SMOPA (2-way)

Signed integer sum of outer products and accumulate

This instruction works with a 32-bit element ZA tile.

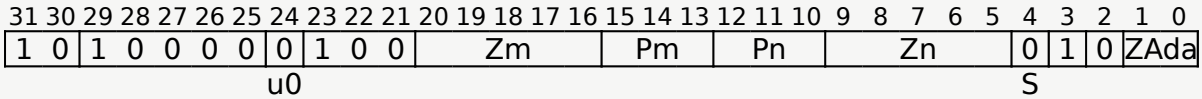
The signed integer sum of outer products and accumulate instructions multiply the sub-matrix in the first source vector by the sub-matrix in the second source vector. The first source holds SVL_SÃ—2 sub-matrix of signed 16-bit integer values, and the second source holds 2Ã—SVL_S sub-matrix of signed 16-bit integer values.

Each source vector is independently predicated by a corresponding governing predicate. When a 16-bit source element is inactive, it is treated as having the value 0.

The resulting SVL_SÃ—SVL_S widened 32-bit integer sum of outer products is then destructively added to the 32-bit integer destination tile. This is equivalent to performing a 2-way dot product and accumulate to each of the destination tile elements.

Each 32-bit container of the first source vector holds 2 consecutive column elements of each row of a SVL_SÃ—2 sub-matrix, and each 32-bit container of the second source vector holds 2 consecutive row elements of each column of a 2Ã—SVL_S sub-matrix.

SME2
(FEAT_SME2)



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

```
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;
boolean unsigned = 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;
integer  prod;

for row = 0 to dim-1
    for col = 0 to dim-1
        bits(esize) sum = Elem[operand3, row*dim+col, esize];
        for k = 0 to 1
            if ActivePredicateElement(mask1, 2*row + k, esize DIV 2) &&
                    ActivePredicateElement(mask2, 2*col + k, esize DIV
                prod = (Int(Elem[operand1, 2*row + k, esize DIV 2], uns
                        Int(Elem[operand2, 2*col + k, esize DIV 2], uns
                if sub_op then prod = -prod;
                sum = sum + prod;

        Elem[result, row*dim+col, esize] = sum;

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