

DC CISW, Data or unified Cache line Clean and Invalidate by Set/Way

The DC CISW characteristics are:

Purpose

Clean and Invalidate data cache by set/way.

Configuration

AArch64 System instruction DC CISW performs the same function as AArch32 System instruction [DCCISW](#).

Attributes

DC CISW 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																																
SetWay																															Level	RES0
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 [63:32]

Reserved, res0.

SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are res0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$, $L = \text{Log}_2(\text{LINELEN})$, $B = (L + S)$, $S = \text{Log}_2(\text{NSETS})$.

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

Bit [0]

Reserved, res0.

Executing DC CISW

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is constrained unpredictable and one of the following occurs:

- The instruction is undefined.
- The instruction performs cache maintenance on one of:
 - No cache lines.
 - A single arbitrary cache line.
 - Multiple arbitrary cache lines.

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

DC CISW, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b000	0b0111	0b1110	0b010

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TSW == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
        IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
        SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCISW == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.DC(X[t, 64], CacheType_Data,
        CacheOp_CleanInvalidate, CacheOpScope_SetWay);
elsif PSTATE.EL == EL2 then
    AArch64.DC(X[t, 64], CacheType_Data,
    CacheOp_CleanInvalidate, CacheOpScope_SetWay);
elsif PSTATE.EL == EL3 then
    AArch64.DC(X[t, 64], CacheType_Data,
    CacheOp_CleanInvalidate, CacheOpScope_SetWay);
```

[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.