TLBI RVALE1, TLBI RVALE1NXS, TLB Range Invalidate by VA, Last level, EL1

The TLBI RVALE1, TLBI RVALE1NXS 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, if TTL is `0b00`.

- The entry would be used to translate any of the VAs in the specified address range, and one of the following applies:
  - The entry is a global entry from the final level of lookup.
  - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- The entry is within the address range determined by the formula [BaseADDR <= VA < BaseADDR + ((NUM +1)*$2^{(5*SCALE +1)}$ * 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 the PE that executes this System instruction.

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 00000000000.
- For the 64K translation granule:
  - If TTL==01 and BaseADDR[41:16] is not equal to 00000000000000000000000000.
  - If TTL==10 and BaseADDR[28:16] is not equal to 0000000000000.

For more information about the architectural requirements for this System instruction, see 'Invalidation of TLB entries from stage 2 translations'.

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.

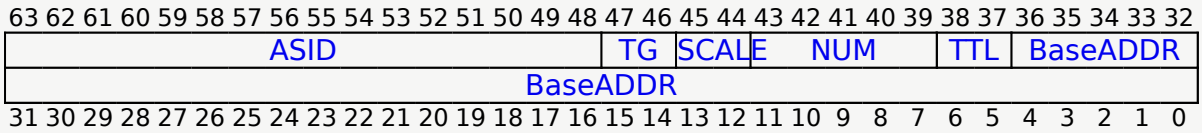
Configuration

This instruction is present only when FEAT_TLBIRANGE is implemented. Otherwise, direct accesses to TLBI RVALE1, TLBI RVALE1NXS are undefined.

Attributes

TLBI RVALE1, TLBI RVALE1NXS 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.

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 RVALE1, TLBI RVALE1NXS

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

TLBI RVALE1{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1000	0b0110	0b101

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TTLB == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HFGITR_EL2.TLBIRVALE1 ==
'1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        if IsFeatureImplemented(FEAT_XS) &&
IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        if IsFeatureImplemented(FEAT_XS) &&
IsFeatureImplemented(FEAT_HCX) && IsHCRXEL2Enabled()
&& HCRX_EL2.FnXS == '1' then
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
        else
            AArch64.TLBI_RVA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL2 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_AllAttr, X[t, 64]);
```

TLBI RVALE1NXS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b000	0b1001	0b0110	0b101

```
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, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') &&
IsFeatureImplemented(FEAT_HCX) && (!
IsHCRXEL2Enabled() || HCRX_EL2.FGTnXS == '0') &&
HFGITR_EL2.TLBIRVALE1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FB == '1' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_ISH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL2 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
elsif PSTATE.EL == EL3 then
    if HCR_EL2.<E2H,TGE> == '11' then
        AArch64.TLBI_RVA(SecurityStateAtEL(EL2),
Regime_EL20, VMID_NONE, Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
    else
        AArch64.TLBI_RVA(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_NSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t, 64]);
```