MVFR0_EL1, AArch32 Media and VFP Feature Register 0

The MVFR0_EL1 characteristics are:

Purpose

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with MVFR1_EL1 and MVFR2_EL1.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configuration

AArch64 System register MVFR0_EL1 bits [31:0] are architecturally mapped to AArch32 System register MVFR0[31:0].

In an implementation where at least one Exception level supports execution in AArch32 state, but there is no support for Advanced SIMD and floating-point operation, this register is RAZ.

Attributes

MVFR0_EL1 is a 64-bit register.

Field descriptions

When AArch32 is supported:

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

FPRound	FPShVec	FPSqrt	FPDivide	FPTrap	FPDP	FPSP	SIMDReg
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

Bits [63:32]

Reserved, res0.

FPRound, bits [31:28]

Floating-Point Rounding modes. Indicates whether the floating-point implementation provides support for rounding modes. Defined values are:

FPRound	Meaning
`0b0000`	Not implemented, or only Round to Nearest mode supported, except that Round towards Zero mode is supported for VCVT instructions that always use that rounding mode regardless of the FPSCR setting.
`0b0001`	All rounding modes supported.

All other values are reserved.

In Armv8-A the permitted values are `0b0000` and `0b0001`.

FPShVec, bits [27:24]

Short Vectors. Indicates whether the floating-point implementation provides support for the use of short vectors. Defined values are:

FPShVec	Meaning
`0b0000`	Short vectors not supported.
`0b0001`	Short vector operation supported.

All other values are reserved.

In Armv8-A the only permitted value is `0b0000`.

FPSqrt, bits [23:20]

Square Root. Indicates whether the floating-point implementation provides support for the ARMv6 VFP square root operations. Defined values are:

FPSqrt	Meaning
`0b0000`	Not supported in hardware.
`0b0001`	Supported.

All other values are reserved.

In Armv8-A the permitted values are `0b0000` and `0b0001`.

The VSQRT.F32 instruction also requires the single-precision floating-point attribute, bits [7:4], and the VSQRT.F64 instruction also requires the double-precision floating-point attribute, bits [11:8].

FPDivide, bits [19:16]

Indicates whether the floating-point implementation provides support for VFP divide operations. Defined values are:

FPDivide	Meaning
`0b0000`	Not supported in hardware.
`0b0001`	Supported.

All other values are reserved.

In Armv8-A the permitted values are `0b0000` and `0b0001`.

The VDIV.F32 instruction also requires the single-precision floating-point attribute, bits [7:4], and the VDIV.F64 instruction also requires the double-precision floating-point attribute, bits [11:8].

FPTrap, bits [15:12]

Floating Point Exception Trapping. Indicates whether the floating-point implementation provides support for exception trapping. Defined values are:

FPTrap	Meaning
`0b0000`	Not supported.
`0b0001`	Supported.

All other values are reserved.

A value of `0b0001` indicates that, when the corresponding trap is enabled, a floating-point exception generates an exception.

FPDP, bits [11:8]

Double Precision. Indicates whether the floating-point implementation provides support for double-precision operations. Defined values are:

FPDP	Meaning
`0b0000`	Not supported in hardware.
`0b0001`	Supported, VFPv2.
`0b0010`	Supported, VFPv3, VFPv4, or Armv8. VFPv3 and Armv8 add an instruction to load a double-precision floating-point constant, and conversions between double-precision and fixed-point values.

All other values are reserved.

In Armv8-A the permitted values are `0b0000` and `0b0010`.

A value of `0b0001` or `0b0010` indicates support for all VFP double-precision instructions in the supported version of VFP, except that, in addition to this field being nonzero:

- VSQRT.F64 is only available if the Square root field is `0b0001`.
- VDIV.F64 is only available if the Divide field is `0b0001`.
- Conversion between double-precision and single-precision is only available if the single-precision field is nonzero.

FPSP, bits [7:4]

Single Precision. Indicates whether the floating-point implementation provides support for single-precision operations. Defined values are:

FPSP	Meaning
`0b0000`	Not supported in hardware.
`0b0001`	Supported, VFPv2.
`0b0010`	Supported, VFPv3 or VFPv4. VFPv3 adds an instruction to load a single-precision floating-point constant, and conversions between single-precision and fixed-point values.

All other values are reserved.

In Armv8-A the permitted values are `0b0000` and `0b0010`.

A value of `0b0001` or `0b0010` indicates support for all VFP single-precision instructions in the supported version of VFP, except that, in addition to this field being nonzero:

- VSQRT.F32 is only available if the Square root field is `0b0001`.
- VDIV.F32 is only available if the Divide field is `0b0001`.
- Conversion between double-precision and single-precision is only available if the double-precision field is nonzero.

SIMDReg, bits [3:0]

Advanced SIMD registers. Indicates whether the Advanced SIMD and floating-point implementation provides support for the Advanced SIMD and floating-point register bank. Defined values are:

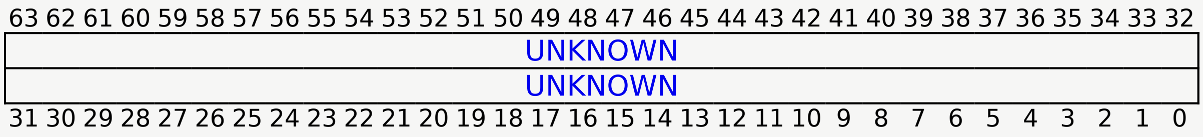
SIMDReg	Meaning
`0b0000`	The implementation has no Advanced SIMD and floating-point support.

`0b0001`	The implementation includes floating-point support with 16 x 64-bit registers.
`0b0010`	The implementation includes Advanced SIMD and floating-point support with 32 x 64-bit registers.

All other values are reserved.

In Armv8-A the permitted values are `0b0000` and `0b0010`.

Otherwise:



Bits [63:0]

Reserved, unknown.

Accessing MVFR0_EL1

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

```
MRS <Xt>, MVFR0_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	0b000

```
if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_IDST) then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        X[t, 64] = MVFR0_EL1;
elsif PSTATE.EL == EL2 then
    X[t, 64] = MVFR0_EL1;
elsif PSTATE.EL == EL3 then
```

```
    X[t, 64] = MVFR0_EL1;
```