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 HVBAR[31:0].

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 … 32	Vector Base Address
31 … 11	Vector Base Address
10 … 0	RES0

Bits [63:11]

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

Note

If FEAT_LVA3 is implemented:

- If HCR_EL2.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 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 HCR_EL2.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 HCR_EL2.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 == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011'
then
        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 == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '011'
then
        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> ==
'111' then
        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];
```
