FPSR, Floating-point Status Register

The FPSR characteristics are:

Purpose

Provides floating-point system status information.

Configuration

AArch64 System register FPSR bits [31:27] are architecturally mapped to AArch32 System register FPSCR[31:27].

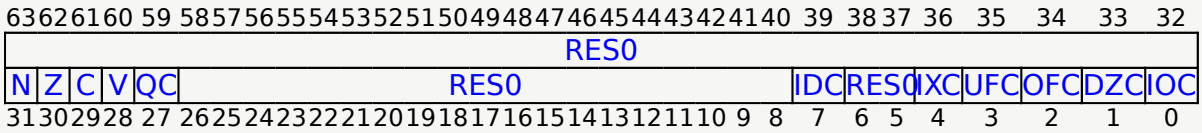
AArch64 System register FPSR bit [7] is architecturally mapped to AArch32 System register FPSCR[7].

AArch64 System register FPSR bits [4:0] are architecturally mapped to AArch32 System register FPSCR[4:0].

Attributes

FPSR is a 64-bit register.

Field descriptions



Bits [63:32]

Reserved, res0.

N, bit [31]
When AArch32 is supported and AArch32 floating-point is implemented:

Negative condition flag for AArch32 floating-point comparison operations.

Note

AArch64 floating-point comparisons set the PSTATE.N flag instead.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

Z, bit [30]
When AArch32 is supported and AArch32 floating-point is implemented:

Zero condition flag for AArch32 floating-point comparison operations.

Note

AArch64 floating-point comparisons set the PSTATE.Z flag instead.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

C, bit [29]
When AArch32 is supported and AArch32 floating-point is implemented:

Carry condition flag for AArch32 floating-point comparison operations.

Note

AArch64 floating-point comparisons set the PSTATE.C flag instead.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

V, bit [28]
When AArch32 is supported and AArch32 floating-point is implemented:

Overflow condition flag for AArch32 floating-point comparison operations.

Note

AArch64 floating-point comparisons set the PSTATE.V flag instead.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

QC, bit [27]

Cumulative saturation bit, Advanced SIMD only. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated since 0 was last written to this bit.

The reset behavior of this field is:

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

Bits [26:8]

Reserved, res0.

IDC, bit [7]

Input Denormal cumulative floating-point exception bit. This bit is set to 1 to indicate that the Input Denormal floating-point exception has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the FPCR.IDE bit. This bit is set to 1 to indicate a floating-point exception only if FPCR.IDE is 0.

The reset behavior of this field is:

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

Bits [6:5]

Reserved, res0.

IXC, bit [4]

Inexact cumulative floating-point exception bit. This bit is set to 1 to indicate that the Inexact floating-point exception has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the FPCR.IXE bit. This bit is set to 1 to indicate a floating-point exception only if FPCR.IXE is 0.

The criteria for the Inexact floating-point exception to occur are affected by whether denormalized numbers are flushed to zero and by the value of the FPCR.AH bit. For more information, see 'Floating-point exceptions and exception traps'.

The reset behavior of this field is:

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

UFC, bit [3]

Underflow cumulative floating-point exception bit. This bit is set to 1 to indicate that the Underflow floating-point exception has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the FPCR.UFE bit. This bit is set to 1 to indicate a floating-point exception only if FPCR.UFE is 0 or if flushing denormalized numbers to zero is enabled.

The criteria for the Underflow floating-point exception to occur are affected by whether denormalized numbers are flushed to zero and by the value of the FPCR.AH bit. For more information, see 'Floating-point exceptions and exception traps'.

The reset behavior of this field is:

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

OFC, bit [2]

Overflow cumulative floating-point exception bit. This bit is set to 1 to indicate that the Overflow floating-point exception has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the FPCR.OFE bit. This bit is set to 1 to indicate a floating-point exception only if FPCR.OFE is 0.

The reset behavior of this field is:

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

DZC, bit [1]

Divide by Zero cumulative floating-point exception bit. This bit is set to 1 to indicate that the Divide by Zero floating-point exception has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the FPCR.DZE bit. This bit is set to 1 to indicate a floating-point exception only if FPCR.DZE is 0.

The reset behavior of this field is:

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

IOC, bit [0]

Invalid Operation cumulative floating-point exception bit. This bit is set to 1 to indicate that the Invalid Operation floating-point exception has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the FPCR.IOE bit. This bit is set to 1 to indicate a floating-point exception only if FPCR.IOE is 0.

The criteria for the Invalid Operation floating-point exception to occur are affected by the value of the FPCR.AH bit. For more information, see 'Floating-point exceptions and exception traps'.

The reset behavior of this field is:

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

Accessing FPSR

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

MRS <Xt>, FPSR

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b001

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> ==
'11') && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' &&
CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' &&
CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPSR;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif CPACR_EL1.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' &&
CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' &&
CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPSR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
```

```
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN ==
'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPSR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPSR;
```

MSR FPSR, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b001

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> ==
'11') && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' &&
CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' &&
CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPSR = X[t, 64];
elsif PSTATE.EL == EL1 then
```

```
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif CPACR_EL1.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' &&
CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' &&
CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPSR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN ==
'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPSR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPSR = X[t, 64];
```


28/03/2023 16:01; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94