

FAR_EL2, Fault Address Register (EL2)

The FAR_EL2 characteristics are:

Purpose

Holds the faulting Virtual Address for all synchronous Instruction Abort exceptions, Data Abort exceptions, PC alignment fault exceptions and Watchpoint exceptions that are taken to EL2.

Configuration

AArch64 System register FAR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HDFAR\[31:0\]](#).

AArch64 System register FAR_EL2 bits [63:32] are architecturally mapped to AArch32 System register [HIFAR\[31:0\]](#).

AArch64 System register FAR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [DFAR\[31:0\]](#) (DFAR_S) when EL2 is implemented.

AArch64 System register FAR_EL2 bits [63:32] are architecturally mapped to AArch32 System register [IFAR\[31:0\]](#) (IFAR_S) when EL2 is implemented.

If EL2 is not implemented, this register is res0 from EL3.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

FAR_EL2 is a 64-bit register.

Field descriptions

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
Faulting Virtual Address for synchronous exceptions taken to EL2																															
Faulting Virtual Address for synchronous exceptions taken to EL2																															
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]

Faulting Virtual Address for synchronous exceptions taken to EL2.
Exceptions that set the FAR_EL2 are Instruction Aborts (EC 0x20 or

0x21), Data Aborts (EC 0x24 or 0x25), PC alignment faults (EC 0x22), and Watchpoints (EC 0x34 or 0x35). [ESR_EL2](#).EC holds the EC value for the exception.

For a synchronous External abort, if the VA that generated the abort was from an address range for which $\text{TCR_ELx.TBI}\{\text{<0|1>}\} == 1$ for the translation regime in use when the abort was generated, then the top eight bits of FAR_EL2 are unknown.

For a synchronous External abort other than a synchronous External abort on a translation table walk, this field is valid only if [ESR_EL2](#).FnV is 0, and FAR_EL2 is unknown if [ESR_EL2](#).FnV is 1.

If a memory fault that sets FAR_EL2, other than a Tag Check Fault, is generated from a data cache maintenance or other DC instruction, this field holds the address specified in the register argument of the instruction.

On an exception due to a Tag Check Fault caused by a data cache maintenance or other DC instruction, the address held in FAR_EL2 is implementation defined as one of the following:

- The lowest address that gave rise to the fault.
- The address specified in the register argument of the instruction as generated by MMU faults caused by [DC ZVA](#).

If the exception that updates FAR_EL2 is taken from an Exception level using AArch32, the top 32 bits are all zero, unless both of the following apply, in which case the top 32 bits of FAR_ELx are 0x00000001:

- The faulting address was generated by a load or store instruction that sequentially incremented from address 0xFFFFFFFF. Such a load or store instruction is constrained unpredictable.
- The implementation treats such incrementing as setting bit[32] of the virtual address to 1.

When the PE sets [ESR_EL2](#).{ISV,FnP} to {0,1} on taking a Data Abort exception, or sets [ESR_EL2](#).{FnV,FnP} to {0,1} on taking a Watchpoint exception, the PE sets FAR_EL2 to any address within the naturally-aligned fault granule that contains the virtual address of the memory access that generated the Data Abort exception or Watchpoint exception.

The naturally-aligned fault granule is one of:

- When ESR_EL2.DFSC is 0b010001, indicating a Synchronous Tag Check fault, it is a 16-byte tag granule.
- When ESR_EL2.DFSC is 0b11010x, indicating an implementation defined fault, it is an implementation defined granule.
- Otherwise, it is the smallest implemented translation granule.

When FEAT_MOPS is implemented, the value in FAR_EL2 on a synchronous exception from any of the Memory Copy and Memory Set instructions represents the first element that has not been copied or set, and is determined as follows:

- For a Data Abort generated by the MMU, the value is within the address range of the relevant translation granule, aligned to the size of the relevant translation granule of the address that generated the Data Abort. Bits[(n-1):0] of the value are unknown, where 2^n is the relevant translation granule size in bytes. For the purpose of calculating the relevant translation granule, if the MMU is disabled for a stage of translation, then the current translation granule size is equal to 2^{64} for stage 1, and the PARange for stage 2. The relevant translation granule is:
 - For MMU faults generated at stage 1, the current stage 1 translation granule.
 - For MMU faults generated at stage 2, the smaller of the current stage 1 translation granule and the current stage 2 translation granule.
 - If FEAT_RME is implemented, for a synchronous data abort generated as the result of a GPF, the smallest of the current stage 1 translation granule, the current stage 2 translation granule and the configured granule size in [GPCCR_EL3.PGS](#).
- For a Data Abort generated by a Tag Check failure, the value is the lowest address that failed the Tag Check within the block size of the load or store.
- For a Watchpoint exception, the value is an address range of the size defined by the [DCZID_EL0.BS](#) field. This address does not need to be the element with a watchpoint, but can be some earlier element.
- Otherwise, the value is the lowest address in the block size of the load or store.

For a Data Abort exception or Watchpoint exception, if address tagging is enabled for the address accessed by the data access that caused the exception, then this field includes the tag. For more information about address tagging, see 'Address tagging'.

When FEAT_MTE_TAGGED_FAR is not implemented, on a synchronous Tag Check Fault abort, bits[63:60] are unknown.

Execution at EL1 or EL0 makes FAR_EL2 become unknown.

Note

The address held in this field is an address accessed by the instruction fetch or data access that caused the exception that actually gave rise to the instruction or data abort. It is the lower address that gave rise to the fault that is reported. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores an unaligned address that crosses a page boundary, the architecture does not prioritize which fault is reported.

For all other exceptions taken to EL2, FAR_EL2 is unknown.

FAR_EL2 is made unknown on an exception return from EL2.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Accessing FAR_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic FAR_EL2 or FAR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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

MRS <Xt>, FAR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = FAR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = FAR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = FAR_EL2;

```

MSR FAR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0110	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        FAR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    FAR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    FAR_EL2 = X[t, 64];

```

When FEAT_VHE is implemented

MRS <Xt>, FAR_EL1

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

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);

```

```

        elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HFGTR_EL2.FAR_EL1 == '1'
then
    AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        X[t, 64] = NVMem[0x220];
    else
        X[t, 64] = FAR_EL1;
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            X[t, 64] = FAR_EL2;
        else
            X[t, 64] = FAR_EL1;
    elsif PSTATE.EL == EL3 then
        X[t, 64] = FAR_EL1;

```

When FEAT_VHE is implemented

MSR FAR_EL1, <Xt>

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

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HFGWTR_EL2.FAR_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        NVMem[0x220] = X[t, 64];
    else
        FAR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            FAR_EL2 = X[t, 64];
        else
            FAR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        FAR_EL1 = X[t, 64];

```

[AArch32
Registers](#)

[AArch64
Registers](#)

[AArch32
Instructions](#)

[AArch64
Instructions](#)

[Index by
Encoding](#)

[External
Registers](#)

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94

Copyright © 2010-2023 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.