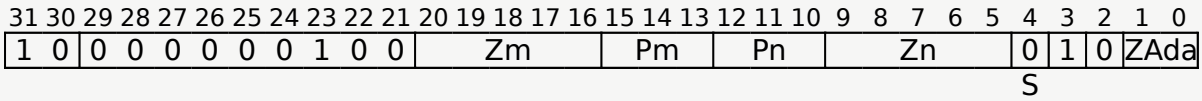
BMOPA

Bitwise exclusive NOR population count outer product and accumulate

This instruction works with 32-bit element ZA tile. This instruction generates an outer product of the first source SVL_SÃ—1 vector and the second source 1Ã—SVL_S vector. Each outer product element is obtained as population count of the bitwise XNOR result of the corresponding 32-bit elements of the first source vector and the second source vector. Each source vector is independently predicated by a corresponding governing predicate. When either source vector element is inactive the corresponding destination tile element remains unmodified. The resulting SVL_SÃ—SVL_S product is then destructively added to the destination tile.

SME2
(FEAT_SME2)



BMOPA <ZAda>.S, <Pn>/M, <Pm>/M, <Zn>.S, <Zm>.S

```
if !HaveSME2() then UNDEFINED;
constant integer esize = 32;
integer a = UInt(Pn);
integer b = UInt(Pm);
integer n = UInt(Zn);
integer m = UInt(Zm);
integer da = UInt(ZAda);
boolean sub_op = FALSE;
```

Assembler Symbols

<ZAda>	Is the name of the ZA tile ZA0-ZA3, encoded in the "ZAda" field.
<Pn>	Is the name of the first governing scalable predicate register P0-P7, encoded in the "Pn" field.
<Pm>	Is the name of the second governing scalable predicate register P0-P7, encoded in the "Pm" 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

```
CheckStreamingSVEAndZAEnabled();
constant integer VL = CurrentVL;
constant integer PL = VL DIV 8;
constant integer dim = VL DIV esize;
bits(PL) mask1 = P[a, PL];
bits(PL) mask2 = P[b, PL];
bits(VL) operand1 = Z[n, VL];
bits(VL) operand2 = Z[m, VL];
bits(dim*dim*esize) operand3 = ZAtile[da, esize, dim*dim*esize];
bits(dim*dim*esize) result;

for row = 0 to dim-1
    bits(esize) element1 = Elem[operand1, row, esize];
    for col = 0 to dim-1
        bits(esize) element2 = Elem[operand2, col, esize];
        bits(esize) element3 = Elem[operand3, row*dim + col, esize];
        if (ActivePredicateElement(mask1, row, esize) &&
              ActivePredicateElement(mask2, col, esize)) then
            integer res = BitCount(NOT(element1 EOR element2));
            if sub_op then res = -res;
            Elem[result, row*dim + col, esize] = element3 + res;
        else
            Elem[result, row*dim + col, esize] = element3;
ZAtile[da, esize, dim*dim*esize] = 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 operand registers when its governing predicate registers contain 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 registers contain the same value for each execution.
  - The values of the NZCV flags.

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