ESR_EL1, Exception Syndrome Register (EL1)

The ESR_EL1 characteristics are:

Purpose

Holds syndrome information for an exception taken to EL1.

Configuration

AArch64 System register ESR_EL1 bits [31:0] are architecturally mapped to AArch32 System register DFSR[31:0].

Attributes

ESR_EL1 is a 64-bit register.

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	ISS2

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
EC	IL	ISS

ESR_EL1 is made unknown as a result of an exception return from EL1.

When an unpredictable instruction is treated as undefined, and the exception is taken to EL1, the value of ESR_EL1 is unknown. The value written to ESR_EL1 must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not unpredictable at that Exception level, in order to avoid the possibility of a privilege violation.

Bits [63:56]

Reserved, res0.

ISS2, bits [55:32]

ISS2 encoding for an exception, the bit assignments are:

ISS2 encoding for an exception from a Data Abort (EC == 0b100100 or EC == 0b100101)

23 22 21 20 19 18 17 16 15 14 13 12 11	10	9	8	7	6	5	4 3 2 1 0
RES0	TnD	TagAccess	GCS	RES0	Overlay	DirtyBit	Xs

Bits [23:11]

Reserved, res0.

TnD, bit [10]
When FEAT_MTE_CANONICAL_TAGS is implemented:

Tag not Data.

If a memory access generates a Data Abort for a stage 1 Permission fault, this field indicates whether the fault is due to an Allocation Tag access.

TnD	Meaning
0b0	Permission fault is not due to an Allocation Tag access
0b1	Permission fault is due to an Allocation Tag access.

For any other fault, this field is res0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

TagAccess, bit [9]
When FEAT_MTE_PERM is implemented:

NoTagAccess fault.

If a memory access generates a Data Abort for a Permission fault, this field indicates whether the fault is due to the NoTagAccess memory attribute.

TagAccess	Meaning
0b0	Permission fault is not due to the NoTagAccess memory attribute.

0b1	Permission fault is due to the NoTagAccess memory attribute.

For any other fault, this field is res0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

GCS, bit [8]
When FEAT_GCS is implemented:

Guarded control stack data access.

If a memory access generates a Data Abort, this field indicates whether the fault is due to a Guarded control stack data access.

GCS	Meaning
0b0	The Data Abort is not due to a Guarded control stack data access.
0b1	The Data Abort is due to a Guarded control stack data access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Bit [7]

Reserved, res0.

Overlay, bit [6]
When FEAT_S1POE is implemented:

Overlay flag.

If a memory access generates a Data Abort for a Permission fault, then this field holds information about the fault.

Overlay	Meaning
0b0	Data Abort is not due to Overlay Permissions.
0b1	Data Abort due to Overlay Permissions.

For any other fault, this field is res0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

DirtyBit, bit [5]
When FEAT_S1PIE is implemented:

DirtyBit flag.

If a write access to memory generates a Data Abort for a Permission fault using Indirect Permission, then this field holds information about the fault.

DirtyBit	Meaning
0b0	Permission Fault is not due to dirty state.
0b1	Permission Fault is due to dirty state.

For any other fault or Access, this field is res0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Xs, bits [4:0]
When FEAT_LS64 is implemented:

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort exception for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

Otherwise, this field is res0.

Otherwise:

Reserved, res0.

ISS2 encoding for an exception from an Instruction Abort (EC == 0b100000 or EC == 0b100001)

23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6	5 4 3 2 1 0
RES0	Overlay	RES0

Bits [23:7]

Reserved, res0.

Overlay, bit [6]
When FEAT_S1POE is implemented or FEAT_S2POE is implemented:

Overlay flag.

If a memory access generates a Instruction Abort for a Permission fault, then this field holds information about the fault.

Overlay	Meaning
`0b0`	Instruction Abort is not due to Overlay Permissions.
`0b1`	Instruction Abort due to Overlay Permissions.

For any other fault, this field is res0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Bits [5:0]

Reserved, res0.

ISS2 encoding for an exception from a Watchpoint exception (EC == 0b110100 or EC == 0b110101)

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0															GCS	RES0							

Bits [23:9]

Reserved, res0.

GCS, bit [8]
When FEAT_GCS is implemented:

Guarded control stack data access.

Indicates that the Watchpoint exception is due to a Guarded control stack data access.

GCS	Meaning
`0b0`	The Watchpoint exception is not due to a Guarded control stack data access.
`0b1`	The Watchpoint exception is due to a Guarded control stack data access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Bits [7:0]

Reserved, res0.

EC, bits [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about.

For each EC value, the table references a subsection that gives information about:

- The cause of the exception, for example the configuration required to enable the trap.
- The encoding of the associated ISS.

Possible values of the EC field are:

EC	Meaning
0b000000	Unknown reason.
0b000001	Trapped WF* instruction execution. Conditional WF* instructions that fail their condition code check cause an exception.
0b000011	Trapped MCR or MRC access with (coproc==0b1111) that is not reported using EC 0b000000.
0b000100	Trapped MCRR or MRRC access with (coproc==0b1111) that is reported using EC 0b000000.
0b000101	Trapped MCR or MRC access with (coproc==0b1110).
0b000110	Trapped LDC or STC access. The only architected uses of these instruction are: • An STC to write data to memory from DBGDTRRXint. • An LDC to read data from memory to DBGDTRTXint.

0b000111	Access to SME, SVE, Advanced SIMD or floating-point functiona trapped by CPACR_EL1.FPEN, CPTR_EL2.FPEN, CPTR_EL2.TFP CPTR_EL3.TFP control. Excludes exceptions resulting from CPACR_EL1 when the value HCR_EL2.TGE is 1, or because SVE or Advanced SIMD and float point are not implemented. These are reported with EC value 0b
0b001010	Trapped execution of an LD64B or ST64B* instruction.
0b001100	Trapped MRRC access with (coproc==0b1110).
0b001101	Branch Target Exception.
0b001110	Illegal Execution state.
0b010001	SVC instruction execution in AArch32 state.
0b010101	SVC instruction execution in AArch64 state.
0b011000	Trapped MSR, MRS or System instruction execution in AArch64 that is not reported using EC 0b000000, 0b000001, or 0b000111. This includes all instructions that cause exceptions that are part encoding space defined in 'System instruction class encoding ov except for those exceptions reported using EC values 0b000000, 0b000001, or 0b000111.

0b010100	Trapped MSRR, MRRS or System instruction execution in AArch state, that is not reported using EC 0b000000.
0b011001	Access to SVE functionality trapped as a result of CPACR_EL1.Z CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ, that is not rep using EC 0b000000.
0b011011	Exception from an access to a TSTART instruction at EL0 when SCTLR_EL1.TME0 == 0, EL0 when SCTLR_EL2.TME0 == 0, at when SCTLR_EL1.TME == 0, at EL2 when SCTLR_EL2.TME == EL3 when SCTLR_EL3.TME == 0.
0b011100	Exception from a Pointer Authentication instruction authenticati failure
0b011101	Access to SME functionality trapped as a result of CPACR_EL1.S CPTR_EL2.SMEN, CPTR_EL2.TSM, CPTR_EL3.ESM, or an atten execution of an instruction that is illegal because of the value of PSTATE.SM or PSTATE.ZA, that is not reported using EC 0b0000
0b100000	Instruction Abort from a lower Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or E errors. Not used for debug-related exceptions.
0b100001	Instruction Abort taken without a change in Exception level. Used for MMU faults generated by instruction accesses and synchronous External aborts, including synchronous parity or E errors. Not used for debug-related exceptions.
0b100010	PC alignment fault exception.

0b100100	Data Abort exception from a lower Exception level. Used for MMU faults generated by data accesses, alignment fau than those caused by Stack Pointer misalignment, and synchron External aborts, including synchronous parity or ECC errors. No for debug-related exceptions.
0b100101	Data Abort exception taken without a change in Exception level. Used for MMU faults generated by data accesses, alignment fau than those caused by Stack Pointer misalignment, and synchron External aborts, including synchronous parity or ECC errors. No for debug-related exceptions.
0b100110	SP alignment fault exception.
0b100111	Memory Operation Exception.
0b101000	Trapped floating-point exception taken from AArch32 state. This EC value is valid if the implementation supports trapping o floating-point exceptions, otherwise it is reserved. Whether a flo point implementation supports trapping of floating-point excepti implementation defined.
0b101100	Trapped floating-point exception taken from AArch64 state. This EC value is valid if the implementation supports trapping o floating-point exceptions, otherwise it is reserved. Whether a flo point implementation supports trapping of floating-point excepti implementation defined.
0b101101	GCS exception.
0b101111	SError exception.
0b110000	Breakpoint exception from a lower Exception level.
0b110001	Breakpoint exception taken without a change in Exception level.

0b110010	Software Step exception from a lower Exception level.
0b110011	Software Step exception taken without a change in Exception le
0b110100	Watchpoint exception from a lower Exception level.
0b110101	Watchpoint exception taken without a change in Exception level
0b111000	BKPT instruction execution in AArch32 state.
0b111100	BRK instruction execution in AArch64 state.
0b111101	PMU exception

All other EC values are reserved by Arm, and:

- Unused values in the range 0b000000 - 0b101100 (0x00 - 0x2C) are reserved for future use for synchronous exceptions.
- Unused values in the range 0b101101 - 0b111111 (0x2D - 0x3F) are reserved for future use, and might be used for synchronous or asynchronous exceptions.

The effect of programming this field to a reserved value is that behavior is constrained unpredictable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

IL, bit [25]

Instruction Length for synchronous exceptions. Possible values of this bit are:

IL	Meaning
`0b0`	16-bit instruction trapped.

`0b1`	32-bit instruction trapped. This value is also used when the exception is one of the following:
	• An SError exception.
	• An Instruction Abort exception.
	• A PC alignment fault exception.
	• An SP alignment fault exception.
	• A Data Abort exception for which the value of the ISV bit is 0.
	• An Illegal Execution state exception.
	• Any debug exception except for Breakpoint instruction exceptions. For Breakpoint instruction exceptions, this bit has its standard meaning:
	  ◦ `0b0`: 16-bit T32 BKPT instruction.
	  ◦ `0b1`: 32-bit A32 BKPT instruction or A64 BRK instruction.
	• An exception reported using EC value `0b000000`.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

ISS, bits [24:0]

Instruction Specific Syndrome. Architecturally, this field can be defined independently for each defined Exception class. However, in practice, some ISS encodings are used for more than one Exception class.

Typically, an ISS encoding has a number of subfields. When an ISS subfield holds a register number, the value returned in that field is the AArch64 view of the register number.

For an exception taken from AArch32 state, see 'Mapping of the general-purpose registers between the Execution states'.

If the AArch32 register descriptor is `0b1111`, then:

- If the instruction that generated the exception was not unpredictable, the field takes the value `0b11111`.
- If the instruction that generated the exception was unpredictable, the field takes an unknown value that must be either:
  - The AArch64 view of the register number of a register that might have been used at the Exception level from which the exception was taken.
  - The value `0b11111`.

ISS encoding for exceptions with an unknown reason

<table>
<tr><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="25">RES0</td></tr>
</table>

Bits [24:0]

Reserved, res0.

When an exception is reported using this EC code the IL field is set to 1.

This EC code is used for all exceptions that are not covered by any other EC value. This includes exceptions that are generated in the following situations:

- The attempted execution of an instruction bit pattern that has no allocated instruction or that is not accessible at the current Exception level and Security state, including:
  - A read access using a System register pattern that is not allocated for reads or that does not permit reads at the current Exception level and Security state.
  - A write access using a System register pattern that is not allocated for writes or that does not permit writes at the current Exception level and Security state.
  - Instruction encodings that are unallocated.
  - Instruction encodings for instructions or System registers that are not implemented in the implementation.
- In Debug state, the attempted execution of an instruction bit pattern that is not accessible in Debug state.
- In Non-debug state, the attempted execution of an instruction bit pattern that is not accessible in Non-debug state.
- In AArch32 state, attempted execution of a short vector floating-point instruction.
- In an implementation that does not include Advanced SIMD and floating-point functionality, an attempted access to Advanced SIMD or floating-point functionality under

conditions where that access would be permitted if that functionality was present. This includes the attempted execution of an Advanced SIMD or floating-point instruction, and attempted accesses to Advanced SIMD and floating-point System registers.

- An exception generated because of the value of one of the SCTLR_EL1.{ITD, SED, CP15BEN} control bits.
- Attempted execution of:
    - An HVC instruction when disabled by HCR_EL2.HCD or SCR_EL3.HCE.
    - An SMC instruction when disabled by SCR_EL3.SMD.
    - An HLT instruction when disabled by EDSCR.HDE.
- Attempted execution of an MSR or MRS instruction to access SP_EL0 when the value of SPSel.SP is 0.
- Attempted execution of an MSR or MRS instruction using a _EL12 register name when HCR_EL2.E2H == 0.
- Attempted execution, in Debug state, of:
    - A DCPS1 instruction when the value of HCR_EL2.TGE is 1 and EL2 is disabled or not implemented in the current Security state.
    - A DCPS2 instruction from EL1 or EL0 when EL2 is disabled or not implemented in the current Security state.
    - A DCPS3 instruction when the value of EDSCR.SDD is 1, or when EL3 is not implemented.
- When EL3 is using AArch64, attempted execution from Secure EL1 of an SRS instruction using R13_mon.
- In Debug state when the value of EDSCR.SDD is 1, the attempted execution at EL2, EL1, or EL0 of an instruction that is configured to trap to EL3.
- In AArch32 state, the attempted execution of an MRS (banked register) or an MSR (banked register) instruction to SPSR_mon, SP_mon, or LR_mon.

- An exception that is taken to EL2 because the value of HCR_EL2.TGE is 1 that, if the value of HCR_EL2.TGE was 0 would have been reported with an ESR_ELx.EC value of `0b000111`.

- In Non-transactional state, attempted execution of a TCOMMIT instruction.

ISS encoding for an exception from a WF* instruction

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
CV	COND	RES0	RN	RES0	RV	TI

CV, bit [24]

Condition code valid.

CV	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is implementation defined whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to `0b1110`.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
    - If the instruction is conditional, COND is set to the condition code field value from the instruction.
    - If the instruction is unconditional, COND is set to `0b1110`.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
    - With COND set to `0b1110`, the value for unconditional.
    - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is implementation defined whether:
    - CV is set to 0 and COND is set to an unknown value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
    - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped

conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is implementation defined whether the COND field is set to `0b1110`, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bits [19:10]

Reserved, res0.

RN, bits [9:5]
When FEAT_WFxT is implemented:

Register Number. Indicates the register number supplied for a `WFET` or `WFIT` instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Bits [4:3]

Reserved, res0.

RV, bit [2]
When FEAT_WFxT is implemented:

Register field Valid.

If TI[1] == 1, then this field indicates whether RN holds a valid register number for the register argument to the trapped `WFET` or `WFIT` instruction.

RV	Meaning
`0b0`	Register field invalid.
`0b1`	Register field valid.

If TI[1] == 0, then this field is res0.

This field is set to 1 on a trap on `WFET` or `WFIT`.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

TI, bits [1:0]

Trapped instruction. Possible values of this bit are:

TI	Meaning	Applies when
`0b00`	WFI trapped.	
`0b01`	WFE trapped.	
`0b10`	WFIT trapped.	When FEAT_WFxT is implemented
`0b11`	WFET trapped.	When FEAT_WFxT is implemented

When FEAT_WFxT is implemented, this is a two bit field as shown. Otherwise, bit[1] is res0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

The following fields describe configuration settings for generating this exception:

- SCTLR_EL1.{nTWE, nTWI}.
- HCR_EL2.{TWE, TWI}.
- SCR_EL3.{TWE, TWI}.

ISS encoding for an exception from an MCR or MRC access

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
CV	COND	Opc2	Opc1	CRn	Rt	CRm	Direction

CV, bit [24]

Condition code valid.

CV	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is implementation defined whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to `0b1110`.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to `0b1110`.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to `0b1110`, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is implementation defined whether:
  - CV is set to 0 and COND is set to an unknown value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.

    - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is implementation defined whether the COND field is set to `0b1110`, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Opc2, bits [19:17]

The Opc2 value from the issued instruction.

For a trapped VMRS access, holds the value `0b000`.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Opc1, bits [16:14]

The Opc1 value from the issued instruction.

For a trapped VMRS access, holds the value `0b111`.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

CRn, bits [13:10]

The CRn value from the issued instruction.

For a trapped VMRS access, holds the reg field from the VMRS instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

If the Rt value is not `0b1111`, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is `0b1111`:

- If the instruction that generated the exception is not unpredictable, then the register specifier takes the value `0b11111`.
- If the instruction that generated the exception is unpredictable, then the register specifier takes an unknown value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value `0b11111`.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

CRm, bits [4:1]

The CRm value from the issued instruction.

For a trapped VMRS access, holds the value `0b0000`.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
`0b0`	Write to System register space. MCR instruction.

0b1	Read from System register space. MRC or VMRS instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000011:

- If FEAT_TIDCP1 is implemented, SCTLR_EL1.TIDCP, for EL0 accesses to implementation defined functionality using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1.
- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- PMUSERENR_EL0.{ER, CR, SW, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- AMUSERENR_EL0.EN, for accesses to Activity Monitors registers from EL0 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TTLB, for execution of TLB maintenance instructions at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.{TSW, TPC, TPU} for execution of cache maintenance instructions at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TACR, for accesses to the Auxiliary Control Register at EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.TIDCP, for accesses to lockdown, DMA, and TCM operations at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- If FEAT_TIDCP1 is implemented, SCTLR_EL2.TIDCP, for EL0 accesses to implementation defined functionality using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.
- HCR_EL2.{TID1, TID2, TID3}, for accesses to ID registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1111) trapped to EL2.

- CPTR_EL2.TCPAC, for accesses to CPACR_EL1 or CPACR using AArch32 state, MCR or MRC access (coproc == `0b1111`) trapped to EL2.
- HSTR_EL2.T<n>, for accesses to System registers using AArch32 state, MCR or MRC access (coproc == `0b1111`) trapped to EL2.
- CNTHCTL_EL2.EL1PCEN, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == `0b1111`) trapped to EL2.
- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == `0b1111`) trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers from EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == `0b1111`) trapped to EL2.
- CPTR_EL3.TCPAC, for accesses to CPACR from EL1 and EL2, and accesses to HCPTR from EL2 using AArch32 state, MCR or MRC access (coproc == `0b1111`) trapped to EL3.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == `0b1111`) trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCR or MRC access (coproc == `0b1111`) trapped to EL3.
- If FEAT_FGT is implemented, MCR or MRC access to some registers at EL0, trapped to EL2.

The following fields describe configuration settings for generating exceptions that are reported using EC value `0b000101`:

- CPACR_EL1.TTA for accesses to trace registers, MCR or MRC access (coproc == `0b1110`) trapped to EL1 or EL2.
- MDSCR_EL1.TDCC, for accesses to the Debug Communications Channel (DCC) registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == `0b1110`) trapped to EL1 or EL2.
- If FEAT_FGT is implemented, MDCR_EL2.TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- HCR_EL2.TID0, for accesses to the JIDR register in the ID group 0 at EL0 and EL1 using AArch32, MRC access (coproc == `0b1110`) trapped to EL2.
- CPTR_EL2.TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == `0b1110`) trapped to EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers DBGDRAR and DBGDSAR using AArch32, MCR or MRC access (coproc == `0b1110`) trapped to EL2.

- MDCR_EL2.TDOSA, for accesses to powerdown debug registers, using AArch32 state, MCR or MRC access (coproc == `0b1110`) trapped to EL2.
- MDCR_EL2.TDA, for accesses to other debug registers, using AArch32 state, MCR or MRC access (coproc == `0b1110`) trapped to EL2.
- CPTR_EL3.TTA, for accesses to trace registers using AArch32, MCR or MRC access (coproc == `0b1110`) trapped to EL3.
- MDCR_EL3.TDOSA, for accesses to powerdown debug registers using AArch32, MCR or MRC access (coproc == `0b1110`) trapped to EL3.
- MDCR_EL3.TDA, for accesses to other debug registers, using AArch32, MCR or MRC access (coproc == `0b1110`) trapped to EL3.

The following fields describe configuration settings for generating exceptions that are reported using EC value `0b001000`:

- HCR_EL2.TID0, for accesses to the FPSID register in ID group 0 at EL1 using AArch32 state, VMRS access trapped to EL2.
- HCR_EL2.TID3, for accesses to registers in ID group 3 including MVFR0, MVFR1 and MVFR2, VMRS access trapped to EL2.

ISS encoding for an exception from an LD64B or ST64B* instruction

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
ISS

ISS, bits [24:0]

ISS	Meaning	Applies when
`0b0000000000000000000000000`	ST64BV instruction trapped.	When FEAT_LS64 is implemented
`0b0000000000000000000000001`	ST64BV0 instruction trapped.	When FEAT_LS64_ACC is implemented
`0b0000000000000000000000010`	LD64B or ST64B instruction trapped.	When FEAT_LS64 implemented

All other values are reserved.

ISS encoding for an exception from an MCRR or MRRC access

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
CV	COND	Opc1	RES0	Rt2	Rt	CRm	Direction

CV, bit [24]

Condition code valid.

CV	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is implementation defined whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to `0b1110`.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to `0b1110`.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to `0b1110`, the value for unconditional.
  - With the COND value held in the instruction.

- When a T32 instruction is trapped, it is implementation defined whether:
  - CV is set to 0 and COND is set to an unknown value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is implementation defined whether the COND field is set to `0b1110`, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Opc1, bits [19:16]

The Opc1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bit [15]

Reserved, res0.

Rt2, bits [14:10]

The Rt2 value from the issued instruction, the second general-purpose register used for the transfer.

If the Rt2 value is not `0b1111`, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt2 value is `0b1111`:

- If the instruction that generated the exception is not unpredictable, then the register specifier takes the value `0b11111`.

- If the instruction that generated the exception is unpredictable, then the register specifier takes an unknown value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value `0b11111`.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Rt, bits [9:5]

The Rt value from the issued instruction, the first general-purpose register used for the transfer.

If the Rt value is not `0b1111`, then the reported value gives the AArch64 view of the register. Otherwise, if the Rt value is `0b1111`:

- If the instruction that generated the exception is not unpredictable, then the register specifier takes the value `0b11111`.
- If the instruction that generated the exception is unpredictable, then the register specifier takes an unknown value, which is restricted to either:
  - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
  - The value `0b11111`.

See 'Mapping of the general-purpose registers between the Execution states'.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write to System register space. MCRR instruction.
0b1	Read from System register space. MRRC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000100:

- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN}, for accesses to the Generic Timer Registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- PMUSERENR_EL0.{CR, EN}, for accesses to Performance Monitor registers from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- AMUSERENR_EL0.{EN}, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers from EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- HSTR_EL2.T<n>, for accesses to System registers using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.
- CNTHCTL_EL2.{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == 0b1111) trapped to EL2.

- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == `0b1111`) trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers AMEVCNTR0<n> and AMEVCNTR1<n> from EL0 and EL1 using AArch32 state, MCRR or MRRC access (coproc == `0b1111`) trapped to EL2.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == `0b1111`) trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers from EL0, EL1 and EL2 using AArch32 state, MCRR or MRRC access (coproc == `0b1111`) trapped to EL3.
- If FEAT_FGT is implemented, HDFGRTR_EL2.PMCCNTR_EL0 for MRRC access and HDFGWTR_EL2.PMCCNTR_EL0 for MCRR access to PMCCNTR at EL0, trapped to EL2.

The following fields describe configuration settings for generating exceptions that are reported using EC value `0b001100`:

- MDSCR_EL1.TDCC, for accesses to the Debug ROM registers DBGDSAR and DBGDRAR at EL0 using AArch32 state, MCRR or MRRC access (coproc == `0b1110`) trapped to EL1 or EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers DBGDRAR and DBGDSAR using AArch32, MCRR or MRRC access (coproc == `0b1110`) trapped to EL2.
- MDCR_EL3.TDA, for accesses to debug registers, using AArch32, MCRR or MRRC access (coproc == `0b1110`) trapped to EL3.
- CPACR_EL1.TTA for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == `0b1110`) trapped to EL1 or EL2.
- CPTR_EL2.TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == `0b1110`) trapped to EL2.
- CPTR_EL3.TTA, for accesses to trace registers using AArch32, MCRR or MRRC access (coproc == `0b1110`) trapped to EL3.

Note

> If the Armv8-A architecture is implemented with an ETMv4 implementation, MCRR and MRRC accesses to trace registers are undefined and the resulting exception is higher priority than an exception due to these traps.

ISS encoding for an exception from an LDC or STC instruction

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
CV	COND	imm8	RES0	Rn	Offset	AM	Direction

CV, bit [24]

Condition code valid.

CV	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is implementation defined whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to `0b1110`.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to `0b1110`.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to `0b1110`, the value for unconditional.
  - With the COND value held in the instruction.

- When a T32 instruction is trapped, it is implementation defined whether:
    - CV is set to 0 and COND is set to an unknown value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
    - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is implementation defined whether the COND field is set to `0b1110`, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

imm8, bits [19:12]

The immediate value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bits [11:10]

Reserved, res0.

Rn, bits [9:5]

The Rn value from the issued instruction, the general-purpose register used for the transfer.

If the Rn value is not `0b1111`, then the reported value gives the AArch64 view of the register. Otherwise, if the Rn value is `0b1111`:

- If the instruction that generated the exception is not unpredictable, then the register specifier takes the value `0b11111`.

- If the instruction that generated the exception is unpredictable, then the register specifier takes an unknown value, which is restricted to either:
    - The AArch64 view of one of the registers that could have been used in AArch32 state at the Exception level that the instruction was executed at.
    - The value `0b11111`.

See 'Mapping of the general-purpose registers between the Execution states'.

This field is valid only when AM[2] is 0, indicating an immediate form of the LDC or STC instruction. When AM[2] is 1, indicating a literal form of the LDC or STC instruction, this field is unknown.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Offset, bit [4]

Indicates whether the offset is added or subtracted:

Offset	Meaning
`0b0`	Subtract offset.
`0b1`	Add offset.

This bit corresponds to the U bit in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

AM, bits [3:1]

Addressing mode. The permitted values of this field are:

AM	Meaning
`0b000`	Immediate unindexed.
`0b001`	Immediate post-indexed.
`0b010`	Immediate offset.
`0b011`	Immediate pre-indexed.

0b100	For a trapped STC instruction or a trapped T32 LDC instruction this encoding is reserved.
0b110	For a trapped STC instruction, this encoding is reserved.

The values 0b101 and 0b111 are reserved. The effect of programming this field to a reserved value is that behavior is constrained unpredictable, as described in 'Reserved values in System and memory-mapped registers and translation table entries'.

Bit [2] in this subfield indicates the instruction form, immediate or literal.

Bits [1:0] in this subfield correspond to the bits {P, W} in the instruction encoding.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
0b0	Write to memory. STC instruction.
0b1	Read from memory. LDC instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

The following fields describe the configuration settings for the traps that are reported using EC value 0b000110:

- MDSCR_EL1.TDCC, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint trapped to EL1 or EL2.
- MDCR_EL2.TDA, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint MCR or MRC access trapped to EL2.

- MDCR_EL3.TDA, for accesses using AArch32 state, LDC access to DBGDTRTXint or STC access to DBGDTRRXint MCR or MRC access trapped to EL3.
- If FEAT_FGT is implemented, MDCR_EL2.TDCC for LDC and STC accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.

ISS encoding for an exception from an access to SVE, Advanced SIMD or floating-point functionality, resulting from the FPEN and TFP traps

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
CV	COND	RES0

The accesses covered by this trap include:

- Execution of SVE or Advanced SIMD and floating-point instructions.
- Accesses to the Advanced SIMD and floating-point System registers.
- Execution of SME instructions.

For an implementation that does not include either SVE or support for Advanced SIMD and floating-point, the exception is reported using the EC value `0b000000`.

CV, bit [24]

Condition code valid.

CV	Meaning
`0b0`	The COND field is not valid.
`0b1`	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is implementation defined whether CV is set to 1 or set to 0. See the description of the COND field for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

COND, bits [23:20]

For exceptions taken from AArch64, this field is set to `0b1110`.

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
    - If the instruction is conditional, COND is set to the condition code field value from the instruction.
    - If the instruction is unconditional, COND is set to `0b1110`.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
    - With COND set to `0b1110`, the value for unconditional.
    - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is implementation defined whether:
    - CV is set to 0 and COND is set to an unknown value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
    - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is implementation defined whether the COND field is set to `0b1110`, or to the value of any condition that applied to the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bits [19:0]

Reserved, res0.

The following fields describe the configuration settings for the traps that are reported using EC value `0b000111`:

- CPACR_EL1.FPEN, for accesses to SIMD and floating-point registers trapped to EL1.
- CPTR_EL2.FPEN and CPTR_EL2.TFP, for accesses to SIMD and floating-point registers trapped to EL2.
- CPTR_EL3.TFP, for accesses to SIMD and floating-point registers trapped to EL3.

ISS encoding for an exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ

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
RES0																								

The accesses covered by this trap include:

- Execution of SVE instructions when the PE is not in Streaming SVE mode.
- Accesses to the SVE System registers, ZCR_ELx.

For an implementation that does not include SVE, the exception is reported using the EC value `0b000000`.

Bits [24:0]

Reserved, res0.

The following fields describe the configuration settings for the traps that are reported using EC value `0b011001`:

- CPACR_EL1.ZEN, for execution of SVE instructions and accesses to SVE registers at EL0 or EL1, trapped to EL1.
- CPTR_EL2.ZEN and CPTR_EL2.TZ, for execution of SVE instructions and accesses to SVE registers at EL0, EL1, or EL2, trapped to EL2.
- CPTR_EL3.EZ, for execution of SVE instructions and accesses to SVE registers from all Exception levels, trapped to EL3.

ISS encoding for a PMU exception

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
RES0																								SYNC

Bits [24:1]

Reserved, res0.

SYNC, bit [0]

Indicates whether the exception was taken synchronously or asynchronously.

SYNC	Meaning	Applies when
`0b0`	The exception was taken asynchronously because an overflow status flag was set.	
`0b1`	The exception was taken synchronously because PSTATE.PPEND was set.	When FEAT_SEBEP is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

ISS encoding for an exception from an Illegal Execution state, or a PC or SP alignment fault

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
RES0																								

Bits [24:0]

Reserved, res0.

There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about PC alignment fault exceptions, see 'PC alignment checking'.

'SP alignment checking' describes the configuration settings for generating SP alignment fault exceptions.

ISS encoding for an exception from the Memory Copy and Memory Set instructions

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
MemInst	isSETG	Options	FromEpilogue	WrongOption	OptionA	RES0	destreg	srcreg	sizereg

MemInst, bit [24]

Indicates the memory instruction class causing the exception.

MemInst	Meaning
0b0	CPYFE*, CPYFM*, CPYE*, and CPYM* instructions.
0b1	SETE*, SETM*, SETGE*, and SETGM* instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

isSETG, bit [23]

Indicates whether the instruction belongs to SETGM* or SETGE* class of instruction.

isSETG	Meaning
0b0	Not a SETGM* or SETGE* instruction.
0b1	SETGM* or SETGE* instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Options, bits [22:19]

Options : the Options field of the instruction.

For Memory Copy instructions, bits[22:19] forms the Options field, which holds the bits[15:12] of the instruction.

For Memory Set instructions:

- Bits[22:21] are res0.
- Bits[20:19] form the Options field, which holds the bits[13:12] of the instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

FromEpilogue, bit [18]

Indicates whether the instruction belongs to the epilogue class of Memory Copy or Memory Set instructions.

FromEpilogue	Meaning
0b0	Not an epilogue instruction.
0b1	CPYE*, CPYFE*, SETE*, or SETGE* instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

WrongOption, bit [17]

Algorithm option.

WrongOption	Meaning
0b0	WrongOption is false.
0b1	WrongOption is true.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

OptionA, bit [16]

Algorithm type indicated by the PSTATE.C bit.

OptionA	Meaning
0b0	OptionB indicated by PSTATE.C is 0.
0b1	OptionA indicated by PSTATE.C is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bit [15]

Reserved, res0.

destreg, bits [14:10]

The destination register value from the issued instruction, containing the destination address.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

srcreg, bits [9:5]

The source register value from the issued instruction, containing either the source address or the source data.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

sizereg, bits [4:0]

The size register value from the issued instruction, containing the number of bytes to be transfered or set.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

ISS encoding for an exception from HVC or SVC instruction execution

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
RES0									imm16															

Bits [24:16]

Reserved, res0.

imm16, bits [15:0]

The value of the immediate field from the HVC or SVC instruction.

For an HVC instruction, and for an A64 SVC instruction, this is the value of the imm16 field of the issued instruction.

For an A32 or T32 SVC instruction:

- If the instruction is unconditional, then:
  - For the T32 instruction, this field is zero-extended from the imm8 field of the instruction.
  - For the A32 instruction, this field is the bottom 16 bits of the imm24 field of the instruction.
- If the instruction is conditional, this field is unknown.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

In AArch32 state, the HVC instruction is unconditional, and a conditional SVC instruction generates an exception only if it passes its condition code check. Therefore, the syndrome information for these exceptions does not require conditionality information.

For T32 and A32 instructions, see 'SVC' and 'HVC'.

For A64 instructions, see 'SVC' and 'HVC'.

If FEAT_FGT is implemented, HFGITR_EL2.{SVC_EL1, SVC_EL0} control fine-grained traps on SVC execution.

ISS encoding for an exception from MSR, MRS, or System instruction execution in AArch64 state

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
RES0	Op0	Op2	Op1	CRn	Rt	CRm	Direction

Bits [24:22]

Reserved, res0.

Op0, bits [21:20]

The Op0 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Op2, bits [19:17]

The Op2 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Op1, bits [16:14]

The Op1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

CRn, bits [13:10]

The CRn value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Rt, bits [9:5]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
`0b0`	Write access, including MSR instructions.

0b1	Read access, including MRS instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

For exceptions caused by System instructions, see 'System instructions' subsection of 'Branches, exception generating and System instructions' for the encoding values returned by an instruction.

The following fields describe configuration settings for generating the exception that is reported using EC value `0b011000`:

- If FEAT_TIDCP1 is implemented, SCTLR_EL1.TIDCP, for EL0 accesses to implementation defined functionality using AArch64 state, MSR or MRS access trapped to EL1.
- SCTLR_EL1.UCI, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- SCTLR_EL1.UCT, for accesses to CTR_EL0 using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- SCTLR_EL1.DZE, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- SCTLR_EL1.UMA, for accesses to the PSTATE interrupt masks using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- CPACR_EL1.TTA, for accesses to the trace registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- MDSCR_EL1.TDCC, for accesses to the Debug Communications Channel (DCC) registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- If FEAT_FGT is implemented, MDCR_EL2.TDCC for accesses to the DCC registers at EL0 and EL1 trapped to EL2, and MDCR_EL3.TDCC for accesses to the DCC registers at EL0, EL1, and EL2 trapped to EL3.
- CNTKCTL_EL1.{EL0PTEN, EL0VTEN, EL0PCTEN, EL0VCTEN} accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- PMUSERENR_EL0.{ER, CR, SW, EN}, for accesses to the Performance Monitor registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- AMUSERENR_EL0.EN, for accesses to Activity Monitors registers using AArch64 state, MSR or MRS access trapped to EL1 or EL2.
- HCR_EL2.{TRVM, TVM}, for accesses to virtual memory control registers using AArch64 state, MSR or MRS access trapped to EL2.

- HCR_EL2.TDZ, for execution of DC ZVA instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TTLB, for execution of TLB maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{TSW, TPC, TPU}, for execution of cache maintenance instructions using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TACR, for accesses to the Auxiliary Control Register, ACTLR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.TIDCP, for accesses to lockdown, DMA, and TCM operations using AArch64 state, MSR or MRS access trapped to EL2.
- If FEAT_TIDCP1 is implemented, SCTLR_EL2.TIDCP, for EL0 accesses to implementation defined functionality using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{TID1, TID2, TID3}, for accesses to ID group 1, ID group 2 or ID group 3 registers, using AArch64 state, MSR or MRS access trapped to EL2.
- CPTR_EL2.TCPAC, for accesses to CPACR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.
- CPTR_EL2.TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TTRF, for accesses to the trace filter control register, TRFCR_EL1, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDRA, for accesses to Debug ROM registers, using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDOSA, for accesses to powerdown debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- CNTHCTL_EL2.{EL1PCEN, EL1PCTEN}, for accesses to the Generic Timer registers using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.TDA, for accesses to debug registers using AArch64 state, MSR or MRS access trapped to EL2.
- MDCR_EL2.{TPM, TPMCR}, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL2.
- CPTR_EL2.TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.APK, for accesses to Pointer authentication key registers. using AArch64 state, MSR or MRS access trapped to EL2.
- HCR_EL2.{NV, NV1}, for Nested virtualization register access, using AArch64 state, MSR or MRS access, trapped to EL2.
- HCR_EL2.AT, for execution of AT S1E* instructions, using AArch64 state, MSR or MRS access, trapped to EL2.
- HCR_EL2.{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access, trapped to EL2.

- SCR_EL3.APK, for accesses to Pointer authentication key registers, using AArch64 state, MSR or MRS access trapped to EL3.
- SCR_EL3.ST, for accesses to the Counter-timer Physical Secure timer registers, using AArch64 state, MSR or MRS access trapped to EL3.
- SCR_EL3.{TERR, FIEN}, for accesses to RAS registers, using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TCPAC, for accesses to CPTR_EL2 and CPACR_EL1 using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TTA, for accesses to the trace registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TTRF, for accesses to the trace filter control registers, TRFCR_EL1 and TRFCR_EL2, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TDA, for accesses to debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TDOSA, for accesses to powerdown debug registers, using AArch64 state, MSR or MRS access trapped to EL3.
- MDCR_EL3.TPM, for accesses to Performance Monitor registers, using AArch64 state, MSR or MRS access trapped to EL3.
- CPTR_EL3.TAM, for accesses to Activity Monitors registers, using AArch64 state, MSR or MRS access, trapped to EL3.
- If FEAT_EVT is implemented, the following registers control traps for EL1 and EL0 Cache controls that use this EC value:
    - HCR_EL2.{TTLBOS, TTLBIS, TICAB, TOCU, TID4}.
    - HCR2.{TTLBIS, TICAB, TOCU, TID4}.
- If FEAT_FGT is implemented:
    - SCR_EL3.FGTEn, for accesses to the fine-grained trap registers, MSR or MRS access at EL2 trapped to EL3.
    - HFGRTR_EL2 for reads and HFGWTR_EL2 for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 trapped to EL2.
    - HFGITR_EL2 for execution of system instructions, MSR or MRS access trapped to EL2
    - HDFGRTR_EL2 for reads and HDFGWTR_EL2 for writes of registers, using AArch64 state, MSR or MRS access at EL0 and EL1 state trapped to EL2.
    - HAFGRTR_EL2 for reads of Activity Monitor counters, using AArch64 state, MRS access at EL0 and EL1 trapped to EL2.
- If FEAT_RNG_TRAP is implemented:
    - SCR_EL3.TRNDR for reads of RNDR and RNDRRS using AArch64 state, MRS access trapped to EL3.
- If FEAT_SME is implemented:
    - CPTR_EL3.ESM, for MSR or MRS accesses to SMPRI_EL1 at EL1, EL2, and EL3, trapped to EL3.
    - CPTR_EL3.ESM, for MSR or MRS accesses to SMPRIMAP_EL2 at EL2 and EL3, trapped to EL3.

    - SCTLR_EL1.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, trapped to EL1 or EL2.
    - SCTLR_EL2.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, trapped to EL2.
    - SCR_EL3.EnTP2, for MSR or MRS accesses to TPIDR2_EL0 at EL0, EL1, and EL2, trapped to EL3.
- If FEAT_NMI is implemented, HCRX_EL2.TALLINT, for MSR writes of ALLINT at EL1, trapped to EL2.

ISS encoding for an exception from MSRR, MRRS, or 128-bit System instruction execution in AArch64 state

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
RES0	Op0	Op2	Op1	CRn	Rt	RES0	CRm	Direction

Bits [24:22]

Reserved, res0.

Op0, bits [21:20]

The Op0 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Op2, bits [19:17]

The Op2 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Op1, bits [16:14]

The Op1 value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

CRn, bits [13:10]

The CRn value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Rt, bits [9:6]

The Rt value from the issued instruction, the general-purpose register used for the transfer.

Note

> This value represents register pair of X[Rt:0], X[Rt:1].

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bit [5]

Reserved, res0.

CRm, bits [4:1]

The CRm value from the issued instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Direction, bit [0]

Indicates the direction of the trapped instruction.

Direction	Meaning
`0b0`	Write access, MSRR instructions.
`0b1`	Read access, MRRS instructions.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

ISS encoding for an exception from an Instruction Abort

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
RES0	PFV	RES0	SET	FnV	EA	RES0	S1PTW	RES0	IFSC

When FEAT_S1POE or FEAT_S2POE is implemented, if a memory access generates a Instruction Abort due to a Permission fault, the ISS2 encoding for an exception from an Instruction Abort includes further information about the exception.

Bits [24:15]

Reserved, res0.

PFV, bit [14]

When FEAT_PFAR is implemented and (IFSC == 0b010000, or IFSC == 0b01001x or IFSC == 0b0101xx):

FAR Valid. Describes whether the PFAR_EL1 is valid.

PFV	Meaning
`0b0`	PFAR_EL1 is unknown.
`0b1`	PFAR_EL1 is valid.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Bit [13]

Reserved, res0.

SET, bits [12:11]

When FEAT_RAS is implemented and (IFSC == 0b010000, or IFSC == 0b01001x or IFSC == 0b0101xx):

Synchronous Error Type. Describes the PE error state after taking the Instruction Abort exception.

SET	Meaning	Applies when
`0b00`	Recoverable state (UER).	

`0b10`	Uncontainable (UC).	When FEAT_RASv2 is not implemented
`0b11`	Restartable state (UEO).	

All other values are reserved.

Note

> Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
`0b0`	FAR is valid.
`0b1`	FAR is not valid, and holds an unknown value.

This field is valid only if the IFSC code is `0b010000`. It is res0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

EA, bit [9]

External abort type. This bit can provide an implementation defined classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bit [8]

Reserved, res0.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
`0b0`	Fault not on a stage 2 translation for a stage 1 translation table walk.
`0b1`	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is res0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bit [6]

Reserved, res0.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning	Applies when

`0b000000`	Address size fault, level 0 of translation or translation table base register.	
`0b000001`	Address size fault, level 1.	
`0b000010`	Address size fault, level 2.	
`0b000011`	Address size fault, level 3.	
`0b000100`	Translation fault, level 0.	
`0b000101`	Translation fault, level 1.	
`0b000110`	Translation fault, level 2.	
`0b000111`	Translation fault, level 3.	
`0b001001`	Access flag fault, level 1.	
`0b001010`	Access flag fault, level 2.	
`0b001011`	Access flag fault, level 3.	
`0b001000`	Access flag fault, level 0.	When FEAT_LPA2 is implemented
`0b001100`	Permission fault, level 0.	When FEAT_LPA2 is implemented
`0b001101`	Permission fault, level 1.	
`0b001110`	Permission fault, level 2.	
`0b001111`	Permission fault, level 3.	
`0b010000`	Synchronous External abort, not on translation table walk or hardware update of translation table.	

0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	

0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented

`0b011101`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
`0b011110`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented
`0b011111`	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
`0b100010`	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented

0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented
0b100110	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
0b100111	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented

0b101000	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
0b101001	Address size fault, level -1.	When FEAT_LPA2 is implemented
0b101010	Translation fault, level -2.	When FEAT_D128 is implemented
0b101011	Translation fault, level -1.	When FEAT_LPA2 is implemented
0b101100	Address Size fault, level -2.	When FEAT_D128 is implemented
0b110000	TLB conflict abort.	
0b110001	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The lookup level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

ISS encoding for an exception due to SME functionality

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
RES0	SMTC

The accesses covered by this trap include:

- Execution of SME instructions.
- Execution of SVE and Advanced SIMD instructions, when the PE is in Streaming SVE mode.
- Direct accesses of SVCR, SMCR_EL1, SMCR_EL2, SMCR_EL3.

Bits [24:3]

Reserved, res0.

SMTC, bits [2:0]

SME Trap Code. Identifies the reason for instruction trapping.

SMTC	Meaning
`0b000`	Access to SME functionality trapped as a result of CPACR_EL1.SMEN, CPTR_EL2.SMEN, CPTR_EL2.TSM, or CPTR_EL3.ESM, that is not reported using EC `0b000000`.
`0b001`	Advanced SIMD, SVE, or SVE2 instruction trapped because PSTATE.SM is 1.
`0b010`	SME instruction trapped because PSTATE.SM is 0.
`0b011`	SME instruction trapped because PSTATE.ZA is 0.
`0b100`	Access to the SME2 ZT0 register trapped as a result of SMCR_EL1.EZT0, SMCR_EL2.EZT0, or SMCR_EL3.EZT0.

All other values are reserved.

The following fields describe the configuration settings for the traps that are reported using the EC value `0b011101`:

- CPACR_EL1.SMEN, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR and SMCR_EL1 System registers at EL1 and EL0, trapped to EL1 or EL2.
- CPTR_EL2.SMEN and CPTR_EL2.TSM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR, SMCR_EL1, SMCR_EL2 at EL2, EL1, or EL0, trapped to EL2.
- CPTR_EL3.ESM, for execution of SME instructions, SVE instructions when the PE is in Streaming SVE mode, and instructions that directly access SVCR, SMCR_EL1, SMCR_EL2, SMCR_EL3 from all Exception levels and any Security state, trapped to EL3.

- If FEAT_SME2 is implemented:
    - SMCR_EL1.EZT0, for accesses to ZT0 at EL1 and EL0, trapped to EL1 or EL2.
    - SMCR_EL2.EZT0, for accesses to ZT0 at EL2, EL1, and EL0, trapped to EL2.
    - SMCR_EL3.EZT0, for accesses to ZT0 at any Exception level, trapped to EL3.

ISS encoding for an exception from a Data Abort

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
ISV	SAS	SSE	Bits[20:16]	Bit[15]	Bit[14]	VNCR	Bits[12:11]	FnV	EA	CM	S1PTW	WnR	DFSC

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, this ISS encoding includes ISS2, bits[36:32].

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, this ISS encoding includes ISS2, bits[36:32].

When FEAT_S1POE is implemented, if a memory access generates a Data Abort due to a Permission fault, the ISS2 encoding for an exception from a Data Abort includes further information about the exception.

When FEAT_S1PIE is implemented, if a memory write access generates a Data Abort due to a Permission fault, the ISS2 encoding for an exception from a Data Abort includes further information about the exception.

ISV, bit [24]

Instruction Syndrome Valid. Indicates whether the syndrome information in ISS[23:14] is valid.

ISV	Meaning
`0b0`	No valid instruction syndrome. ISS[23:14] are res0.
`0b1`	ISS[23:14] hold a valid instruction syndrome.

In ESR_EL2, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

In ESR_EL2, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For other faults reported in ESR_EL2, ISV is 0 except for the following stage 2 aborts:

- AArch64 loads and stores of a single general-purpose register (including the register specified with `0b11111`, including those with Acquire/Release semantics, but excluding Load Exclusive or Store Exclusive and excluding those with writeback).
- AArch32 instructions where the instruction:
    - Is an LDR, LDA, LDRT, LDRSH, LDRSHT, LDRH, LDAH, LDRHT, LDRSB, LDRSBT, LDRB, LDAB, LDRBT, STR, STL, STRT, STRH, STLH, STRHT, STRB, STLB, or STRBT instruction.
    - Is not performing register writeback.
    - Is not using R15 as a source or destination register.

For these stage 2 aborts, ISV is unknown if the exception was generated in Debug state in memory access mode, and otherwise indicates whether ISS[23:14] hold a valid syndrome.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64 is implemented and a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_V is implemented and a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

For faults reported in ESR_EL1 or ESR_EL3, ISV is 1 when FEAT_LS64_ACCDATA is implemented and a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

When FEAT_RAS is not implemented, it is implementation defined whether ISV is set to 1 or 0 on a synchronous External abort on a stage 2 translation table walk.

For ISS reporting, a stage 2 abort on a stage 1 translation table walk does not return a valid instruction syndrome, and therefore ISV is 0 for these aborts.

When FEAT_MTE2 is implemented, for a synchronous Tag Check Fault abort taken to ELx, ESR_ELx.FnV is 0 and FAR_ELx is valid.

When FEAT_MOPS is implemented, for a synchronous Data Abort on a Memory Copy and Memory Set instruction, ISV is 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

SAS, bits [23:22]
When ISV == 1:

Syndrome Access Size. Indicates the size of the access attempted by the faulting operation.

SAS	Meaning
`0b00`	Byte
`0b01`	Halfword
`0b10`	Word
`0b11`	Doubleword

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is `0b11`.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is `0b11`.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is `0b11`.

This field is unknown when the value of ISV is unknown.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

SSE, bit [21]
When ISV == 1:

Syndrome Sign Extend. For a byte, halfword, or word load operation, indicates whether the data item must be sign extended.

SSE	Meaning
0b0	Sign-extension not required.
0b1	Data item must be sign-extended.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

For all other operations, this field is 0.

This field is unknown when the value of ISV is unknown.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Bits[20:16]
When ISV == 1:

SRT, bits [4:0] of bits [20:16]

Syndrome Register Transfer. The register number of the Wt/Xt/Rt operand of the faulting instruction.

If the exception was taken from an Exception level that is using AArch32, then this is the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states'.

This field is unknown when the value of ISV is unknown.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

When ISV == 0, FEAT_RASv2 is implemented and (DFSC == 0b010000, or DFSC == 0b01001x or DFSC == 0b0101xx):

Bits [4:2] of bits [20:16]

Reserved, res0.

WU, bits [1:0] of bits [20:16]

Write Update. Describes whether a store instruction that generated an External abort updated the location.

WU	Meaning
`0b00`	Not a store instruction or translation table update, or the location might have been updated.
`0b10`	Store instruction or translation table update that did not update the location.
`0b11`	Store instruction or translation table update that updated the location.

In the description of this field, a store instruction is any memory-writing instruction that explicitly performs a store. This includes instructions that both read and write memory.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Bit[15]
When ISV == 1:

SF, bit [15]

Sixty Four bit general-purpose register transfer. Width of the register accessed by the instruction is 64-bit.

SF	Meaning
`0b0`	Instruction loads/stores a 32-bit general-purpose register.
`0b1`	Instruction loads/stores a 64-bit general-purpose register.

Note

This field specifies the register width identified by the instruction, not the Execution state.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 1.

This field is unknown when the value of ISV is unknown.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

When ISV == 0:

FnP, bit [15]

FAR not Precise.

FnP	Meaning	Applies when
`0b0`	The FAR holds the faulting virtual address that generated the Data Abort.	
`0b1`	The FAR holds any virtual address within the naturally-aligned granule that contains the faulting virtual address that generated a Data Abort due to an SVE contiguous vector load/store instruction, or an SME load/store instruction. For more information about the naturally-aligned fault granule, see FAR_ELx (for example, FAR_EL1).	When FEAT_SME is implemented or FEAT_SVE is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Bit[14]
When ISV == 1:

AR, bit [14]

Acquire/Release.

AR	Meaning
0b0	Instruction did not have acquire/release semantics.
0b1	Instruction did have acquire/release semantics.

When FEAT_LS64 is implemented, if a memory access generated by an LD64B or ST64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_V is implemented, if a memory access generated by an ST64BV instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

When FEAT_LS64_ACCDATA is implemented, if a memory access generated by an ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field is 0.

This field is unknown when the value of ISV is unknown.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

When FEAT_PFAR is implemented and (DFSC == 0b010000, or DFSC == 0b01001x or DFSC == 0b0101xx):

PFV, bit [14]

FAR Valid. Describes whether the PFAR_EL1 is valid.

PFV	Meaning
0b0	PFAR_EL1 is unknown.
0b1	PFAR_EL1 is valid.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

VNCR, bit [13]

Indicates that the fault came from use of VNCR_EL2 register by EL1 code.

VNCR	Meaning	Applies when
0b0	The watchpoint was not generated by the use of VNCR_EL2 by EL1 code.	
0b1	The watchpoint was generated by the use of VNCR_EL2 by EL1 code.	When FEAT_NV2 is implemented

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bits[12:11]
When (DFSC == 0b00xxxx || DFSC == 0b101011) && DFSC != 0b0000xx:

LST, bits [1:0] of bits [12:11]

Load/Store Type. Used when a Translation fault, Access flag fault, or Permission fault generates a Data Abort.

LST	Meaning	Applies when
0b00	The instruction that generated the Data Abort is not specified.	

0b01	An ST64BV instruction generated the Data Abort.	When FEAT_LS64_V is implemented
0b10	An LD64B or ST64B instruction generated the Data Abort.	When FEAT_LS64 is implemented
0b11	An ST64BV0 instruction generated the Data Abort.	When FEAT_LS64_ACCDATA is implemented

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

When FEAT_RAS is implemented and (DFSC == 0b010000, or DFSC == 0b01001x or DFSC == 0b0101xx):

SET, bits [1:0] of bits [12:11]

Synchronous Error Type. Used when a Syncronous External abort, not on a Translation table walk or hardware update of the Translation table, generated the Data Abort. Describes the PE error state after taking the Data Abort exception.

SET	Meaning	Applies when
0b00	Recoverable state (UER).	
0b10	Uncontainable (UC).	When FEAT_RASv2 is not implemented
0b11	Restartable state (UEO).	

Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in a PE state that is not recoverable.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

FnV, bit [10]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
`0b0`	FAR is valid.
`0b1`	FAR is not valid, and holds an unknown value.

This field is valid only if the DFSC code is `0b010000`. It is res0 for all other aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

EA, bit [9]

External abort type. This bit can provide an implementation defined classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

CM, bit [8]

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

CM	Meaning
`0b0`	The Data Abort was not generated by the execution of one of the System instructions identified in the description of value 1.
`0b1`	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as cache maintenance instructions, and therefore their execution cannot cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
`0b0`	Fault not on a stage 2 translation for a stage 1 translation table walk.
`0b1`	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is res0.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

WnR, bit [6]

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
`0b0`	Abort caused by an instruction reading from a memory location.
`0b1`	Abort caused by an instruction writing to a memory location.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault which is being reported, otherwise it is set to 1. The architecture permits, but does not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is unknown for:

- If FEAT_RASv2 is implemented, an External abort on an Atomic access, reported with ESR_ELx.WU set to `0b00`.
- A fault reported using a DFSC value of `0b110101` or `0b110001`, indicating an unsupported Exclusive or atomic access.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

DFSC, bits [5:0]

Data Fault Status Code.

DFSC	Meaning	Applies when

0b000000	Address size fault, level 0 of translation or translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
0b000100	Translation fault, level 0.	
0b000101	Translation fault, level 1.	
0b000110	Translation fault, level 2.	
0b000111	Translation fault, level 3.	
0b001001	Access flag fault, level 1.	
0b001010	Access flag fault, level 2.	
0b001011	Access flag fault, level 3.	
0b001000	Access flag fault, level 0.	When FEAT_LPA2 is implemented
0b001100	Permission fault, level 0.	When FEAT_LPA2 is implemented
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk or hardware update of translation table.	
0b010001	Synchronous Tag Check Fault.	When FEAT_MTE2 is implemented

0b010010	Synchronous External abort on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented
0b010011	Synchronous External abort on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented
0b010100	Synchronous External abort on translation table walk or hardware update of translation table, level 0.	
0b010101	Synchronous External abort on translation table walk or hardware update of translation table, level 1.	
0b010110	Synchronous External abort on translation table walk or hardware update of translation table, level 2.	
0b010111	Synchronous External abort on translation table walk or hardware update of translation table, level 3.	

0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011011	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level -1.	When FEAT_LPA2 is implemented and FEAT_RAS is not implemented
0b011100	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 0.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 2.	When FEAT_RAS is not implemented

0b011111	Synchronous parity or ECC error on memory access on translation table walk or hardware update of translation table, level 3.	When FEAT_RAS is not implemented
0b100001	Alignment fault.	
0b100010	Granule Protection Fault on translation table walk or hardware update of translation table, level -2.	When FEAT_D128 is implemented and FEAT_RME is implemented
0b100011	Granule Protection Fault on translation table walk or hardware update of translation table, level -1.	When FEAT_RME is implemented and FEAT_LPA2 is implemented
0b100100	Granule Protection Fault on translation table walk or hardware update of translation table, level 0.	When FEAT_RME is implemented
0b100101	Granule Protection Fault on translation table walk or hardware update of translation table, level 1.	When FEAT_RME is implemented

`0b100110`	Granule Protection Fault on translation table walk or hardware update of translation table, level 2.	When FEAT_RME is implemented
`0b100111`	Granule Protection Fault on translation table walk or hardware update of translation table, level 3.	When FEAT_RME is implemented
`0b101000`	Granule Protection Fault, not on translation table walk or hardware update of translation table.	When FEAT_RME is implemented
`0b101001`	Address size fault, level -1.	When FEAT_LPA2 is implemented
`0b101010`	Translation fault, level -2.	When FEAT_D128 is implemented
`0b101011`	Translation fault, level -1.	When FEAT_LPA2 is implemented
`0b101100`	Address Size fault, level -2.	When FEAT_D128 is implemented
`0b110000`	TLB conflict abort.	
`0b110001`	Unsupported atomic hardware update fault.	When FEAT_HAFDBS is implemented
`0b110100`	implementation defined fault (Lockdown).	

0b110101	implementation defined fault (Unsupported Exclusive or Atomic access).

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The lookup level associated with MMU faults'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

ISS encoding for an exception from a trapped floating-point exception

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
RES0	TFV	RES0	VECITR	IDF	RES0	IXF	UFF	OFF	DZF	IOF

Bit [24]

Reserved, res0.

TFV, bit [23]

Trapped Fault Valid bit. Indicates whether the IDF, IXF, UFF, OFF, DZF, and IOF bits hold valid information about trapped floating-point exceptions.

TFV	Meaning
0b0	The IDF, IXF, UFF, OFF, DZF, and IOF bits do not hold valid information about trapped floating-point exceptions and are unknown.

0b1	One or more floating-point exceptions occurred during an operation performed while executing the reported instruction. The IDF, IXF, UFF, OFF, DZF, and IOF bits indicate trapped floating-point exceptions that occurred. For more information, see 'Floating-point exceptions and exception traps'.

It is implementation defined whether this field is set to 0 on an exception generated by a trapped floating-point exception from an instruction that is performing floating-point operations on more than one lane of a vector.

Note

This is not a requirement. Implementations can set this field to 1 on a trapped floating-point exception from an instruction and return valid information in the {IDF, IXF, UFF, OFF, DZF, IOF} fields.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bits [22:11]

Reserved, res0.

VECITR, bits [10:8]

For a trapped floating-point exception from an instruction executed in AArch32 state this field is res1.

For a trapped floating-point exception from an instruction executed in AArch64 state this field is unknown.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

IDF, bit [7]

Input Denormal floating-point exception trapped bit. If the TFV field is 0, this bit is unknown. Otherwise, the possible values of this bit are:

IDF	Meaning
`0b0`	Input denormal floating-point exception has not occurred.
`0b1`	Input denormal floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bits [6:5]

Reserved, res0.

IXF, bit [4]

Inexact floating-point exception trapped bit. If the TFV field is 0, this bit is unknown. Otherwise, the possible values of this bit are:

IXF	Meaning
`0b0`	Inexact floating-point exception has not occurred.
`0b1`	Inexact floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

UFF, bit [3]

Underflow floating-point exception trapped bit. If the TFV field is 0, this bit is unknown. Otherwise, the possible values of this bit are:

UFF	Meaning

0b0	Underflow floating-point exception has not occurred.
0b1	Underflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

OFF, bit [2]

Overflow floating-point exception trapped bit. If the TFV field is 0, this bit is unknown. Otherwise, the possible values of this bit are:

OFF	Meaning
0b0	Overflow floating-point exception has not occurred.
0b1	Overflow floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

DZF, bit [1]

Divide by Zero floating-point exception trapped bit. If the TFV field is 0, this bit is unknown. Otherwise, the possible values of this bit are:

DZF	Meaning
0b0	Divide by Zero floating-point exception has not occurred.
0b1	Divide by Zero floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

IOF, bit [0]

Invalid Operation floating-point exception trapped bit. If the TFV field is 0, this bit is unknown. Otherwise, the possible values of this bit are:

IOF	Meaning
`0b0`	Invalid Operation floating-point exception has not occurred.
`0b1`	Invalid Operation floating-point exception occurred during execution of the reported instruction.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

In an implementation that supports the trapping of floating-point exceptions:

- From an Exception level using AArch64, the FPCR.{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.
- From an Exception level using AArch32, the FPSCR.{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.

ISS encoding for a GCS exception

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
RES0	ExType	RES0	Raddr	Bits[9:5]	IT

Bit [24]

Reserved, res0.

ExType, bits [23:20]

The first level classification of GCS exceptions.

ExType	Meaning

0b0000	The exception reported is a Guarded control stack Data Check Exception.
0b0001	The exception reported is an EXLOCK Exception.
0b0010	The exception reported is a trap exception on GCSSTR or GCSSTTR instruction execution.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bits [19:15]

Reserved, res0.

Raddr, bits [14:10]
When ExType == 0b0010 :

Indicates the data address register number supplied in the instruction that has been trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Bits[9:5]
When ExType == 0b0000 :

Rn, bits [4:0] of bits [9:5]

Indicates the register number supplied in the instruction that caused the Guarded control stack Data Check Exception.

This field is unknown if ESR_ELx.ISS.IT is reported as `0b00101` or `0b01000`

This field is `0b11111` if ESR_ELx.ISS.IT is reported as `0b01001`

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

When ExType == 0b0010 :

Rvalue, bits [4:0] of bits [9:5]

Indicates the data value register number supplied in the instruction that has been trapped.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

IT, bits [4:0]
When ExType == 0b0000 :

Type of the instruction that caused the Guarded control stack Data Check Exception.

IT	Meaning
`0b00000`	Guarded control stack Data Check Exception is from a procedure return instruction without Pointer authentication.
`0b00001`	Guarded control stack Data Check Exception is from a GCSPOPM instruction.
`0b00010`	Guarded control stack Data Check Exception is from a procedure return instruction with Pointer authentication that uses key A.
`0b00011`	Guarded control stack Data Check Exception is from a procedure return instruction with Pointer authentication that uses key B.

0b00100	Guarded control stack Data Check Exception is from a GCSSS1 instruction.
0b00101	Guarded control stack Data Check Exception is from a GCSSS2 instruction.
0b01000	Guarded control stack Data Check Exception is from a GCSPOPCX instruction.
0b01001	Guarded control stack Data Check Exception is from a GCSPOPX instruction.

All other values are reserved

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

ISS encoding for an SError exception

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
IDS	RES0	ELS	WU	VFV	PFV	IESB	AET	EA	RES0	WnRV	WnR	DFSC

IDS, bit [24]

implementation defined syndrome.

IDS	Meaning
0b0	Bits [23:0] of the ISS field holds the fields described in this encoding. **Note** If FEAT_RAS is not implemented, bits [23:0] of the ISS field are res0.

0b1	Bits [23:0] of the ISS field holds implementation defined syndrome information that can be used to provide additional information about the SError exception.

Note

This field was previously called ISV.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bits [23:19]

Reserved, res0.

ELS, bit [18]
When FEAT_RASv2 is implemented and DFSC == 0b010001:

Meaning of ELR_ELx.

ELS	Meaning
0b0	Asynchronous. Does not indicate the trigger for the exception.
0b1	Synchronous. The exception was triggered by the instruction at ELR_ELx.

SError exceptions that report this field is 1 are not required to be precise.

The ESR_EL1.AET field describes whether the exception is precise or imprecise.

Corrected, Recoverable or Restartable exceptions are precise. Unrecoverable or Uncontainable exceptions are imprecise.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

WU, bits [17:16]
When FEAT_RASv2 is implemented and DFSC == 0b010001:

Write Update. Describes whether a store instruction that generated an External abort updated the location.

WU	Meaning
`0b00`	Not a store instruction or translation table update, or the location might have been updated.
`0b10`	Store instruction or translation table update that did not update the location.
`0b11`	Store instruction or translation table update that updated the location.

In the description of this field, a store instruction is any memory-writing instruction that explicitly performs a store. This includes instructions that both read and write memory.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

VFV, bit [15]
When FEAT_RASv2 is implemented and DFSC == 0b010001:

FAR Valid. Indicates the FAR_ELx register contains a valid virtual address.

VFV	Meaning
`0b0`	FAR_ELx is not valid, and holds an unknown value.
`0b1`	FAR_ELx contains a valid virtual address associated with the error.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

PFV, bit [14]
When FEAT_PFAR is implemented and DFSC == 0b010001:

FAR Valid. Describes whether the PFAR_EL1 is valid.

PFV	Meaning
`0b0`	PFAR_EL1 is unknown.
`0b1`	PFAR_EL1 is valid.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

IESB, bit [13]
When FEAT_IESB is implemented and DFSC == 0b010001:

Implicit error synchronization event.

IESB	Meaning
`0b0`	The SError exception was either not synchronized by the implicit error synchronization event or not taken immediately.
`0b1`	The SError exception was synchronized by the implicit error synchronization event and taken immediately.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

AET, bits [12:10]
When FEAT_RAS is implemented and DFSC == 0b010001:

Asynchronous Error Type.

Describes the PE error state after taking the SError interrupt exception.

AET	Meaning
`0b000`	Uncontainable (UC).
`0b001`	Unrecoverable state (UEU).
`0b010`	Restartable state (UEO).
`0b011`	Recoverable state (UER).
`0b110`	Corrected (CE).

All other values are reserved.

If multiple errors are taken as a single SError interrupt exception, the overall PE error state is reported.

Note

> Software can use this information to determine what recovery might be possible. The recovery software must also examine any implemented fault records to determine the location and extent of the error.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

EA, bit [9]
When FEAT_RAS is implemented and DFSC == 0b010001:

External abort type. Provides an implementation defined classification of External aborts.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Bit [8]

Reserved, res0.

WnRV, bit [7]
When FEAT_RASv2 is implemented and DFSC == 0b010001:

ESR_ELx.WnR valid.

WnRV	Meaning
`0b0`	ESR_ELx.WnR is not valid and has been set to `0b0`.
`0b1`	ESR_ELx.WnR is valid.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

WnR, bit [6]
When FEAT_RASv2 is implemented and DFSC == 0b010001:

Write-not-Read. When the WnRV field is `0b1`, indicates whether an exception was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning

`0b0`	Exception was caused by an instruction reading from a memory location.
`0b1`	Exception was caused by an instruction writing to a memory location.

Accessing this bit has the following behavior:

- This bit is res0 if ESR_ELx.WnRV==`0b0`.
- This bit is not valid and reads unknown if an External abort on a Atomic access, reported with ESR_ELx.WU == `0b00`.
- Otherwise RW.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

DFSC, bits [5:0]
When FEAT_RAS is implemented:

Data Fault Status Code.

DFSC	Meaning
`0b000000`	Uncategorized error.
`0b010001`	Asynchronous SError interrupt.

All other values are reserved.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

ISS encoding for an exception from a Breakpoint or Vector Catch debug exception

24–6	5–0
RES0	IFSC

Bits [24:6]

Reserved, res0.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning
`0b100010`	Debug exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

For more information about generating these exceptions:

- For exceptions from AArch64, see 'Breakpoint exceptions'.
- For exceptions from AArch32, see 'Breakpoint exceptions' and 'Vector Catch exceptions'.

ISS encoding for an exception from a Software Step exception

24	23–7	6	5–0
ISV	RES0	EX	IFSC

ISV, bit [24]

Instruction syndrome valid. Indicates whether the EX bit, ISS[6], is valid, as follows:

ISV	Meaning
`0b0`	EX bit is res0.
`0b1`	EX bit is valid.

See the EX bit description for more information.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bits [23:7]

Reserved, res0.

EX, bit [6]

Exclusive operation. If the ISV bit is set to 1, this bit indicates whether a Load-Exclusive instruction was stepped.

EX	Meaning
0b0	An instruction other than a Load-Exclusive instruction was stepped.
0b1	A Load-Exclusive instruction was stepped.

If the ISV bit is set to 0, this bit is res0, indicating no syndrome data is available.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

IFSC, bits [5:0]

Instruction Fault Status Code.

IFSC	Meaning
0b100010	Debug exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

For more information about generating these exceptions, see 'Software Step exceptions'.

ISS encoding for an exception from a Watchpoint exception

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
RES0	WPT	WPTV	WPF	FnP	RES0	VNCR	RES0	FnV	RES0	CM	RES0	WnR	DFSC

Bit [24]

Reserved, res0.

WPT, bits [23:18]
When FEAT_Debugv8p2 is implemented:

Watchpoint number.

All other values are reserved.

Otherwise:

Reserved, res0.

WPTV, bit [17]
When FEAT_SME is implemented or FEAT_Debugv8p2 is implemented:

Watchpoint number Valid.

WPTV	Meaning	Applies when
`0b0`	The WPT field is invalid, and holds an unknown value.	When FEAT_Debugv8p9 is not implemented
`0b1`	The WPT field is valid, and holds the number of a watchpoint that triggered a Watchpoint exception.	

When a Watchpoint exception is triggered by a watchpoint match:

- If the PE sets any of FnV, FnP, or WPF to 1, then the PE sets WPTV to 1.
- If the PE sets all of FnV, FnP, and WPF to 0, then the PE sets WPTV to an implementation defined value, 0 or 1.

Otherwise:

Reserved, res0.

WPF, bit [16]

Watchpoint might be false-positive.

WPF	Meaning	Applies when
`0b0`	The watchpoint matched the original address of the access or set of contiguous accesses.	

0b1	The watchpoint matched an access or set of contiguous accesses where the lowest accessed address was rounded down to the nearest multiple of 16 bytes and the highest accessed address was rounded up to the nearest multiple of 16 bytes minus 1, but the watchpoint might not have matched the original address of the access or set of contiguous accesses.	When FEAT_SVE is implemented or FEAT_SME is implemented

FnP, bit [15]

FAR not Precise.

This field only has meaning if the FAR is valid; that is, when the FnV field is 0. If the FnV field is 1, the FnP field is 0.

FnP	Meaning	Applies when

0b0	If the FnV field is 0, the FAR holds the virtual address of an access or set of contiguous accesses that triggered a Watchpoint exception.	
0b1	The FAR holds any address within the smallest implemented translation granule that contains the virtual address of an access or set of contiguous accesses that triggered a Watchpoint exception.	When FEAT_SVE is implemented or FEAT_SME is implemented

Bit [14]

Reserved, res0.

VNCR, bit [13]

Indicates that the watchpoint came from use of VNCR_EL2 register by EL1 code.

VNCR	Meaning	Applies when
0b0	The watchpoint was not generated by the use of VNCR_EL2 by EL1 code.	

0b1	The watchpoint was generated by the use of VNCR_EL2 by EL1 code.	When FEAT_NV2 is implemented

This field is 0 in ESR_EL1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bits [12:11]

Reserved, res0.

FnV, bit [10]

FAR not Valid.

FnV	Meaning	Applies when
0b0	The FAR is valid, and its value is as described by the FnP field.	
0b1	The FAR is invalid, and holds an unknown value.	When FEAT_SVE is implemented or FEAT_SME is implemented

Bit [9]

Reserved, res0.

CM, bit [8]

Cache maintenance. Indicates whether the Watchpoint exception came from a cache maintenance instruction:

CM	Meaning

0b0	The Watchpoint exception was not generated by the execution of one of the System instructions identified in the description of value 1.
0b1	The Watchpoint exception was generated by the execution of a cache maintenance instruction. The DC ZVA, DC GVA, and DC GZVA instructions are not classified as a cache maintenance instructions, and therefore their execution does not cause this field to be set to 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bit [7]

Reserved, res0.

WnR, bit [6]

Write not Read. Indicates whether the Watchpoint exception was caused by an instruction writing to a memory location, or by an instruction reading from a memory location.

WnR	Meaning
0b0	Watchpoint exception caused by an instruction reading from a memory location.
0b1	Watchpoint exception caused by an instruction writing to a memory location.

For Watchpoint exceptions on cache maintenance instructions, this bit always returns a value of 1.

For Watchpoint exceptions from an atomic instruction, this field is set to 0 if a read of the location would have generated the Watchpoint exception, otherwise it is set to 1.

If multiple watchpoints match on the same access, it is unpredictable which watchpoint generates the Watchpoint exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

DFSC, bits [5:0]

Data Fault Status Code.

DFSC	Meaning
`0b100010`	Debug exception.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

For more information about generating these exceptions, see 'Watchpoint exceptions'.

ISS encoding for an exception from execution of a Breakpoint instruction

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
RES0	Comment

Bits [24:16]

Reserved, res0.

Comment, bits [15:0]

Set to the instruction comment field value, zero extended as necessary.

For the AArch32 BKPT instructions, the comment field is described as the immediate field.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

For more information about generating these exceptions, see 'Breakpoint instruction exceptions'.

ISS encoding for an exception from a TSTART instruction

24–10	9–5	4–0
RES0	Rd	RES0

Bits [24:10]

Reserved, res0.

Rd, bits [9:5]

The Rd value from the issued instruction, the general purpose register used for the destination.

Bits [4:0]

Reserved, res0.

ISS encoding for an exception from Branch Target Identification instruction

24–2	1–0
RES0	BTYPE

Bits [24:2]

Reserved, res0.

BTYPE, bits [1:0]

This field is set to the PSTATE.BTYPE value that generated the Branch Target Exception.

For more information about generating these exceptions, see 'The AArch64 application level programmers model'.

ISS encoding for an exception from a Pointer Authentication instruction authentication failure

24–2	1	0
RES0	Exception as a result of an Instruction key or a Data key	Exception as a result of an A key or a B key

Bits [24:2]

Reserved, res0.

Bit [1]

This field indicates whether the exception is as a result of an Instruction key or a Data key.

	Meaning
`0b0`	Instruction Key.
`0b1`	Data Key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

Bit [0]

This field indicates whether the exception is as a result of an A key or a B key.

	Meaning
`0b0`	A key.
`0b1`	B key.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

The following instructions generate an exception when the Pointer Authentication Code (PAC) is incorrect:

- `AUTIASP`, `AUTIAZ`, `AUTIA1716`.
- `AUTIBSP`, `AUTIBZ`, `AUTIB1716`.
- `AUTIA`, `AUTDA`, `AUTIB`, `AUTDB`.
- `AUTIZA`, `AUTIZB`, `AUTDZA`, `AUTDZB`.

It is implementation defined whether the following instructions generate an exception directly from the authorization failure, rather than changing the address in a way that will generate a Translation fault when the address is accessed:

- `RETAA`, `RETAB`.
- `BRAA`, `BRAB`, `BLRAA`, `BLRAB`.
- `BRAAZ`, `BRABZ`, `BLRAAZ`, `BLRABZ`.
- `ERETAA`, `ERETAB`.

- `LDRAA`, `LDRAB`, whether the authenticated address is written back to the base register or not.

Accessing ESR_EL1

When HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic ESR_EL1 or ESR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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

MRS <Xt>, ESR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HFGRTR_EL2.ESR_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        X[t, 64] = NVMem[0x138];
    else
        X[t, 64] = ESR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = ESR_EL2;
    else
        X[t, 64] = ESR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ESR_EL1;
```

MSR ESR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HFGWTR_EL2.ESR_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        NVMem[0x138] = X[t, 64];
    else
        ESR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        ESR_EL2 = X[t, 64];
    else
        ESR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ESR_EL1 = X[t, 64];
```

MRS <Xt>, ESR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101'
then
        X[t, 64] = NVMem[0x138];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = ESR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) &&
HCR_EL2.E2H == '1' then
        X[t, 64] = ESR_EL1;
    else
        UNDEFINED;
```

MSR ESR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101'
then
        NVMem[0x138] = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        ESR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) &&
HCR_EL2.E2H == '1' then
        ESR_EL1 = X[t, 64];
    else
        UNDEFINED;
```

MRS <Xt>, ESR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        X[t, 64] = ESR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    X[t, 64] = ESR_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = ESR_EL2;
```

MSR ESR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0101	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        ESR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    ESR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    ESR_EL2 = X[t, 64];
```


28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94