TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS, TLB Invalidate Pair by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

The TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS characteristics are:

Purpose

If 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 128-bit stage 2 only translation table entry, from the final level of the translation table walk.

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

- If FEAT_RME is implemented, one of the following applies:
  - SCR_EL3.{NSE, NS} is {0, 0} and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
  - SCR_EL3.{NSE, NS} is {0, 1} and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.
  - SCR_EL3.{NSE, NS} is {1, 1} and the entry would be required to translate the specified IPA using the Realm EL1&0 translation regime.

- If FEAT_RME is not implemented, one of the following applies:
  - SCR_EL3.NS is 0 and the entry would be required to translate the specified IPA using the Secure EL1&0 translation regime.
  - SCR_EL3.NS is 1 and the entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.

- The entry would be used with the current VMID.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

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

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.

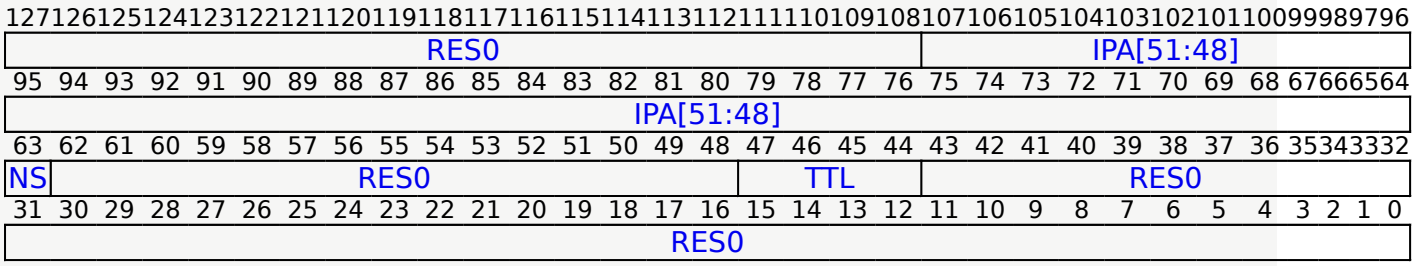
Configuration

This instruction is present only when FEAT_D128 is implemented. Otherwise, direct accesses to TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS are undefined.

Attributes

TLBIP IPAS2LE1OS, TLBIP IPAS2LE1OSNXS is a 128-bit System instruction.

Field descriptions



Bits [127:108]

Reserved, res0.

IPA[51:48], bits [107:64]

Bits[55:12] of the intermediate physical address to match.

NS, bit [63]
When FEAT_RME is implemented:

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 0}, NS selects the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {1, 1}, this field is res0, and the instruction applies only to the Realm IPA space.

When the instruction is executed and SCR_EL3.{NSE, NS} == {0, 1}, this field is res0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is implemented and FEAT_RME is not implemented:

Not Secure. Specifies the IPA space.

NS	Meaning
0b0	IPA is in the Secure IPA space.
0b1	IPA is in the Non-secure IPA space.

When the instruction is executed in Non-secure state, this field is res0, and the instruction applies only to the Non-secure IPA space.

When FEAT_SEL2 is not implemented, or if EL2 is disabled in the current Security state, this field is res0.

Otherwise:

Reserved, res0.

Bits [62:48]

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 IPAS2LE1OS, TLBIP IPAS2LE1OSNXS

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

TLBIP IPAS2LE1OS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1000	0b0100	0b100

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x14);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_OSH,
TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_OSH,
TLBILevel_Last, TLBI_AllAttr, X[t2, 64]:X[t, 64]);
```

TLBIP IPAS2LE1OSNXS{, <Xt>, <Xt2>}

op0	op1	CRn	CRm	op2
0b01	0b100	0b1001	0b0100	0b100

```
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, 0x14);
    else
```

```
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_OSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        return;
    else
        AArch64.TLBIP_IPAS2(SecurityStateAtEL(EL1),
Regime_EL10, VMID[], Shareability_OSH,
TLBILevel_Last, TLBI_ExcludeXS, X[t2, 64]:X[t, 64]);
```

28/03/2023 16:01; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94

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