BFMLALB, BFMLALT (by element)

BFloat16 floating-point widening multiply-add long (by element) widens the even-numbered (bottom) or odd-numbered (top) 16-bit elements in the first source vector, and the indexed element in the second source vector from Bfloat16 to single-precision format. The instruction then multiplies and adds these values without intermediate rounding to single-precision elements of the destination vector that overlap with the corresponding BFloat16 elements in the first source vector.

ID_AA64ISAR1_EL1.BF16 indicates whether this instruction is supported.

Vector
(FEAT_BF16)

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	Q	0	0	1	1	1	1	1	1	L	M	Rm				1	1	1	1	H	0	Rn					Rd				

```
BFMLAL<bt> <Vd>.4S, <Vn>.8H, <Vm>.H[<index>]
```

```
if !IsFeatureImplemented(FEAT_BF16) then UNDEFINED;
integer n = UInt(Rn);
integer m = UInt('0':Rm);
integer d = UInt(Rd);
integer index = UInt(H:L:M);

integer elements = 128 DIV 32;
integer sel = UInt(Q);
```

Assembler Symbols

<bt>
Is the bottom or top element specifier, encoded in “Q”:

Q	<bt>
0	B
1	T

<Vd>
Is the name of the SIMD&FP destination register, encoded in the "Rd" field.

<Vn>
Is the name of the first SIMD&FP source register, encoded in the "Rn" field.

<Vm>
Is the name of the second SIMD&FP source register, in the range V0 to V15, encoded in the "Rm" field.

<index>
Is the element index, in the range 0 to 7, encoded in the "H:L:M" fields.

Operation

```
CheckFPAdvSIMDEnabled64();
bits(128) result;
bits(128) operand1 = V[n, 128];
bits(128) operand2 = V[m, 128];
bits(128) operand3 = V[d, 128];
bits(16) element2 = Elem[operand2, index, 16];

for e = 0 to elements-1
    bits(16) element1 = Elem[operand1, 2*e+sel, 16];
    bits(32) addend = Elem[operand3, e, 32];
    Elem[result, e, 32] = BFMulAddH(addend, element1, element2, FPCR[])

V[d, 128] = result;
```

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