TLBI VALE2IS, TLBI VALE2ISNXS, TLB Invalidate by VA, Last level, EL2, Inner Shareable

The TLBI VALE2IS, TLBI VALE2ISNXS characteristics are:

Purpose

When EL2 is implemented and enabled in the current Security state, invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 64-bit stage 1 translation table entry.

  Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, if TTL[3:2] is `0b00`.

- The entry would be used to translate the specified VA using the EL2 or EL2&0 translation regime, as determined by the current value of the HCR_EL2.E2H bit, for the Security state.

- If HCR_EL2.E2H == 0, the entry is from the final level of the translation table walk.

- If HCR_EL2.E2H == 1, one of the following applies:

  - The entry is a global entry from the final level of the translation table walk.

  - The entry is a non-global entry from the final level of the translation table walk that matches the specified ASID.

The Security state is indicated by the value of SCR_EL3.NS if FEAT_RME is not implemented, or SCR_EL3.{NSE, NS} if FEAT_RME is implemented.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

If FEAT_XS is implemented, the nXS variant of this System instruction is defined.

Both variants perform the same invalidation, but the TLBI System instruction without the nXS qualifier waits for all memory accesses using in-scope old translation information to complete before it is considered complete.

The TLBI System instruction with the nXS qualifier is considered complete when the subset of these memory accesses with XS attribute set to 0 are complete.

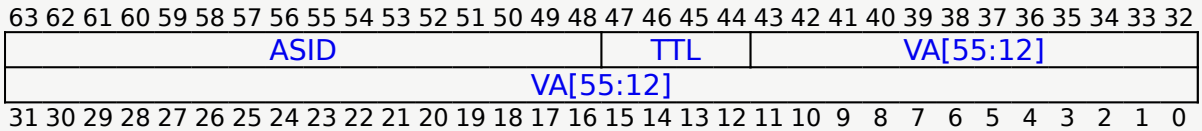
Configuration

There are no configuration notes.

Attributes

TLBI VALE2IS, TLBI VALE2ISNXS is a 64-bit System instruction.

Field descriptions



ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this System instruction.

Global TLB entries that match the VA value will be affected by this System instruction, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, then the upper 8 bits of the ASID must be written to 0 by software when the context being invalidated only uses 8 bits.

TTL, bits [47:44]
When FEAT_TTL is implemented:

Translation Table Level. Indicates the level of the translation table walk that holds the leaf entry for the address being invalidated.

TTL	Meaning
0b00xx	No information supplied as to the translation table level. Hardware must assume that the entry can be from any level. In this case, TTL<1:0> is res0.

`0b01xx`	The entry comes from a 4KB translation granule. The level of walk for the leaf level `0bxx` is encoded as: `0b00` : If FEAT_LPA2 is implemented, level 0. Otherwise, treat as if TTL<3:2> is `0b00`. `0b01` : Level 1. `0b10` : Level 2. `0b11` : Level 3.
`0b10xx`	The entry comes from a 16KB translation granule. The level of walk for the leaf level `0bxx` is encoded as: `0b00` : Reserved. Treat as if TTL<3:2> is `0b00`. `0b01` : If FEAT_LPA2 is implemented, level 1. Otherwise, treat as if TTL<3:2> is `0b00`. `0b10` : Level 2. `0b11` : Level 3.
`0b11xx`	The entry comes from a 64KB translation granule. The level of walk for the leaf level `0bxx` is encoded as: `0b00` : Reserved. Treat as if TTL<3:2> is `0b00`. `0b01` : Level 1. `0b10` : Level 2. `0b11` : Level 3.

If an incorrect value of the TTL field is specified for the entry being invalidated by the instruction, then no entries are required by the architecture to be invalidated from the TLB.

Otherwise:

Reserved, res0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this System instruction.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as res0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are res0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are res0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing TLBI VALE2IS, TLBI VALE2ISNXS

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

```
TLBI VALE2IS{, <Xt>}
```

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0011	0b101

```
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
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_ISH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2),
Regime_EL2, VMID_NONE, Shareability_ISH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2),
```

```
Regime_EL20, VMID_NONE, Shareability_ISH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2),
Regime_EL2, VMID_NONE, Shareability_ISH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
```

TLBI VALE2ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0011	0b101

```
if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif 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
    if HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_ISH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2),
Regime_EL2, VMID_NONE, Shareability_ISH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    elsif HCR_EL2.E2H == '1' then
        AArch64.TLBI_VA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_ISH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_VA(SecurityStateAtEL(EL2),
Regime_EL2, VMID_NONE, Shareability_ISH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
```