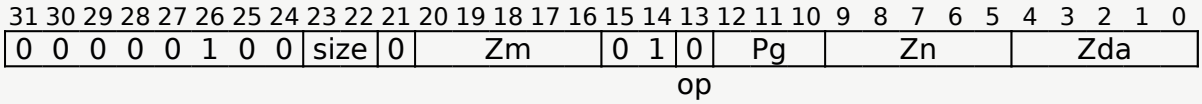
MLA (vectors)

Multiply-add vectors (predicated), writing addend [Zda = Zda + Zn * Zm]

Multiply the corresponding active elements of the first and second source vectors and add to elements of the third source (addend) vector. Destructively place the results in the destination and third source (addend) vector. Inactive elements in the destination vector register remain unmodified.



MLA <Zda>.<T>, <Pg>/M, <Zn>.<T>, <Zm>.<T>

```
if !HaveSVE() && !HaveSME() then UNDEFINED;
constant integer esize = 8 << UInt(size);
integer g = UInt(Pg);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer da = UInt(Zda);
boolean sub_op = FALSE;
```

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	B
01	H
10	S
11	D

<Pg> Is the name of the governing scalable predicate register P0-P7, encoded in the "Pg" field.

<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 PL = VL DIV 8;
constant integer elements = VL DIV esize;
bits(PL) mask = P[g, PL];
bits(VL) operand1 = if AnyActiveElement(mask, esize) then Z[n, VL] else
bits(VL) operand2 = if AnyActiveElement(mask, esize) then Z[m, VL] else
bits(VL) operand3 = Z[da, VL];
bits(VL) result;

for e = 0 to elements-1
    if ActivePredicateElement(mask, e, esize) then
        integer element1 = UInt(Elem[operand1, e, esize]);
        integer element2 = UInt(Elem[operand2, e, esize]);
        integer product = element1 * element2;
        if sub_op then
            Elem[result, e, esize] = Elem[operand3, e, esize] - product
        else
            Elem[result, e, esize] = Elem[operand3, e, esize] + product
    else
        Elem[result, e, esize] = Elem[operand3, e, 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 operand registers when its governing predicate register contains the same value for each execution.
  - 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 operand registers when its governing predicate register contains the same value for each execution.
  - 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, or be predicated using the same governing predicate register and source element size as this instruction.
- 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.

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