USMMLA

Unsigned by signed integer matrix multiply-accumulate

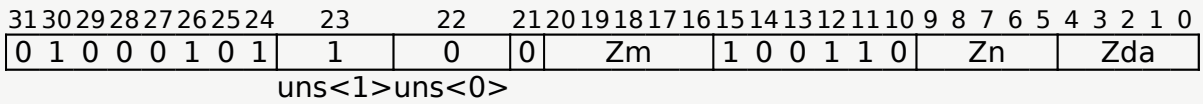
The unsigned by signed integer matrix multiply-accumulate instruction multiplies the 2Ã—8 matrix of unsigned 8-bit integer values held in each 128-bit segment of the first source vector by the 8Ã—2 matrix of signed 8-bit integer values in the corresponding segment of the second source vector. The resulting 2Ã—2 widened 32-bit integer matrix product is then destructively added to the 32-bit integer matrix accumulator held in the corresponding segment of the addend and destination vector. This is equivalent to performing an 8-way dot product per destination element.

This instruction is unpredicated.

ID_AA64ZFR0_EL1.I8MM indicates whether this instruction is implemented.

This instruction is illegal when executed in Streaming SVE mode, unless FEAT_SME_FA64 is implemented and enabled.

SVE
(FEAT_I8MM)



USMMLA <Zda>.S, <Zn>.B, <Zm>.B

```
if !HaveSVE() || !HaveInt8MatMulExt() then UNDEFINED;
integer n = UInt(Zn);
integer m = UInt(Zm);
integer da = UInt(Zda);
boolean op1_unsigned = TRUE;
boolean op2_unsigned = FALSE;
```

Assembler Symbols

<Zda>	Is the name of the third source and destination scalable vector register, encoded in the "Zda" 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

```
CheckNonStreamingSVEEnabled();
constant integer VL = CurrentVL;
constant integer segments = VL DIV 128;
bits(VL) operand1 = Z[n, VL];
```

```
bits(VL) operand2 = Z[m, VL];
bits(VL) operand3 = Z[da, VL];
bits(VL) result = Zeros(VL);
bits(128) op1, op2;
bits(128) res, addend;

for s = 0 to segments-1
    op1    = Elem[operand1, s, 128];
    op2    = Elem[operand2, s, 128];
    addend = Elem[operand3, s, 128];
    res    = MatMulAdd(addend, op1, op2, op1_unsigned, op2_unsigned);
    Elem[result, s, 128] = res;

Z[da, VL] = result;
```

Operational information

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.
- 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