

## FCVTAU (vector)

Floating-point Convert to Unsigned integer, rounding to nearest with ties to Away (vector). This instruction converts each element in a vector from a floating-point value to an unsigned integer value using the Round to Nearest with Ties to Away rounding mode and writes the result to the SIMD&FP 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.

It has encodings from 4 classes: [Scalar half precision](#) , [Scalar single-precision and double-precision](#) , [Vector half precision](#) and [Vector single-precision and double-precision](#)

### Scalar half precision (FEAT\_FP16)

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |    |   |   |   |   |   |   |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|----|---|---|---|---|---|---|
| 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 |
| 0  | 1  | 1  | 1  | 1  | 1  | 0  | 0  | 1  | 1  | 1  | 1  | 0  | 0  | 1  | 1  | 1  | 0  | 0  | 1  | 0  | Rn |   |   |   | Rd |   |   |   |   |   |   |
|    |    | U  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |    |   |   |   |   |   |   |

**FCVTAU** [<Hd>](#) , [<Hn>](#)

```

if !IsFeatureImplemented(FEAT_FP16) then UNDEFINED;

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

constant integer esize = 16;
constant integer datasize = esize;
integer elements = 1;

FPRounding rounding = FPRounding\_TIEAWAY;
boolean unsigned = (U == '1');
```

### Scalar single-precision and double-precision

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |    |   |   |   |   |   |   |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|----|---|---|---|---|---|---|
| 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 |
| 0  | 1  | 1  | 1  | 1  | 1  | 0  | 0  | sz | 1  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 0  | 0  | 1  | 0  | Rn |   |   |   | Rd |   |   |   |   |   |   |
|    |    | U  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |    |   |   |   |   |   |   |

**FCVTAU** [<V><d>](#) , [<V><n>](#)

```

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

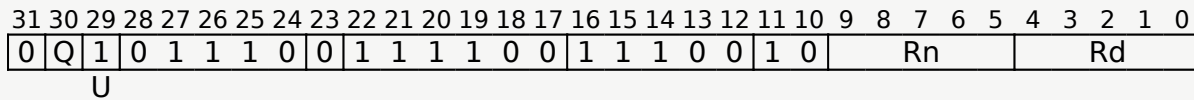
```

constant integer esize = 32 << UInt(sz);
constant integer datasize = esize;
integer elements = 1;

FPRounding rounding = FPRounding_TIEAWAY;
boolean unsigned = (U == '1');

```

## Vector half precision (FEAT\_FP16)



**FCVTAU** <Vd>.<T>, <Vn>.<T>

```

if !IsFeatureImplemented(FEAT_FP16) then UNDEFINED;

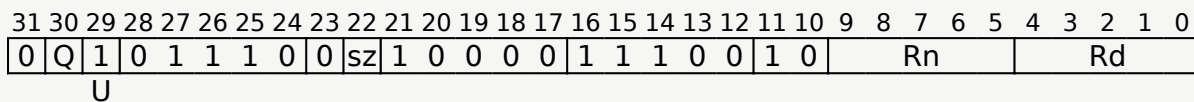
integer d = UInt(Rd);
integer n = UInt(Rn);

constant integer esize = 16;
constant integer datasize = 64 << UInt(Q);
integer elements = datasize DIV esize;

FPRounding rounding = FPRounding_TIEAWAY;
boolean unsigned = (U == '1');

```

## Vector single-precision and double-precision



**FCVTAU** <Vd>.<T>, <Vn>.<T>

```

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

if sz:Q == '10' then UNDEFINED;
constant integer esize = 32 << UInt(sz);
constant integer datasize = 64 << UInt(Q);
integer elements = datasize DIV esize;

FPRounding rounding = FPRounding_TIEAWAY;
boolean unsigned = (U == '1');

```

## Assembler Symbols

- <Hd> Is the 16-bit name of the SIMD&FP destination register, encoded in the "Rd" field.
- <Hn> Is the 16-bit name of the SIMD&FP source register, encoded in the "Rn" field.

<V>

Is a width specifier, encoded in "sz":

| sz | <V> |
|----|-----|
| 0  | S   |
| 1  | D   |

<d>

Is the number of the SIMD&FP destination register, encoded in the "Rd" field.

<n>

Is the number of the SIMD&FP source register, encoded in the "Rn" field.

<Vd>

Is the name of the SIMD&FP destination register, encoded in the "Rd" field.

<T>

For the half-precision variant: is an arrangement specifier, encoded in "Q":

| Q | <T> |
|---|-----|
| 0 | 4H  |
| 1 | 8H  |

For the single-precision and double-precision variant: is an arrangement specifier, encoded in "sz:Q":

| sz | Q | <T>      |
|----|---|----------|
| 0  | 0 | 2S       |
| 0  | 1 | 4S       |
| 1  | 0 | RESERVED |
| 1  | 1 | 2D       |

<Vn>

Is the name of the SIMD&FP source register, encoded in the "Rn" field.

## Operation

```
CheckFPAdvSIMDEnabled64();
bits(datasize) operand = V[n, datasize];

bits(esize) element;
FPCRTYPE fpcr = FPCR[];
boolean merge = elements == 1 && IsMerging(fpcr);
bits(128) result = if merge then V[d, 128] else Zeros(128);

for e = 0 to elements-1
    element = Elem[operand, e, esize];
    Elem[result, e, esize] = FPToFixed(element, 0, unsigned, fpcr, round);

V[d, 128] = result;
```

[Base  
Instructions](#)

[SIMD&FP  
Instructions](#)

[SVE  
Instructions](#)

[SME  
Instructions](#)

[Index by  
Encoding](#)

[Sh  
Pseu](#)

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

Copyright Â© 2010-2023 Arm Limited or its affiliates. All rights reserved. This  
document is Non-Confidential.