DIT, Data Independent Timing

The DIT characteristics are:

Purpose

Allows access to the Data Independent Timing bit.

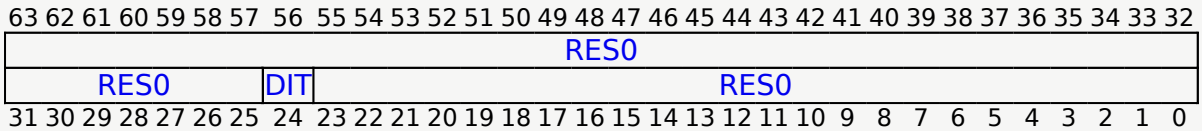
Configuration

This register is present only when FEAT_DIT is implemented. Otherwise, direct accesses to DIT are undefined.

Attributes

DIT is a 64-bit register.

Field descriptions



Bits [63:25]

Reserved, res0.

DIT, bit [24]

Data Independent Timing.

DIT	Meaning
0b0	The architecture makes no statement about the timing properties of any instructions.

`0b1` The architecture requires that:

- The timing of every load and store instruction is insensitive to the value of the data being loaded or stored.
- For certain data processing instructions, the instruction takes a time which is independent of:
  - The values of the data supplied in any of its registers.
  - The values of the NZCV flags.
- For certain data processing instructions, the response of the instruction to asynchronous exceptions does not vary based on:
  - The values of the data supplied in any of its registers.
  - The values of the NZCV flags.

The data processing instructions affected by this bit are:

- All cryptographic instructions. These instructions are:
  - `AESD`, `AESE`, `AESIMC`, `AESMC`, `SHA1C`, `SHA1H`, `SHA1M`, `SHA1P`, `SHA1SU0`, `SHA1SU1`, `SHA256H`, `SHA256H2`, `SHA256SU0`, `SHA256SU1`, `SHA512H`, `SHA512H2`, `SHA512SU0`, `SHA512SU1`, `EOR3`, `RAX1`, `XAR`, `BCAX`, `SM3SS1`, `SM3TT1A`, `SM3TT1B`, `SM3TT2A`, `SM3TT2B`, `SM3PARTW1`, `SM3PARTW2`, `SM4E`, and `SM4EKEY`.
- A subset of those instructions which use the general-purpose register file. These instructions are:
  - `ADC`, `ADCS`, `ADD`, `ADDS`, `AND`, `ANDS`, `ASR`, `ASRV`, `BFC`, `BFI`, `BFM`, `BFXIL`, `BIC`, `BICS`, `CCMN`, `CCMP`, `CFINV`, `CINC`, `CINV`, `CLS`, `CLZ`, `CMN`, `CMP`, `CNEG`, `CSEL`, `CSET`, `CSETM`, `CSINC`, `CSINV`, `CSNEG`, `EON`, `EOR`, `EXTR`, `LSL`, `LSLV`, `LSR`, `LSRV`, `MADD`, `MNEG`, `MOV`, `MOVK`, `MOVN`, `MOVZ`, `MSUB`, `MUL`, `MVN`, `NEG`, `NEGS`, `NGC`, `NGCS`, `NOP`, `ORN`, `ORR`, `RBIT`, `REV`, `REV16`, `REV32`, `REV64`, `RMIF`, `ROR`, `RORV`, `SBC`, `SBCS`, `SBFIZ`, `SBFM`, `SBFX`, `SETF8`, `SETF16`, `SMADDL`, `SMNEGL`, `SMSUBL`, `SMULH`, `SMULL`, `SUB`, `SUBS`, `SXTB`, `SXTH`, `SXTW`, `TST`, `UBFIZ`, `UBFM`,

UBFX, UMADDL, UMNEGL, UMSUBL, UMULH, UMULL, UXTB, and UXTH.

- If FEAT_CRC32 is implemented, CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX.

- A subset of those instructions which use the SIMD&FP register file. These instructions are:
  - ABS, ADD, ADDHN, ADDHN2, ADDP, ADDV, AND, BIC, BIF, BIT, BSL, CLS, CLZ, CMEQ, CMGE, CMGT, CMHI, CMHS, CMLE, CMLT, CMTST, CNT, DUP, EOR, EXT, FCSEL, INS, MLA, MLS, MOV, MOVI, MUL, MVN, MVNI, NEG, NOT, ORN, ORR, PMUL, PMULL, PMULL2, RADDHN, RADDHN2, RBIT, REV16, REV32, RSHRN, RSHRN2, RSUBHN, RSUBHN2, SABA, SABD, SABAL, SABAL2, SABDL, SABDL2, SADALP, SADDL, SADDL2, SADDLP, SADDLV, SADDW, SADDW2, SHADD, SHL, SHLL, SHLL2, SHRN, SHRN2, SHSUB, SLI, SMAX, SMAXP, SMAXV, SMIN, SMINP, SMINV, SMLAL, SMLAL2, SMLSL, SMLSL2, SMOV, SMULL, SMULL2, SQDMULH (by element), SQDMULH (vector), SQRDMLAH (by element), SQRDMLAH (vector), SQRDMULH (by element), SQRDMULH (vector), SRI, SSHL, SSHLL, SSHLL2, SSHR, SSRA, SSUBL, SSUBL2, SSUBW, SSUBW2, SUB, SUBHN, SUBHN2, SXTL, SXTL2, TBL, TBX, TRN1, TRN2, UABA, UABAL, UABAL2, UABD, UABDL, UABDL2, UADALP, UADDL, UADDL2, UADDLP, UADDLV, UADDW, UADDW2, UHADD, UHSUB, UMAX, UMAXP, UMAXV, UMIN, UMINP, UMINV, UMLAL, UMLAL2, UMLSL, UMOV, UMLSL2, UMULL, UMULL2, USHL, USHLL, USHLL2, USHR, USRA, USUBL, USUBL2, USUBW, USUBW2, UXTL, UXTL2, UZP1, UZP2, XTN, XTN2, ZIP1, and ZIP2.

Note

The architecture makes no statement about the timing properties when the PSTATE.DIT bit is not set. However, it is likely that many of these instructions have timing that is invariant of the data in many situations.

In particular, Arm strongly recommends that the Armv8.3 pointer authentication instructions do not have their timing dependent on the key value used in the pointer authentication in all cases, regardless of the PSTATE.DIT bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0`.

Bits [23:0]

Reserved, res0.

Accessing DIT

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

MRS <Xt>, DIT

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b101

```
if PSTATE.EL == EL0 then
    X[t, 64] = Zeros(39):PSTATE.DIT:Zeros(24);
elsif PSTATE.EL == EL1 then
    X[t, 64] = Zeros(39):PSTATE.DIT:Zeros(24);
elsif PSTATE.EL == EL2 then
    X[t, 64] = Zeros(39):PSTATE.DIT:Zeros(24);
elsif PSTATE.EL == EL3 then
    X[t, 64] = Zeros(39):PSTATE.DIT:Zeros(24);
```

MSR DIT, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b101

```
if PSTATE.EL == EL0 then
    PSTATE.DIT = X[t, 64]<24>;
elsif PSTATE.EL == EL1 then
    PSTATE.DIT = X[t, 64]<24>;
elsif PSTATE.EL == EL2 then
    PSTATE.DIT = X[t, 64]<24>;
elsif PSTATE.EL == EL3 then
    PSTATE.DIT = X[t, 64]<24>;
```

MSR DIT, #<imm>

op0	op1	CRn	op2
0b00	0b011	0b0100	0b010

28/03/2023 16:01; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94