ID_AA64SMFR0_EL1, SME Feature ID Register 0

The ID_AA64SMFR0_EL1 characteristics are:

Purpose

Provides information about the implemented features of the AArch64 Scalable Matrix Extension.

The fields in this register do not follow the standard ID scheme. See 'Alternative ID scheme used for ID_AA64SMFR0_EL1'.

Configuration

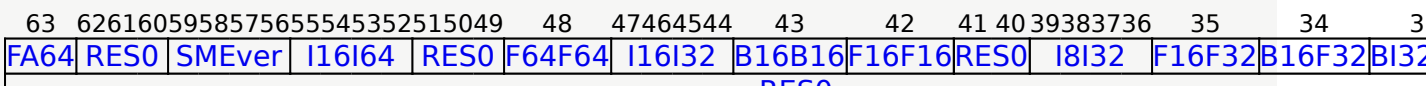
Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, res0 from EL1, EL2, and EL3.

Attributes

ID_AA64SMFR0_EL1 is a 64-bit register.

Field descriptions



FA64, bit [63]

Indicates support for execution of the full A64 instruction set when the PE is in Streaming SVE mode. Defined values are:

FA64	Meaning
0b0	Only those A64 instructions defined as being legal can be executed in Streaming SVE mode.

0b1	All implemented A64 instructions are legal for execution in Streaming SVE mode, when enabled by SMCR_EL1.FA64, SMCR_EL2.FA64, and SMCR_EL3.FA64.

FEAT_SME_FA64 implements the functionality identified by the value 0b1.

Bits [62:60]

Reserved, res0.

SMEver, bits [59:56]
When ID_AA64PFR1_EL1.SME != 0b0000:

Indicates support for SME instructions when FEAT_SME is implemented. Defined values are:

SMEver	Meaning
0b0000	The mandatory SME instructions are implemented.
0b0001	As 0b0000, and adds the mandatory SME2 instructions.
0b0010	As 0b0001, and adds the mandatory SME2.1 instructions.

All other values are reserved.

If FEAT_SME is implemented and FEAT_SME2 is not implemented, the only permitted value is 0b0000.

If FEAT_SME2 is implemented and FEAT_SME2p1 is not implemented, the only permitted value is 0b0001.

FEAT_SME2p1 implements the functionality identified by 0b0010.

Otherwise:

Reserved, res0.

I16I64, bits [55:52]

Indicates SME support for instructions that accumulate into 64-bit integer elements in the ZA array. Defined values are:

I16I64	Meaning

0b0000	Instructions that accumulate into 64-bit integer elements in the ZA array are not implemented.
0b1111	The variants of the ADDHA, ADDVA, SMOPA, SMOPS, SUMOPA, SUMOPS, UMOPA, UMOPS, USMOPA, and USMOPS instructions that accumulate into 64-bit integer tiles are implemented. When FEAT_SME2 is implemented, the variants of the ADD, ADDA, SDOT, SMLALL, SMLSLL, SUB, SUBA, SVDOT, UDOT, UMLALL, UMLSLL, and UVDOT instructions that accumulate into 64-bit integer elements in ZA array vectors are implemented.

All other values are reserved.

FEAT_SME_I16I64 implements the functionality identified by the value 0b1111.

The only permitted values are 0b0000 and 0b1111.

Bits [51:49]

Reserved, res0.

F64F64, bit [48]

Indicates SME support for instructions that accumulate into FP64 double-precision floating-point elements in the ZA array. Defined values are:

F64F64	Meaning
0b0	Instructions that accumulate into double-precision floating-point elements in the ZA array are not implemented.

0b1	The variants of the FMOPA and FMOPS instructions that accumulate into double-precision tiles are implemented. When FEAT_SME2 is implemented, the variants of the FADD, FMLA, FMLS, and FSUB instructions that accumulate into double-precision elements in ZA array vectors are implemented.

FEAT_SME_F64F64 implements the functionality identified by the value 0b1.

I16I32, bits [47:44]

Indicates SME2 support for instructions that accumulate 16-bit outer products into 32-bit integer tiles. Defined values are:

I16I32	Meaning
0b0000	Instructions that accumulate 16-bit outer products into 32-bit integer tiles are not implemented.
0b0101	The SMOPA (2-way), SMOPS (2-way), UMOPA (2-way), and UMOPS (2-way) instructions that accumulate 16-bit outer products into 32-bit integer tiles are implemented.

All other values are reserved.

If FEAT_SME2 is implemented, the only permitted value is 0b0101. Otherwise, the only permitted value is 0b0000.

B16B16, bit [43]

Indicates support for SME2.1 non-widening BFloat16 instructions. Defined values are:

B16B16	Meaning
0b0	SME2.1 non-widening BFloat16 instructions are not implemented.
0b1	SME2.1 non-widening BFloat16 instructions are implemented.

FEAT_B16B16 implements the functionality identified by `0b1`.

This field must indicate the same level of support as ID_AA64ZFR0_EL1.B16B16.

If one or more of FEAT_SVE2p1 and FEAT_SME2p1 is implemented, the values `0b0` and `0b1` are permitted.

Otherwise, the only permitted value is `0b0`.

F16F16, bit [42]

Indicates support for SME2.1 non-widening half-precision floating-point instructions. Defined values are:

F16F16	Meaning
`0b0`	SME2.1 non-widening half-precision floating-point instructions are not implemented.
`0b1`	SME2.1 non-widening half-precision floating-point instructions are implemented.

FEAT_SME_F16F16 implements the functionality identified by `0b1`.

If FEAT_SME2p1 is implemented, the values `0b0` and `0b1` are permitted.

Otherwise, the only permitted value is `0b0`.

Bits [41:40]

Reserved, res0.

I8I32, bits [39:36]

Indicates SME support for instructions that accumulate 8-bit integer outer products into 32-bit integer tiles. Defined values are:

I8I32	Meaning
`0b0000`	Instructions that accumulate 8-bit outer products into 32-bit tiles are not implemented.
`0b1111`	The SMOPA, SMOPS, SUMOPA, SUMOPS, UMOPA, UMOPS, USMOPA, and USMOPS instructions that accumulate 8-bit outer products into 32-bit tiles are implemented.

All other values are reserved.

If FEAT_SME is implemented, the only permitted value is `0b1111`.

F16F32, bit [35]

Indicates SME support for instructions that accumulate FP16 half-precision floating-point outer products into FP32 single-precision floating-point tiles. Defined values are:

F16F32	Meaning
`0b0`	Instructions that accumulate half-precision outer products into single-precision tiles are not implemented.
`0b1`	The FMOPA and FMOPS instructions that accumulate half-precision outer products into single-precision tiles are implemented.

If FEAT_SME is implemented, the only permitted value is `0b1`.

B16F32, bit [34]

Indicates SME support for instructions that accumulate BFloat16 outer products into FP32 single-precision floating-point tiles. Defined values are:

B16F32	Meaning
`0b0`	Instructions that accumulate BFloat16 outer products into single-precision tiles are not implemented.
`0b1`	The BFMOPA and BFMOPS instructions that accumulate BFloat16 outer products into single-precision tiles are implemented.

If FEAT_SME is implemented, the only permitted value is `0b1`.

BI32I32, bit [33]

Indicates SME support for instructions that accumulate thirty-two 1-bit binary outer products into 32-bit integer tiles. Defined values are:

BI32I32	Meaning

0b0	Instructions that accumulate 1-bit binary outer products into 32-bit integer tiles are not implemented.
0b1	The BMOPA and BMOPS instructions that accumulate 1-bit binary outer products into 32-bit integer tiles are implemented.

If FEAT_SME2 is implemented, the only permitted value is 0b1. Otherwise, the only permitted value is 0b0.

F32F32, bit [32]

Indicates SME support for instructions that accumulate FP32 single-precision floating-point outer products into single-precision floating-point tiles. Defined values are:

F32F32	Meaning
0b0	Instructions that accumulate single-precision outer products into single-precision tiles are not implemented.
0b1	The FMOPA and FMOPS instructions that accumulate single-precision outer products into single-precision tiles are implemented.

If FEAT_SME is implemented, the only permitted value is 0b1.

Bits [31:0]

Reserved, res0.

Accessing ID_AA64SMFR0_EL1

This register is read-only and can be accessed from EL1 and higher.

This register is only accessible from the AArch64 state.

Accesses to this register use the following encodings in the System register encoding space:

```
MRS <Xt>, ID_AA64SMFR0_EL1
```

op0	op1	CRn	CRm	op2

0b11	0b000	0b0000	0b0100	0b101

```
if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_AA64SMFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64SMFR0_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64SMFR0_EL1;
```

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94