SSUBWT

Signed subtract wide (top)

Subtract the even-numbered signed elements of the second source vector from the overlapping double-width elements of the first source vector and place the results in the corresponding double-width elements of the destination vector. This instruction is unpredicated.

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	size	0	Zm	0 1 0	1	0	1	Zn	Zd

S U T

      SSUBWT **\<Zd\>.\<T\>, \<Zn\>.\<T\>, \<Zm\>.\<Tb\>**

```
if !HaveSVE2() && !HaveSME() then UNDEFINED;
if size == '00' then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer d = UInt(Zd);
```

Assembler Symbols

<Zd>   Is the name of the destination scalable vector register, encoded in the "Zd" field.

<T>

       Is the size specifier, encoded in "size":

size	\<T\>
00	RESERVED
01	H
10	S
11	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.

<Tb>

       Is the size specifier, encoded in "size":

size	\<Tb\>
00	RESERVED
01	B
10	H
11	S

Operation

```
CheckSVEEnabled();
constant integer VL = CurrentVL;
constant integer elements = VL DIV esize;
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[m, VL];
bits(VL) result;

for e = 0 to elements-1
    integer element1 = SInt(Elem[operand1, e, esize]);
    integer element2 = SInt(Elem[operand2, 2*e + 1, esize DIV 2]);
    Elem[result, e, esize] = (element1 - element2)<esize-1:0>;

Z[d, 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.