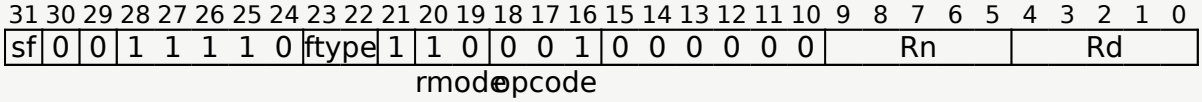
FCVTMU (scalar)

Floating-point Convert to Unsigned integer, rounding toward Minus infinity (scalar). This instruction converts the floating-point value in the SIMD&FP source register to a 32-bit or 64-bit unsigned integer using the Round towards Minus Infinity rounding mode, and writes the result to the general-purpose destination register.

A floating-point exception can be generated by this instruction. Depending on the settings in *FPCR*, the exception results in either a flag being set in *FPSR*, or a synchronous exception being generated. For more information, see *Floating-point exception traps*.

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)
(FEAT_FP16)

```
FCVTMU <Wd>, <Hn>
```

Half-precision to 64-bit (sf == 1 && ftype == 11)
(FEAT_FP16)

```
FCVTMU <Xd>, <Hn>
```

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

```
FCVTMU <Wd>, <Sn>
```

Single-precision to 64-bit (sf == 1 && ftype == 00)

```
FCVTMU <Xd>, <Sn>
```

Double-precision to 32-bit (sf == 0 && ftype == 01)

```
FCVTMU <Wd>, <Dn>
```

Double-precision to 64-bit (sf == 1 && ftype == 01)

```
FCVTMU <Xd>, <Dn>
```

```
if ftype == '10' 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
FPRounding rounding;

rounding = FPDecodeRounding(rmode);
```

Assembler Symbols

<Wd>	Is the 32-bit name of the general-purpose destination register, encoded in the "Rd" field.
<Xd>	Is the 64-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.
<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

```
CheckFPEnabled64();

FPCRType fpcr = FPCR[];
bits(decode_fltsize) fltval;
bits(intsize) intval;

fltval = V[n, decode_fltsize];
intval = FPToFixed(fltval, 0, TRUE, fpcr, rounding, intsize);
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