ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2

The ID_ISAR2_EL1 characteristics are:

Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with ID_ISAR0_EL1, ID_ISAR1_EL1, ID_ISAR3_EL1, ID_ISAR4_EL1, and ID_ISAR5_EL1.

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_ISAR2_EL1 bits [31:0] are architecturally mapped to AArch32 System register ID_ISAR2[31:0].

Attributes

ID_ISAR2_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																															
Reversal				PSR_AR				MultU				MultS				Mult				MultiAccessInt				MemHint				LoadStore			
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.

Reversal, bits [31:28]

Indicates the implemented Reversal instructions. Defined values are:

Reversal	Meaning
0b0000	None implemented.

0b0001	Adds the REV, REV16, and REVSH instructions.
0b0010	As for 0b0001, and adds the RBIT instruction.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0010.

PSR_AR, bits [27:24]

Indicates the implemented A and R-profile instructions to manipulate the PSR. Defined values are:

PSR_AR	Meaning
0b0000	None implemented.
0b0001	Adds the MRS and MSR instructions, and the exception return forms of data-processing instructions.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0001.

The exception return forms of the data-processing instructions are:

- In the A32 instruction set, data-processing instructions with the PC as the destination and the S bit set. These instructions might be affected by the WithShifts attribute.
- In the T32 instruction set, the SUBS PC,LR,#N instruction.

MultU, bits [23:20]

Indicates the implemented advanced unsigned Multiply instructions. Defined values are:

MultU	Meaning
0b0000	None implemented.
0b0001	Adds the UMULL and UMLAL instructions.
0b0010	As for 0b0001, and adds the UMAAL instruction.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0010.

MultS, bits [19:16]

Indicates the implemented advanced signed Multiply instructions. Defined values are:

MultS	Meaning
0b0000	None implemented.
0b0001	Adds the SMULL and SMLAL instructions.
0b0010	As for 0b0001, and adds the SMLABB, SMLABT, SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLATB, SMLATT, SMLAWB, SMLAWT, SMULBB, SMULBT, SMULTB, SMULTT, SMULWB, and SMULWT instructions. Also adds the Q bit in the PSRs.
0b0011	As for 0b0010, and adds the SMLAD, SMLADX, SMLALD, SMLALDX, SMLSD, SMLSDX, SMLSLD, SMLSLDX, SMMLA, SMMLAR, SMMLS, SMMLSR, SMMUL, SMMULR, SMUAD, SMUADX, SMUSD, and SMUSDX instructions.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0011.

Mult, bits [15:12]

Indicates the implemented additional Multiply instructions. Defined values are:

Mult	Meaning
0b0000	No additional instructions implemented. This means only MUL is implemented.
0b0001	Adds the MLA instruction.
0b0010	As for 0b0001, and adds the MLS instruction.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0010.

MultiAccessInt, bits [11:8]

Indicates the support for interruptible multi-access instructions. Defined values are:

MultiAccessInt	Meaning
0b0000	No support. This means the LDM and STM instructions are not interruptible.
0b0001	LDM and STM instructions are restartable.
0b0010	LDM and STM instructions are continuable.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0000.

MemHint, bits [7:4]

Indicates the implemented Memory Hint instructions. Defined values are:

MemHint	Meaning
0b0000	None implemented.
0b0001	Adds the PLD instruction.
0b0010	Adds the PLD instruction. (0b0001 and 0b0010 have identical effects.)
0b0011	As for 0b0001 (or 0b0010), and adds the PLI instruction.
0b0100	As for 0b0011, and adds the PLDW instruction.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0100.

LoadStore, bits [3:0]

Indicates the implemented additional load/store instructions. Defined values are:

LoadStore	Meaning
0b0000	No additional load/store instructions implemented.

0b0001	Adds the LDRD and STRD instructions.
0b0010	As for 0b0001, and adds the Load Acquire (LDAB, LDAH, LDA, LDAEXB, LDAEXH, LDAEX, LDAEXD) and Store Release (STLB, STLH, STL, STLEXB, STLEXH, STLEX, STLEXD) instructions.

All other values are reserved.

In Armv8-A, the only permitted value is 0b0010.

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_ISAR2_EL1

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

MRS <Xt>, ID_ISAR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0010	0b010

```
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_ISAR2_EL1;
```

```
elsif PSTATE.EL == EL2 then
    X[t, 64] = ID_ISAR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ID_ISAR2_EL1;
```

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94