RMR_EL2, Reset Management Register (EL2)

The RMR_EL2 characteristics are:

Purpose

When this register is implemented:

- A write to the register at EL2 can request a Warm reset.
- If EL2 can use all Execution states, this register specifies the Execution state that the PE boots into on a Warm reset.

Configuration

AArch64 System register RMR_EL2 bits [31:0] are architecturally mapped to AArch32 System register HRMR[31:0] when the highest implemented Exception level is EL2.

This register is present only when the highest implemented Exception level is EL2. Otherwise, direct accesses to RMR_EL2 are undefined.

When EL2 is the highest implemented Exception level:

- If EL2 can use all Execution states then this register must be implemented.
- If EL2 cannot use AArch32 then it is implementation defined whether the register is implemented.

Attributes

RMR_EL2 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																															

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
RES0																														RR	AA64

Bits [63:2]

Reserved, res0.

RR, bit [1]

Reset Request. Setting this bit to 1 requests a Warm reset.

The reset behavior of this field is:

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

AA64, bit [0]
When EL2 is capable of using AArch32:

When EL2 can use AArch32, determines which Execution state the PE boots into after a Warm reset:

AA64	Meaning
`0b0`	AArch32.
`0b1`	AArch64.

On coming out of the Warm reset, execution starts at the implementation defined reset vector address of the specified Execution state.

If EL2 can only use AArch64 state, this bit is RAO/WI.

When implemented as a RW field, this field resets to 1 on a Cold reset.

Otherwise:

Reserved, RAO/WI.

Accessing RMR_EL2

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

MRS <Xt>, RMR_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0000	0b010

```
if PSTATE.EL == EL1 && EL2Enabled() &&
IsHighestEL(EL2) && HCR_EL2.NV == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elsif PSTATE.EL == EL2 && IsHighestEL(EL2) then
    X[t, 64] = RMR_EL2;
else
    UNDEFINED;
```

MSR RMR_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b0000	0b010

```
if PSTATE.EL == EL1 && EL2Enabled() &&
IsHighestEL(EL2) && HCR_EL2.NV == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elsif PSTATE.EL == EL2 && IsHighestEL(EL2) then
    RMR_EL2 = X[t, 64];
else
    UNDEFINED;
```

AArch32 Registers | AArch64 Registers | AArch32 Instructions | AArch64 Instructions | Index by Encoding | External Registers

28/03/2023 16:01; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94