ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1

The ID_AA64ISAR1_EL1 characteristics are:

Purpose

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

There are no configuration notes.

Attributes

ID_AA64ISAR1_EL1 is a 64-bit register.

Field descriptions

63:60	59:56	55:52	51:48	47:44	43:40	39:36	35:32
LS64	XS	I8MM	DGH	BF16	SPECRES	SB	FRINTTS

31:28	27:24	23:20	19:16	15:12	11:8	7:4	3:0
GPI	GPA	LRCPC	FCMA	JSCVT	API	APA	DPB

LS64, bits [63:60]

Indicates support for LD64B and ST64B* instructions, and the ACCDATA_EL1 register. Defined values of this field are:

LS64	Meaning
0b0000	The LD64B, ST64B, ST64BV, and ST64BV0 instructions, the ACCDATA_EL1 register, and associated traps are not supported.
0b0001	The LD64B and ST64B instructions are supported.
0b0010	The LD64B, ST64B, and ST64BV instructions, and their associated traps are supported.

0b0011	The LD64B, ST64B, ST64BV, and ST64BV0 instructions, the ACCDATA_EL1 register, and their associated traps are supported.

All other values are reserved.

FEAT_LS64 implements the functionality identified by 0b0001.

FEAT_LS64_V implements the functionality identified by 0b0010.

FEAT_LS64_ACCDATA implements the functionality identified by 0b0011.

From Armv8.7, the permitted values are 0b0000, 0b0001, 0b0010, and 0b0011.

XS, bits [59:56]

Indicates support for the XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the HCRX_EL2.{FGTnXS, FnXS} fields in AArch64 state. Defined values are:

XS	Meaning
0b0000	The XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the HCRX_EL2.{FGTnXS, FnXS} fields are not supported.
0b0001	The XS attribute, the TLBI and DSB instructions with the nXS qualifier, and the HCRX_EL2.{FGTnXS, FnXS} fields are supported.

All other values are reserved.

FEAT_XS implements the functionality identified by 0b0001.

From Armv8.7, the only permitted value is 0b0001.

I8MM, bits [55:52]

Indicates support for Advanced SIMD and Floating-point Int8 matrix multiplication instructions in AArch64 state. Defined values are:

I8MM	Meaning
0b0000	Int8 matrix multiplication instructions are not implemented.

0b0001	SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.

All other values are reserved.

FEAT_I8MM implements the functionality identified by `0b0001`.

When Advanced SIMD and SVE are both implemented, this field must return the same value as ID_AA64ZFR0_EL1.I8MM.

From Armv8.6, the only permitted value is `0b0001`.

DGH, bits [51:48]

Indicates support for the Data Gathering Hint instruction. Defined values are:

DGH	Meaning
0b0000	Data Gathering Hint is not implemented.
0b0001	Data Gathering Hint is implemented.

All other values are reserved.

FEAT_DGH implements the functionality identified by `0b0001`.

From Armv8.0, the permitted values are `0b0000` and `0b0001`.

If the `DGH` instruction has no effect in preventing the merging of memory accesses, the value of this field is `0b0000`.

BF16, bits [47:44]

Indicates support for Advanced SIMD and Floating-point BFloat16 instructions in AArch64 state. Defined values are:

BF16	Meaning
0b0000	BFloat16 instructions are not implemented.
0b0001	BFCVT, BFCVTN, BFCVTN2, BFDOT, BFMLALB, BFMLALT, and BFMMLA instructions are implemented.
0b0010	As `0b0001`, but the FPCR.EBF field is also supported.

All other values are reserved.

FEAT_BF16 adds the functionality identified by `0b0001`.

FEAT_EBF16 adds the functionality identified by `0b0010`.

When FEAT_SVE or FEAT_SME is implemented, this field must return the same value as ID_AA64ZFR0_EL1.BF16.

From Armv8.6 and Armv9.1, the value `0b0000` is not permitted.

SPECRES, bits [43:40]

Indicates support for prediction invalidation instructions in AArch64 state. Defined values are:

SPECRES	Meaning
`0b0000`	Prediction invalidation instructions are not implemented.
`0b0001`	CFP RCTX, DVP RCTX and CPP RCTX instructions are implemented.
`0b0010`	As `0b0001`, and COSP RCTX instruction is implemented.

All other values are reserved.

FEAT_SPECRES implements the functionality identified by `0b0001`.

FEAT_SPECRES2 implements the functionality identified by `0b0010`.

From Armv8.5, the value `0b0000` is not permitted.

From Armv8.9, the value `0b0001` is not permitted.

SB, bits [39:36]

Indicates support for SB instruction in AArch64 state. Defined values are:

SB	Meaning
`0b0000`	SB instruction is not implemented.
`0b0001`	SB instruction is implemented.

All other values are reserved.

FEAT_SB implements the functionality identified by `0b0001`.

In Armv8.0, the permitted values are `0b0000` and `0b0001`.

From Armv8.5, the only permitted value is `0b0001`.

FRINTTS, bits [35:32]

Indicates support for the FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented. Defined values are:

FRINTTS	Meaning
`0b0000`	FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are not implemented.
`0b0001`	FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented.

All other values are reserved.

FEAT_FRINTTS implements the functionality identified by `0b0001`.

From Armv8.5, the only permitted value is `0b0001`.

GPI, bits [31:28]

Indicates support for an implementation defined algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are:

GPI	Meaning
`0b0000`	Generic Authentication using an implementation defined algorithm is not implemented.
`0b0001`	Generic Authentication using an implementation defined algorithm is implemented. This includes the `PACGA` instruction.

All other values are reserved.

FEAT_PACIMP implements the functionality identified by `0b0001`.

From Armv8.3, the permitted values are `0b0000` and `0b0001`.

If the value of ID_AA64ISAR1_EL1.GPA is nonzero, or the value of ID_AA64ISAR2_EL1.GPA3 is nonzero, this field must have the value `0b0000`.

GPA, bits [27:24]

Indicates whether the QARMA5 algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are:

GPA	Meaning
`0b0000`	Generic Authentication using the QARMA5 algorithm is not implemented.
`0b0001`	Generic Authentication using the QARMA5 algorithm is implemented. This includes the `PACGA` instruction.

All other values are reserved.

FEAT_PACQARMA5 implements the functionality identified by `0b0001`.

From Armv8.3, the permitted values are `0b0000` and `0b0001`.

If the value of ID_AA64ISAR1_EL1.GPI is nonzero, or the value of ID_AA64ISAR2_EL1.GPA3 is nonzero, this field must have the value `0b0000`.

LRCPC, bits [23:20]

Indicates support for weaker release consistency, RCpc, based model. Defined values are:

LRCPC	Meaning
`0b0000`	RCpc instructions are not implemented.
`0b0001`	The no offset LDAPR, LDAPRB, and LDAPRH instructions are implemented.
`0b0010`	As `0b0001`, and the LDAPR (unscaled immediate) and STLR (unscaled immediate) instructions are implemented.
`0b0011`	As `0b0010`, and the post-index LDAPR, LDIAPP, STILP, and pre-index STLR instructions are implemented. If Advanced SIMD and floating-point is implemented, then the LDAPUR (SIMD&FP), LDAP1 (SIMD&FP), STLUR (SIMD&FP), and STL1 (SIMD&FP) instructions are implemented in Advanced SIMD and floating-point.

All other values are reserved.

FEAT_LRCPC implements the functionality identified by the value `0b0001`.

FEAT_LRCPC2 implements the functionality identified by the value `0b0010`.

FEAT_LRCPC3 implements the functionality identified by the value `0b0011`.

From Armv8.3, the value `0b0000` is not permitted.

From Armv8.4, the value `0b0001` is not permitted.

FCMA, bits [19:16]

Indicates support for complex number addition and multiplication, where numbers are stored in vectors. Defined values are:

FCMA	Meaning
`0b0000`	The FCMLA and FCADD instructions are not implemented.
`0b0001`	The FCMLA and FCADD instructions are implemented.

All other values are reserved.

FEAT_FCMA implements the functionality identified by the value `0b0001`.

In Armv8.0, Armv8.1, and Armv8.2, the only permitted value is `0b0000`.

From Armv8.3, if Advanced SIMD or Floating-point is implemented, the only permitted value is `0b0001`.

From Armv8.3, if Advanced SIMD or Floating-point is not implemented, the only permitted value is `0b0000`.

JSCVT, bits [15:12]

Indicates support for JavaScript conversion from double precision floating point values to integers in AArch64 state. Defined values are:

JSCVT	Meaning
`0b0000`	The FJCVTZS instruction is not implemented.
`0b0001`	The FJCVTZS instruction is implemented.

All other values are reserved.

FEAT_JSCVT implements the functionality identified by `0b0001`.

In Armv8.0, Armv8.1, and Armv8.2, the only permitted value is `0b0000`.

From Armv8.3, if Advanced SIMD or Floating-point is implemented, the only permitted value is `0b0001`.

From Armv8.3, if Advanced SIMD or Floating-point is not implemented, the only permitted value is `0b0000`.

API, bits [11:8]

Indicates whether an implementation defined algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the `PACGA` instruction. Defined values are:

API	Meaning
`0b0000`	Address Authentication using an implementation defined algorithm is not implemented.
`0b0001`	Address Authentication using an implementation defined algorithm is implemented, with the HaveEnhancedPAC() and HaveEnhancedPAC2() functions returning FALSE.
`0b0010`	Address Authentication using an implementation defined algorithm is implemented, with the HaveEnhancedPAC() function returning TRUE, and the HaveEnhancedPAC2() function returning FALSE.
`0b0011`	Address Authentication using an implementation defined algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.

`0b0100`	Address Authentication using an implementation defined algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning FALSE, and the HaveEnhancedPAC() function returning FALSE.
`0b0101`	Address Authentication using an implementation defined algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.

All other values are reserved.

FEAT_PAuth implements the functionality identified by `0b0001`.

FEAT_EPAC implements the functionality identified by `0b0010`.

FEAT_PAuth2 implements the functionality identified by `0b0011`.

FEAT_FPAC implements the functionality identified by `0b0100`.

FEAT_FPACCOMBINE implements the functionality identified by `0b0101`.

When this field is nonzero, FEAT_PACIMP is implemented.

In Armv8.3, the permitted values are `0b0001`, `0b0010`, `0b0011`, `0b0100`, and `0b0101`.

From Armv8.6, the permitted values are `0b0011`, `0b0100`, and `0b0101`.

If the value of ID_AA64ISAR1_EL1.APA is nonzero, or the value of ID_AA64ISAR2_EL1.APA3 is nonzero, this field must have the value `0b0000`.

APA, bits [7:4]

Indicates whether the QARMA5 algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the `PACGA` instruction.

Defined values are:

APA	Meaning
`0b0000`	Address Authentication using the QARMA5 algorithm is not implemented.
`0b0001`	Address Authentication using the QARMA5 algorithm is implemented, with the HaveEnhancedPAC() and HaveEnhancedPAC2() functions returning FALSE.
`0b0010`	Address Authentication using the QARMA5 algorithm is implemented, with the HaveEnhancedPAC() function returning TRUE and the HaveEnhancedPAC2() function returning FALSE.
`0b0011`	Address Authentication using the QARMA5 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning FALSE, the HaveFPACCombined() function returning FALSE, and the HaveEnhancedPAC() function returning FALSE.
`0b0100`	Address Authentication using the QARMA5 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning FALSE, and the HaveEnhancedPAC() function returning FALSE.

0b0101	Address Authentication using the QARMA5 algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.

All other values are reserved.

FEAT_PAuth implements the functionality identified by `0b0001`.

FEAT_EPAC implements the functionality identified by `0b0010`.

FEAT_PAuth2 implements the functionality identified by `0b0011`.

FEAT_FPAC implements the functionality identified by `0b0100`.

FEAT_FPACCOMBINE implements the functionality identified by `0b0101`.

When this field is nonzero, FEAT_PACQARMA5 is implemented.

In Armv8.3, the permitted values are `0b0001`, `0b0010`, `0b0011`, `0b0100`, and `0b0101`.

From Armv8.6, the permitted values are `0b0011`, `0b0100`, and `0b0101`.

If the value of ID_AA64ISAR1_EL1.API is nonzero, or the value of ID_AA64ISAR2_EL1.APA3 is nonzero, this field must have the value `0b0000`.

DPB, bits [3:0]

Data Persistence writeback. Indicates support for the DC CVAP and DC CVADP instructions in AArch64 state. Defined values are:

DPB	Meaning
0b0000	DC CVAP not supported.
0b0001	DC CVAP supported.
0b0010	DC CVAP and DC CVADP supported.

All other values are reserved.

FEAT_DPB implements the functionality identified by the value `0b0001`.

FEAT_DPB2 implements the functionality identified by the value `0b0010`.

In Armv8.2, the permitted values are `0b0001` and `0b0010`.

From Armv8.5, the only permitted value is `0b0010`.

Accessing ID_AA64ISAR1_EL1

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

```
MRS <Xt>, ID_AA64ISAR1_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b001

```
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_AA64ISAR1_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_AA64ISAR1_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_AA64ISAR1_EL1;
```