AArch64
Instructions

Index by Encoding

External Registers

# VBAR\_EL2, Vector Base Address Register (EL2)

The VBAR EL2 characteristics are:

## **Purpose**

Holds the vector base address for any exception that is taken to EL2.

### **Configuration**

AArch64 System register VBAR\_EL2 bits [31:0] are architecturally mapped to AArch32 System register <a href="https://example.com/hbar.html">https://example.com/hbar.html</a>.

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**

VBAR\_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

05 02 01 00 55 50 57 50 55 51 55 52 51 50 15 10 17 10 15 11 15	12 11 10 33 30 37 30 33 3 1 33 32				
Vector Base Address					
Vector Base Address	RES0				
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 [63:11]

Vector Base Address. Base address of the exception vectors for exceptions taken to EL2.

#### Note

If FEAT\_LVA3 is implemented:

- If  $\underline{HCR}$   $\underline{EL2}$ . $\underline{E2H}$  == 0b1:
  - If tagged addresses are not being used, bits [63:56] of VBAR\_EL2 must be the same or else the use of the vector address will result in a recursive exception.

- If HCR EL2.E2H == 0b0:
  - If tagged addresses are not being used, bits [63:56] of VBAR\_EL2 must be 0 or else the use of the vector address will result in a recursive exception.

#### Otherwise:

#### If FEAT LVA is implemented:

- If HCR EL2.E2H == 0b1:
  - If tagged addresses are being used, bits [55:52] of VBAR\_EL2 must be the same or else the use of the vector address will result in a recursive exception.
  - If tagged addresses are not being used, bits [63:52] of VBAR\_EL2 must be the same or else the use of the vector address will result in a recursive exception.
- If  $\frac{HCR}{EL2}$ .E2H == 0b0:
  - If tagged addresses are being used, bits [55:52] of VBAR\_EL2 must be 0 or else the use of the vector address will result in a recursive exception.
  - If tagged addresses are not being used, bits [63:52] of VBAR\_EL2 must be 0 or else the use of the vector address will result in a recursive exception.

#### If FEAT LVA is not implemented:

- If HCR EL2.E2H == 0b1:
  - If tagged addresses are being used, bits [55:48] of VBAR\_EL2 must be the same or else the use of the vector address will result in a recursive exception.
  - If tagged addresses are not being used, bits [63:48] of VBAR\_EL2 must be the same or else the use of the vector address will result in a recursive exception.
- If  $\underline{HCR}$   $\underline{EL2}$ . $\underline{E2H}$  == 0b0:
  - If tagged addresses are being used, bits [55:48] of VBAR\_EL2 must be 0 or else the use of the

- vector address will result in a recursive exception.
- If tagged addresses are not being used, bits [63:48] of VBAR\_EL2 must be 0 or else the use of the vector address will result in a recursive exception.

The reset behavior of this field is:

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

#### Bits [10:0]

Reserved, res0.

### **Accessing VBAR EL2**

When <u>HCR\_EL2</u>.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic VBAR\_EL2 or VBAR\_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>, VBAR EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = VBAR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = VBAR_EL2;
```

## MSR VBAR EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0000	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    VBAR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    VBAR_EL2 = X[t, 64];
```

## When FEAT\_VHE is implemented MRS <Xt>, VBAR EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0000	0b000

```
if PSTATE.EL == ELO then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011'
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) |
SCR_EL3.FGTEn == '1') && HFGRTR_EL2.VBAR_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        X[t, 64] = NVMem[0x250];
    else
        X[t, 64] = VBAR\_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = VBAR\_EL2;
    else
        X[t, 64] = VBAR\_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = VBAR\_EL1;
```

## When FEAT\_VHE is implemented MSR VBAR EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b0000	0b000

```
if PSTATE.EL == ELO then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2, NV1, NV> == '011'
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3)
SCR_EL3.FGTEn == '1') && HFGWTR_EL2.VBAR_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR EL2.<NV2,NV1,NV> ==
        NVMem[0x250] = X[t, 64];
    else
        VBAR\_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        VBAR EL2 = X[t, 64];
    else
        VBAR\_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    VBAR\_EL1 = X[t, 64];
```

AArch32AArch64AArch32AArch64Index byExternalRegistersRegistersInstructionsInstructionsEncodingRegisters

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94

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