

IC IVAU, Instruction Cache line Invalidate by VA to PoU

The IC IVAU characteristics are:

Purpose

Invalidate instruction cache by address to Point of Unification.

Configuration

AArch64 System instruction IC IVAU performs the same function as AArch32 System instruction [ICIMVAU](#).

Attributes

IC IVAU 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
Virtual address to use																															
Virtual address to use																															
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 [63:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing IC IVAU

If EL0 access is enabled, when executed at EL0, the instruction may generate a Permission fault, subject to the constraints described in 'MMU faults generated by cache maintenance operations'.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The data cache maintenance instruction (DC)'.

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

IC IVAU{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b011	0b0111	0b0101	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11')
    && SCTLR_EL1.UCI == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11'
    && HCR_EL2.TPU == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11'
    && HCR_EL2.TOCU == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11'
    && IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3)
    || SCR_EL3.FGTEn == '1') && HFGITR_EL2.ICIVAU == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
    && SCTLR_EL2.UCI == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.IC(X[t, 64], CacheOpScope_PoU);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.TOCU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() &&
        IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
        SCR_EL3.FGTEn == '1') && HFGITR_EL2.ICIVAU == '1'
        then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.IC(X[t, 64], CacheOpScope_PoU);
    elsif PSTATE.EL == EL2 then
        AArch64.IC(X[t, 64], CacheOpScope_PoU);
    elsif PSTATE.EL == EL3 then
        AArch64.IC(X[t, 64], CacheOpScope_PoU);

```

[AArch32
Registers](#)

[AArch64
Registers](#)

[AArch32
Instructions](#)

[AArch64
Instructions](#)

[Index by
Encoding](#)

[External
Registers](#)

