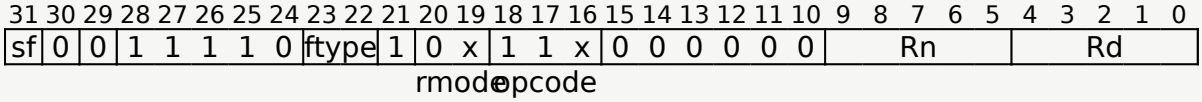
FMOV (general)

Floating-point Move to or from general-purpose register without conversion. This instruction transfers the contents of a SIMD&FP register to a general-purpose register, or the contents of a general-purpose register to a SIMD&FP register.

Depending on the settings in the *CPACR_EL1*, *CPTR_EL2*, and *CPTR_EL3* registers, and the current Security state and Exception level, an attempt to execute the instruction might be trapped.



Half-precision to 32-bit (sf == 0 && ftype == 11 && rmode == 00 && opcode == 110)
(FEAT_FP16)

```
FMOV <Wd>, <Hn>
```

Half-precision to 64-bit (sf == 1 && ftype == 11 && rmode == 00 && opcode == 110)
(FEAT_FP16)

```
FMOV <Xd>, <Hn>
```

32-bit to half-precision (sf == 0 && ftype == 11 && rmode == 00 && opcode == 111)
(FEAT_FP16)

```
FMOV <Hd>, <Wn>
```

32-bit to single-precision (sf == 0 && ftype == 00 && rmode == 00 && opcode == 111)

```
FMOV <Sd>, <Wn>
```

Single-precision to 32-bit (sf == 0 && ftype == 00 && rmode == 00 && opcode == 110)

```
FMOV <Wd>, <Sn>
```

64-bit to half-precision (sf == 1 && ftype == 11 && rmode == 00 && opcode == 111)
(FEAT_FP16)

```
FMOV <Hd>, <Xn>
```

64-bit to double-precision (sf == 1 && ftype == 01 && rmode == 00 && opcode == 111)

```
FMOV <Dd>, <Xn>
```

64-bit to top half of 128-bit (sf == 1 && ftype == 10 && rmode == 01 && opcode == 111)

```
FMOV <Vd>.D[1], <Xn>
```

Double-precision to 64-bit (sf == 1 && ftype == 01 && rmode == 00 && opcode == 110)

```
FMOV <Xd>, <Dn>
```

Top half of 128-bit to 64-bit (sf == 1 && ftype == 10 && rmode == 01 && opcode == 110)

```
FMOV <Xd>, <Vn>.D[1]
```

```
if ftype == '10' && opcode<2:1>:rmode != '11 01' then UNDEFINED;
if ftype == '11' && !IsFeatureImplemented(FEAT_FP16) then UNDEFINED;

integer d = UInt(Rd);
integer n = UInt(Rn);

constant integer intsize = 32 << UInt(sf);
constant integer decode_fltsize = if ftype == '10' then 64 else (8 << UI
FPConvOp op;
FPRounding rounding;
boolean unsigned;
integer part;

case opcode<2:1>:rmode of
    when '00 xx'    // FCVT[NPMZ][US]
        rounding = FPDecodeRounding(rmode);
        unsigned = (opcode<0> == '1');
        op = FPConvOp_CVT_FtoI;
    when '01 00'    // [US]CVTF
        rounding = FPRoundingMode(FPCR[]);
        unsigned = (opcode<0> == '1');
        op = FPConvOp_CVT_ItoF;
    when '10 00'    // FCVTA[US]
        rounding = FPRounding_TIEAWAY;
        unsigned = (opcode<0> == '1');
        op = FPConvOp_CVT_FtoI;
    when '11 00'    // FMOV
        if decode_fltsize != 16 && decode_fltsize != intsize then UNDEFIN
        op = if opcode<0> == '1' then FPConvOp_MOV_ItoF else FPConvOp_M
        part = 0;
    when '11 01'    // FMOV D[1]
        if intsize != 64 || ftype != '10' then UNDEFINED;
        op = if opcode<0> == '1' then FPConvOp_MOV_ItoF else FPConvOp_M
        part = 1;
    when '11 11'    // FJCVTZS
```

```
        if !IsFeatureImplemented(FEAT_JSCVT) then UNDEFINED;
        rounding = FPRounding_ZERO;
        unsigned = (opcode<0> == '1');
        op = FPConvOp_CVT_FtoI_JS;
    otherwise
        UNDEFINED;
```

Assembler Symbols

<Dd>	Is the 64-bit name of the SIMD&FP destination register, encoded in the "Rd" field.
<Hd>	Is the 16-bit name of the SIMD&FP destination register, encoded in the "Rd" field.
<Sd>	Is the 32-bit name of the SIMD&FP destination register, encoded in the "Rd" field.
<Wn>	Is the 32-bit name of the general-purpose source register, encoded in the "Rn" field.
<Vd>	Is the name of the SIMD&FP destination register, encoded in the "Rd" field.
<Xn>	Is the 64-bit name of the general-purpose source register, encoded in the "Rn" field.
<Wd>	Is the 32-bit name of the general-purpose destination register, encoded in the "Rd" field.
<Sn>	Is the 32-bit name of the SIMD&FP source register, encoded in the "Rn" field.
<Xd>	Is the 64-bit name of the general-purpose destination register, encoded in the "Rd" field.
<Vn>	Is the name of the SIMD&FP source register, encoded in the "Rn" field.
<Hn>	Is the 16-bit name of the SIMD&FP source register, encoded in the "Rn" field.
<Dn>	Is the 64-bit name of the SIMD&FP source register, encoded in the "Rn" field.

Operation

```
if op == FPConvOp_CVT_FtoI_JS then
    CheckFPAdvSIMDEnabled64();
else
    CheckFPEnabled64();

FPCRType fpcr = FPCR[];
constant boolean merge = IsMerging(fpcr);
constant integer fltsize = if op == FPConvOp_CVT_ItoF && merge then 128
bits(fltsize) fltval;
bits(intsize) intval;
```

```
case op of
    when FPConvOp_CVT_FtoI
        fltval = V[n, fltsize];
        intval = FPToFixed(fltval, 0, unsigned, fpcr, rounding, intsize)
        X[d, intsize] = intval;
    when FPConvOp_CVT_ItoF
        intval = X[n, intsize];
        fltval = if merge then V[d, fltsize] else Zeros(fltsize);
        Elem[fltval, 0, decode_fltsize] = FixedToFP(intval, 0, unsigned,
        V[d, fltsize] = fltval;
    when FPConvOp_MOV_FtoI
        fltval = Vpart[n, part, fltsize];
        intval = ZeroExtend(fltval, intsize);
        X[d, intsize] = intval;
    when FPConvOp_MOV_ItoF
        intval = X[n, intsize];
        fltval = intval<fltsize-1:0>;
        Vpart[d, part, fltsize] = fltval;
    when FPConvOp_CVT_FtoI_JS
        bit z;
        fltval = V[n, fltsize];
        (intval, z) = FPToFixedJS(fltval, fpcr, TRUE, intsize);
        PSTATE.<N,Z,C,V> = '0':z:'00';
        X[d, intsize] = intval;
```

Operational information

If FEAT_SME is implemented and the PE is in Streaming SVE mode, then any subsequent instruction which is dependent on the general-purpose register written by this instruction might be significantly delayed.

Internal version only: isa v33.64, AdvSIMD v29.12, pseudocode no_diffs_2023_09_RC2, sve v2023-06_rel ; Build timestamp: 2023-09-18T17:56