ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4

The ID_MMFR4_EL1 characteristics are:

Purpose

Provides information about the implemented memory model and memory management support in AArch32 state.

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

Configuration

AArch64 System register ID_MMFR4_EL1 bits [31:0] are architecturally mapped to AArch32 System register ID_MMFR4[31:0].

Attributes

ID_MMFR4_EL1 is a 64-bit register.

Field descriptions

When AArch32 is supported:

63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32
RES0

EVT	CCIDX	LSM	HPDS	CnP	XNX	AC2	SpecSEI
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

Bits [63:32]

Reserved, res0.

EVT, bits [31:28]

Enhanced Virtualization Traps. If EL2 is implemented, indicates support for the HCR2.{TTLBIS, TOCU, TICAB, TID4} traps. Defined values are:

EVT	Meaning
0b0000	HCR2.{TTLBIS, TOCU, TICAB, TID4} traps are not supported.

`0b0001`	HCR2.{TOCU, TICAB, TID4} traps are supported. HCR2.TTLBIS trap is not supported.
`0b0010`	HCR2.{TTLBIS, TOCU, TICAB, TID4} traps are supported.

All other values are reserved.

FEAT_EVT implements the functionality identified by the values `0b0001` and `0b0010`.

If EL2 is not implemented supporting AArch32, the only permitted value is `0b0000`.

In Armv8.2, the permitted values are `0b0000`, `0b0001`, and `0b0010`.

From Armv8.5, the permitted values are:

- `0b0000` when EL2 is not implemented or does not support AArch32.
- `0b0010` when EL2 is implemented and supports AArch32.

CCIDX, bits [27:24]

Support for use of the revised CCSIDR format and the presence of the CCSIDR2 is indicated. Defined values are:

CCIDX	Meaning
`0b0000`	32-bit format implemented for all levels of the CCSIDR, and the CCSIDR2 register is not implemented.
`0b0001`	64-bit format implemented for all levels of the CCSIDR, and the CCSIDR2 register is implemented.

All other values are reserved.

FEAT_CCIDX implements the functionality identified by `0b0001`.

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

LSM, bits [23:20]

Indicates support for LSMAOE and nTLSMD bits in HSCTLR and SCTLR. Defined values are:

LSM	Meaning

0b0000	LSMAOE and nTLSMD bits not supported.
0b0001	LSMAOE and nTLSMD bits supported.

All other values are reserved.

FEAT_LSMAOC implements the functionality identified by the value 0b0001.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

HPDS, bits [19:16]

Hierarchical permission disables bits in translation tables. Defined values are:

HPDS	Meaning
0b0000	Disabling of hierarchical controls not supported.
0b0001	Supports disabling of hierarchical controls using the TTBCR2.HPD0, TTBCR2.HPD1, and HTCR.HPD bits.
0b0010	As for value 0b0001, and adds possible hardware allocation of bits[62:59] of the Translation table descriptors from the final lookup level for implementation defined use.

All other values are reserved.

FEAT_AA32HPD implements the functionality identified by the value 0b0001.

FEAT_HPDS2 implements the functionality added by the value 0b0010.

Note

> The value 0b0000 implies that the encoding for TTBCR2 is undefined.

CnP, bits [15:12]

Common not Private translations. Defined values are:

CnP	Meaning

0b0000	Common not Private translations not supported.
0b0001	Common not Private translations supported.

All other values are reserved.

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

From Armv8.2 the only permitted value is `0b0001`.

XNX, bits [11:8]

Support for execute-never control distinction by Exception level at stage 2. Defined values are:

XNX	Meaning
0b0000	Distinction between EL0 and EL1 execute-never control at stage 2 not supported.
0b0001	Distinction between EL0 and EL1 execute-never control at stage 2 supported.

All other values are reserved.

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

When FEAT_XNX is implemented:

- If all of the following conditions are true, it is implementation defined whether the value of ID_MMFR4_EL1.XNX is `0b0000` or `0b0001`:
    - ID_AA64MMFR1_EL1.XNX ==1.
    - EL2 cannot use AArch32.
    - EL1 can use AArch32.
- If EL2 can use AArch32 then the only permitted value is `0b0001`.

AC2, bits [7:4]

Indicates the extension of the ACTLR and HACTLR registers using ACTLR2 and HACTLR2. Defined values are:

AC2	Meaning
0b0000	ACTLR2 and HACTLR2 are not implemented.
0b0001	ACTLR2 and HACTLR2 are implemented.

All other values are reserved.

In Armv8.0 and Armv8.1 the permitted values are `0b0000` and `0b0001`.

From Armv8.2, the only permitted value is `0b0001`.

SpecSEI, bits [3:0]

When FEAT_RAS is implemented:

Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches.

SpecSEI	Meaning
`0b0000`	The PE never generates an SError interrupt due to an External abort on a speculative read.
`0b0001`	The PE might generate an SError interrupt due to an External abort on a speculative read.

All other values are reserved.

Otherwise:

Reserved, res0.

Otherwise:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
UNKNOWN																															
UNKNOWN																															
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

Bits [63:0]

Reserved, unknown.

Accessing ID_MMFR4_EL1

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

MRS <Xt>, ID_MMFR4_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0010	0b110

```
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() &&
(IsFeatureImplemented(FEAT_FGT) || !
IsZero(ID_MMFR4_EL1) || boolean
IMPLEMENTATION_DEFINED "ID_MMFR4_EL1 trapped by
HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = ID_MMFR4_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_MMFR4_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_MMFR4_EL1;
```

AArch32 Registers AArch64 Registers AArch32 Instructions AArch64 Instructions Index by Encoding External Registers

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94