
TLBI VMALLE1IS, TLBI VMALLE1ISNXS, TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable

The TLBI VMALLE1IS, TLBI VMALLE1ISNXS characteristics are:

Purpose

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

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- 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 all PEs in the same Inner Shareable shareability domain as the PE that executes this System instruction.

Note

From Armv8.4, when a TLB maintenance instruction is generated to the Secure EL1&0 translation regime and is defined to pass a VMID argument, or would be defined to pass

a VMID argument if [SCR_EL3.EEL2==1](#), then:

- A PE with [SCR_EL3.EEL2==1](#) is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3.EEL2==0](#).
- A PE with [SCR_EL3.EEL2==0](#) is not architecturally required to invalidate any entries in the Secure EL1&0 translation of a PE in the same required shareability domain with [SCR_EL3.EEL2==1](#).
- A PE is architecturally required to invalidate all relevant entries in the Secure EL1&0 translation of a System MMU in the same required shareability domain with a VMID of 0.

Note

For the EL1&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

There are no configuration notes.

Attributes

TLBI VMALLE1IS, TLBI VMALLE1ISNXS is a 64-bit System instruction.

Field descriptions

This instruction has no applicable fields.

The value in the register specified by <Xt> is ignored.

Executing TLBI VMALLE1IS, TLBI VMALLE1ISNXS

The Rt field should be set to 0b11111. If the Rt field is not set to 0b11111, it is constrained unpredictable whether:

- The instruction is undefined.
- The instruction behaves as if the Rt field is set to 0b11111.

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

TLBI VMALLE1IS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0011	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
        IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
        SCR_EL3.FGTEn == '1') && HFGITR_EL2.TLBIVMALLE1IS ==
        '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) &&
        IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then

        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1),
        Regime_EL10, VMID[], Shareability_ISH,
        TLBI_ExcludeXS);
        else

        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1),
        Regime_EL10, VMID[], Shareability_ISH, TLBI_AllAttr);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL2),
            Regime_EL20, VMID_NONE, Shareability_ISH,
            TLBI_AllAttr);
        else
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL1),
            Regime_EL10, VMID[], Shareability_ISH, TLBI_AllAttr);
    elsif PSTATE.EL == EL3 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_VMALL(SecurityStateAtEL(EL2),
```

```

Regime_EL20, VMID_NONE, Shareability_ISH,
TLBI_AllAttr);
    else
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH, TLBI_AllAttr);

```

TLBI VMALLE1ISNXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0011	0b000

```

if !IsFeatureImplemented(FEAT_XS) then
    UNDEFINED;
elseif PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.TTLBIS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') &&
IsFeatureImplemented(FEAT_HCX) && (!
IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIVMALLE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBI_ExcludeXS);
elseif PSTATE.EL == EL2 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_ISH,
TLBI_ExcludeXS);
    else
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBI_ExcludeXS);
elseif PSTATE.EL == EL3 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_ISH,
TLBI_ExcludeXS);
    else
        AArch64.TLBI_VMALL(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBI_ExcludeXS);

```

[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.