UMLALT (vectors)

Unsigned multiply-add long to accumulator (top)

Multiply the corresponding odd-numbered unsigned elements of the first and second source vectors and destructively add to the overlapping double-width elements of the addend 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	0	size	0	Zm	0	1	0	0	1	1	Zn	Zda

S U T

UMLALT **<Zda>.<T>**, **<Zn>.<Tb>**, **<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 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 "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.

<Tb>
            Is the size specifier, encoded in "size":

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

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

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 = Z[da, VL];

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

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 Pseu