FMOPA (widening)

Half-precision floating-point sum of outer products and accumulate

The half-precision floating-point sum of outer products and accumulate instruction works with a 32-bit element ZA tile.

This instruction widens the SVL_SÃ—2 sub-matrix of half-precision floating-point values held in the first source vector to single-precision floating-point values and multiplies it by the widened 2Ã—SVL_S sub-matrix of half-precision floating-point values in the second source vector to single-precision floating-point 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 +0.0, but if both pairs of source vector elements that correspond to a 32-bit destination element contain Inactive elements, then the destination element remains unmodified.

The resulting SVL_SÃ—SVL_S single-precision floating-point sum of outer products is then destructively added to the single-precision floating-point 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. Similarly, each 32-bit container of the second source vector holds 2 consecutive row elements of each column of a 2Ã—SVL_S sub-matrix.

This instruction follows SME ZA-targeting floating-point behaviors.

SME
(FEAT_SME)

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
1	0	0	0	0	0	0	1	1	0	1	Zm	Pm	Pn	Zn	0 S	0	0	ZAda

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

```
if !HaveSME() then UNDEFINED;
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 32;
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*32) operand3 = ZAtile[da, 32, dim*dim*32];
bits(dim*dim*32) result;

for row = 0 to dim-1
    for col = 0 to dim-1
        // determine row/col predicates
        boolean prow_0 = (ActivePredicateElement(mask1, 2*row + 0, 16))
        boolean prow_1 = (ActivePredicateElement(mask1, 2*row + 1, 16))
        boolean pcol_0 = (ActivePredicateElement(mask2, 2*col + 0, 16))
        boolean pcol_1 = (ActivePredicateElement(mask2, 2*col + 1, 16))

        bits(32) sum = Elem[operand3, row*dim+col, 32];
        if (prow_0 && pcol_0) || (prow_1 && pcol_1) then
            bits(16) erow_0 = (if prow_0 then Elem[operand1, 2*row + 0,
            bits(16) erow_1 = (if prow_1 then Elem[operand1, 2*row + 1,
            bits(16) ecol_0 = (if pcol_0 then Elem[operand2, 2*col + 0,
            bits(16) ecol_1 = (if pcol_1 then Elem[operand2, 2*col + 1,
            if sub_op then
                if prow_0 then erow_0 = FPNeg(erow_0);
                if prow_1 then erow_1 = FPNeg(erow_1);
            sum = FPDotAdd_ZA(sum, erow_0, erow_1, ecol_0, ecol_1, FPCR

        Elem[result, row*dim+col, 32] = sum;

ZAtile[da, 32, dim*dim*32] = 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