p + 0, esize] = res;
Elem[result, 2*p + 1, esize] = ZeroExtend(carry_out, esize);

Z[da, VL] = result;

```

Operational information

If FEAT_SVE2 is implemented or FEAT_SME is implemented, then 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.

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.

[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.