TLBIP VAALE1, TLBIP VAALE1NXS, TLB Invalidate Pair by VA, All ASID, Last level, EL1

The TLBIP VAALE1, TLBIP VAALE1NXS characteristics are:

Purpose

Invalidates cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a 128-bit stage 1 translation table entry, from the final level of the translation table walk.

  Or the entry is a 64-bit stage 1 translation table entry, from the final level of the translation table walk, if TTL[3:2] is `0b00`.

- When EL2 is implemented and enabled in the current Security state:

  - If HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID and would be required to translate the specified VA using the EL1&0 translation regime for the Security state.
  - If HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate the specified VA using the EL2&0 translation regime for the Security state.

- When EL2 is not implemented or is disabled in the current Security state, the entry would be required to translate the specified VA using the EL1&0 translation regime for the Security state.

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 the PE that executes this System instruction.

Note

For the EL1&0 and EL2&0 translation regimes, the invalidation applies to both global entries and non-global entries with any ASID.

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.

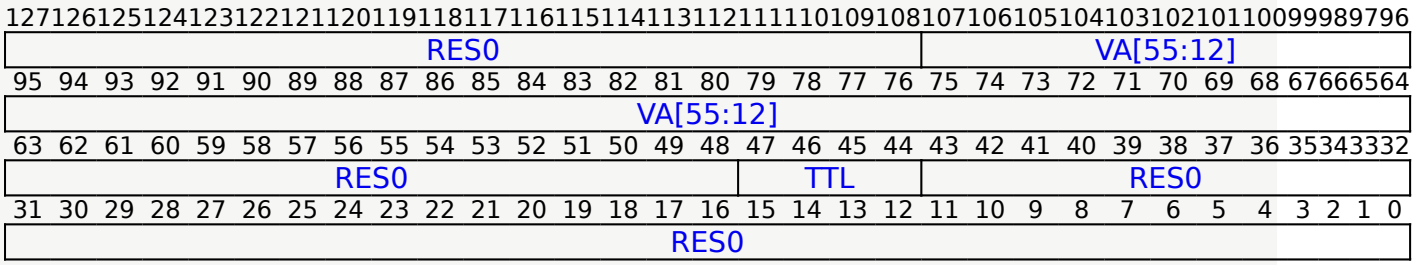
Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP VAALE1, TLBIP VAALE1NXS are undefined.

Attributes

TLBIP VAALE1, TLBIP VAALE1NXS is a 128-bit System instruction.

Field descriptions



Bits [127:108]

Reserved, res0.

VA[55:12], bits [107:64]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this System instruction, regardless of the ASID.

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.

Bits [63:48]

Reserved, res0.

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.

Bits [43:0]

Reserved, res0.

Executing TLBIP VAALE1, TLBIP VAALE1NXS

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

TLBIP VAALE1{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0111	0b111

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HFGITR_EL2.TLBIVAALE1 ==
'1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) &&
IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then

AArch64.TLBIP_VAA(SecurityStateAtEL(EL1),
```

```
Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else

AArch64.TLBIP_VAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        if IsFeatureImplemented(FEAT_XS) &&
IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then

AArch64.TLBIP_VAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
        else

AArch64.TLBIP_VAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL2 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
```

TLBIP VAALE1NXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0111	0b111

```
if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elsif PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
```

```
SCR_EL3.FGTEn == '1') &&
IsFeatureImplemented(FEAT_HCX) && (!
IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIVAALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL2 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
    else
        AArch64.TLBIP_VAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
```

28/03/2023 16:01; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94