MAIR_EL1, Memory Attribute Indirection Register (EL1)

The MAIR_EL1 characteristics are:

Purpose

Provides the memory attribute encodings corresponding to the possible AttrIndx values in a Long-descriptor format translation table entry for stage 1 translations at EL1.

Configuration

AArch64 System register MAIR_EL1 bits [31:0] are architecturally mapped to AArch32 System register PRRR[31:0] when TTBCR.EAE == 0.

AArch64 System register MAIR_EL1 bits [31:0] are architecturally mapped to AArch32 System register MAIR0[31:0] when TTBCR.EAE == 1.

AArch64 System register MAIR_EL1 bits [63:32] are architecturally mapped to AArch32 System register NMRR[31:0] when TTBCR.EAE == 0.

AArch64 System register MAIR_EL1 bits [63:32] are architecturally mapped to AArch32 System register MAIR1[31:0] when TTBCR.EAE == 1.

Attributes

MAIR_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
Attr7	Attr6	Attr5	Attr4
Attr3	Attr2	Attr1	Attr0
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

MAIR_EL1 is permitted to be cached in a TLB.

Attr<n>, bits [8n+7:8n], for n = 7 to 0

Memory Attribute encoding.

When FEAT_AIE is implemented and stage 1 Attributes Index Extension is enabled and AttrIndx[3] in a Long descriptor format translation table entry is 0, or when FEAT_AIE is not implemented, AttrIndx[2:0] gives the value of <n> in Attr<n>.

When FEAT_AIE is implemented and stage 1 Attributes Index Extension is enabled and AttrIndx[3] in a Long descriptor format translation table entry is 1, see MAIR2_ELx.Attr

Attr is encoded as follows:

Attr	Meaning
0b0000dd00	Device memory. See encoding ofÂ 'dd' for the type of Device memory.
0b0000dd01	If FEAT_XS is implemented: Device memory with the XS attribute set to 0. See encoding ofÂ 'dd' for the type of Device memory. Otherwise,Â unpredictable.
0b0000dd1x	unpredictable.
0booooiiii, (oooo != 0000 and iiii != 0000)	Normal memory. See encoding ofÂ 'oooo' andÂ 'iiii' for the type of Normal Memory.
`0b01000000`	If FEAT_XS is implemented: Normal Inner Non-cacheable, Outer Non-cacheable memory with the XS attribute set to 0. Otherwise,Â unpredictable.
`0b10100000`	If FEAT_XS is implemented: Normal Inner Write-through Cacheable, Outer Write-through Cacheable, Read-Allocate, No-Write Allocate, Non-transient memory with the XS attribute set to 0. Otherwise,Â unpredictable.
`0b11110000`	If FEAT_MTE2 is implemented: Tagged Normal Inner Write-Back, Outer Write-Back, Read-Allocate, Write-Allocate Non-transient memory. Otherwise,Â unpredictable.

Attr	Meaning
`0bxxxx0000`, where xxxx != 0000 and xxxx != 0100 and xxxx != 1010 and xxxx != 1111	unpredictable.

'dd' is encoded as follows:

dd	Meaning
`0b00`	Device-nGnRnE memory
`0b01`	Device-nGnRE memory
`0b10`	Device-nGRE memory
`0b11`	Device-GRE memory

'oooo' is encoded as follows:

'oooo'	Meaning
`0b0000`	See encoding of Attr
0b00RW, RW notÂ `0b00`	Normal memory, Outer Write-Through Transient
`0b0100`	Normal memory, Outer Non-cacheable
0b01RW, RW notÂ `0b00`	Normal memory, Outer Write-Back Transient
0b10RW	Normal memory, Outer Write-Through Non-transient
0b11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

'iiii' is encoded as follows:

'iiii'	Meaning
`0b0000`	See encoding of Attr
0b00RW, RW notÂ `0b00`	Normal memory, Inner Write-Through Transient
`0b0100`	Normal memory, Inner Non-cacheable
0b01RW, RW notÂ `0b00`	Normal memory, Inner Write-Back Transient

'iiii'	Meaning
0b10RW	Normal memory, Inner Write-Through Non-transient
0b11RW	Normal memory, Inner Write-Back Non-transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in 'oooo' and 'iiii' fields have the following meanings:

R or W	Meaning
0b0	No Allocate
0b1	Allocate

When FEAT_XS is implemented, stage 1 Inner Write-Back Cacheable, Outer Write-Back Cacheable memory types have the XS attribute set to 0.

The reset behavior of this field is:

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

Accessing MAIR_EL1

When HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic MAIR_EL1 or MAIR_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>, MAIR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	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.MAIR_EL1 == '1'
```

```
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        X[t, 64] = NVMem[0x140];
    else
        X[t, 64] = MAIR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        X[t, 64] = MAIR_EL2;
    else
        X[t, 64] = MAIR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MAIR_EL1;
```

MSR MAIR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	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.MAIR_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        NVMem[0x140] = X[t, 64];
    else
        MAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        MAIR_EL2 = X[t, 64];
    else
        MAIR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MAIR_EL1 = X[t, 64];
```

MRS <Xt>, MAIR_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	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[0x140];
    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] = MAIR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) &&
HCR_EL2.E2H == '1' then
        X[t, 64] = MAIR_EL1;
    else
        UNDEFINED;
```

MSR MAIR_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1010	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101'
then
        NVMem[0x140] = 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
        MAIR_EL1 = X[t, 64];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) &&
HCR_EL2.E2H == '1' then
        MAIR_EL1 = X[t, 64];
    else
        UNDEFINED;
```