

## UMLAL, UMLAL2 (by element)

Unsigned Multiply-Add Long (vector, by element). This instruction multiplies each vector element in the lower or upper half of the first source SIMD&FP register by the specified vector element of the second source SIMD&FP register and accumulates the results with the vector elements of the destination SIMD&FP register. The destination vector elements are twice as long as the elements that are multiplied.

The UMLAL instruction extracts vector elements from the lower half of the first source register. The UMLAL2 instruction extracts vector elements from the upper half of the first source register.

Depending on the settings in the [CPACR\\_EL1](#), [CPTR\\_EL2](#), and [CPTR\\_EL3](#) registers, and the current Security state and Exception level, an attempt to execute the instruction might be trapped.

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	1	0	1	1	1	1	size	L	M			Rm		0	0	1	0	H	0				Rn					Rd		
U									o2																						

**UMLAL{2} <Vd>.<Ta>, <Vn>.<Tb>, <Vm>.<Ts> [<index>]**

```

constant integer idxdsize = 64 << UInt(H);
integer index;
bit Rmhi;
case size of
    when '01' index = UInt(H:L:M); Rmhi = '0';
    when '10' index = UInt(H:L); Rmhi = M;
    otherwise UNDEFINED;

integer d = UInt(Rd);
integer n = UInt(Rn);
integer m = UInt(Rmhi:Rm);

constant integer esize = 8 << UInt(size);
constant integer datasize = 64;
integer part = UInt(Q);
integer elements = datasize DIV esize;

boolean unsigned = (U == '1');
boolean sub_op = (o2 == '1');
```

## Assembler Symbols

2

Is the second and upper half specifier. If present it causes the operation to be performed on the upper 64 bits of the registers holding the narrower elements, and is encoded in "Q":

Q	2
0	[absent]
1	[present]

<Vd>

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

<Ta>

Is an arrangement specifier, encoded in "size":

size	<Ta>
00	RESERVED
01	4S
10	2D
11	RESERVED

<Vn>

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

<Tb>

Is an arrangement specifier, encoded in "size:Q":

size	Q	<Tb>
00	x	RESERVED
01	0	4H
01	1	8H
10	0	2S
10	1	4S
11	x	RESERVED

<Vm>

Is the name of the second SIMD&FP source register, encoded in "size:M:Rm":

size	<Vm>
00	RESERVED
01	0 : Rm
10	M : Rm
11	RESERVED

Restricted to V0-V15 when element size <Ts> is H.

<Ts>

Is an element size specifier, encoded in “size”:

size	<Ts>
00	RESERVED
01	H
10	S
11	RESERVED

<index>

Is the element index, encoded in “size:L:H:M”:

size	<index>
00	RESERVED
01	H:L:M
10	H:L
11	RESERVED

## Operation

```
CheckFPAdvSIMDEnabled64();
bits(datasize) operand1 = Vpart[n, part, datasize];
bits(idxdsize) operand2 = V[m, idxdsize];
bits(2*datasize) operand3 = V[d, 2*datasize];
bits(2*datasize) result;
integer element1;
integer element2;
bits(2*esize) product;

element2 = Int(Elem[operand2, index, esize], unsigned);
for e = 0 to elements-1
    element1 = Int(Elem[operand1, e, esize], unsigned);
    product = (element1*element2)<2*esize-1:0>;
    if sub_op then
        Elem[result, e, 2*esize] = Elem[operand3, e, 2*esize] - product
    else
        Elem[result, e, 2*esize] = Elem[operand3, e, 2*esize] + product
V[d, 2*datasize] = result;
```

## 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  
Pseu](#)

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.