
TLBI RVAAE1IS, TLBI RVAAE1ISNXS, TLB Range Invalidate by VA, All ASID, EL1, Inner Shareable

The TLBI RVAAE1IS, TLBI RVAAE1ISNXS characteristics are:

Purpose

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, from any level of the translation table walk up to the level indicated in the TTL hint.

Or if FEAT_D128 is implemented, and the entry is a 128-bit stage 1 translation table entry, from any level of the translation table walk, if TTL is 0b00.

- The entry is within the address range determined by the formula $[\text{BaseADDR} \leq \text{VA} < \text{BaseADDR} + ((\text{NUM} + 1) * 2^{(5 * \text{SCALE} + 1)} * \text{Translation_Granule_Size})]$.
- 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 any of the VAs in the specified address range 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 any of the VAs in the specified address range 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 any of the VAs in the specified address range 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

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

For 64-bit translation table entry, the range of addresses invalidated is unpredictable when:

- For the 4K translation granule:
 - If $TTL==01$ and $BaseADDR[29:12]$ is not equal to 000000000000000000.
 - If $TTL==10$ and $BaseADDR[20:12]$ is not equal to 000000000.
- For the 16K translation granule:
 - If $TTL==10$ and $BaseADDR[24:14]$ is not equal to 000000000000.

For the 64K translation granule:

- - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000000000.
 - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000000.

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_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVAAE1IS, TLBI RVAAE1ISNXS are undefined.

Attributes

TLBI RVAAE1IS, TLBI RVAAE1ISNXS is a 64-bit System instruction.

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 | | | |
| RES0 | | | | | | | | | | | | | | | | TG | SCALE | NUM | | | TTL | BaseADDR | | | | | | | | | | | | |
| BaseADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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 [63:48]

Reserved, res0.

TG, bits [47:46]

Translation granule size.

| TG | Meaning |
|------|--------------------------|
| 0b00 | Reserved. |
| 0b01 | 4K translation granule. |
| 0b10 | 16K translation granule. |
| 0b11 | 64K translation granule. |

The instruction takes a translation granule size for the translations that are being invalidated. If the translations used a different

translation granule size than the one being specified, then the architecture does not require that the instruction invalidates any entries.

SCALE, bits [45:44]

The exponent element of the calculation that is used to produce the upper range.

NUM, bits [43:39]

The base element of the calculation that is used to produce the upper range.

TTL, bits [38:37]

TTL Level hint. The TTL hint is only guaranteed to invalidate:

- Non-leaf-level entries in the range up to but not including the level described by the TTL hint.
- Leaf-level entries in the range that match the level described by the TTL hint.

| TTL | Meaning |
|------------|--|
| 0b00 | The entries in the range can be using any level for the translation table entries. |
| 0b01 | The TTL hint indicates level 1. If FEAT_LPA2 is not implemented, when using a 16KB translation granule, this value is reserved and hardware should treat this field as 0b00. |
| 0b10 | The TTL hint indicates level 2. |
| 0b11 | The TTL hint indicates level 3. |

BaseADDR, bits [36:0]

When FEAT_LPA2 is implemented and TCR_EL1.DS == 1:

The starting address for the range of the maintenance instructions. This field is BaseADDR[52:16] for all translation granules.

When using a 4KB translation granule, BaseADDR[15:12] is treated as 0b0000.

When using a 16KB translation granule, BaseADDR[15:14] is treated as 0b00.

Otherwise:

The starting address for the range of the maintenance instruction.

When using a 4KB translation granule, this field is BaseADDR[48:12].

When using a 16KB translation granule, this field is BaseADDR[50:14].

When using a 64KB translation granule, this field is BaseADDR[52:16].

Executing TLBI RVAAE1IS, TLBI RVAAE1ISNXS

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

TLBI RVAAE1IS{, <Xt>}

| op0 | op1 | CRn | CRm | op2 |
|------|-------|--------|--------|-------|
| 0b01 | 0b000 | 0b1000 | 0b0010 | 0b011 |

```
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.TLBIRVAAE1IS ==
        '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        if IsFeatureImplemented(FEAT_XS) &&
        IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
        && HCRX_EL2.FnXS == '1' then

        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1),
        Regime_EL10, VMID[], Shareability_ISH,
        TLBI_Level_Any, TLBI_ExcludeXS, X[t, 64]);
        else

        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1),
        Regime_EL10, VMID[], Shareability_ISH,
        TLBI_Level_Any, TLBI_AllAttr, X[t, 64]);
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.<E2H,TGE> == '11' then
            AArch64.TLBI_RVAA(SecurityStateAtEL(EL2),
```

```

Regime_EL20, VMID_NONE, Shareability_ISH,
TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
else
    AArch64.TLBI_RVAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_ISH,
TLBILevel_Any, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Any, TLBI_AllAttr, X[t, 64]);

```

TLBI RVAAE1ISNXS{, <Xt>}

| op0 | op1 | CRn | CRm | op2 |
|------|-------|--------|--------|-------|
| 0b01 | 0b000 | 0b1001 | 0b0010 | 0b011 |

```

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.TLBIRVAAE1IS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
elseif PSTATE.EL == EL2 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_ISH,
TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
elseif PSTATE.EL == EL3 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL2),

```

```
Regime_EL20, VMID_NONE, Shareability_ISH,  
TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);  
    else  
        AArch64.TLBI_RVAA(SecurityStateAtEL(EL1),  
Regime_EL10, VMID[], Shareability_ISH,  
TLBILevel_Any, TLBI_ExcludeXS, X[t, 64]);
```

[AArch32
Registers](#)

[AArch64
Registers](#)

[AArch32
Instructions](#)

[AArch64
Instructions](#)

[Index by
Encoding](#)

[External
Registers](#)

28/03/2023 16:01; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94

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