

## SQXTN, SQXTN2

Signed saturating extract Narrow. This instruction reads each vector element from the source SIMD&FP register, saturates the value to half the original width, places the result into a vector, and writes the vector to the lower or upper half of the destination SIMD&FP register. The destination vector elements are half as long as the source vector elements. All the values in this instruction are signed integer values.

If overflow occurs with any of the results, those results are saturated. If saturation occurs, the cumulative saturation bit *FPSR.QC* is set.

The *SQXTN* instruction writes the vector to the lower half of the destination register and clears the upper half, while the *SQXTN2* instruction writes the vector to the upper half of the destination register without affecting the other bits of the 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.

It has encodings from 2 classes: [Scalar](#) and [Vector](#)

### Scalar

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	0	1	1	1	1	0	size	1	0	0	0	0	1	0	1	0	0	1	0	Rn			Rd							
U																															

**SQXTN** <Vb><d>, <Va><n>

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

if size == '11' then UNDEFINED;
constant integer esize = 8 << UInt(size);
constant integer datasize = esize;
integer part = 0;
integer elements = 1;

boolean unsigned = (U == '1');
```

### Vector

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	Q	0	0	1	1	1	0	size	1	0	0	0	0	1	0	1	0	0	1	0	Rn			Rd							
U																															

**SQXTN{2}** <Vd>.<Tb>, <Vn>.<Ta>

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

if size == '11' then UNDEFINED;
```

```

constant integer esize = 8 << UInt(size);
constant integer datasize = 64;
integer part = UInt(Q);
integer elements = datasize DIV esize;

boolean unsigned = (U == '1');

```

## Assembler Symbols

2

Is the second and upper half specifier. If present it causes the operation to be performed on the upper 64 bits of the registers holding the narrower elements, and is encoded in “Q”:

Q	2
0	[absent]
1	[present]

<Vd>

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

<Tb>

Is an arrangement specifier, encoded in “size:Q”:

size	Q	<Tb>
00	0	8B
00	1	16B
01	0	4H
01	1	8H
10	0	2S
10	1	4S
11	x	RESERVED

<Vn>

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

<Ta>

Is an arrangement specifier, encoded in “size”:

size	<Ta>
00	8H
01	4S
10	2D
11	RESERVED

<Vb>

Is the destination width specifier, encoded in "size":

size	<Vb>
00	B
01	H
10	S
11	RESERVED

<d>

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

<Va>

Is the source width specifier, encoded in "size":

size	<Va>
00	H
01	S
10	D
11	RESERVED

<n>

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

## Operation

```
CheckFPAdvSIMDEnabled64();
bits(2*datasize) operand = V[n, 2*datasize];
bits(datasize) result;
bits(2*esize) element;
boolean sat;

for e = 0 to elements-1
    element = Elem[operand, e, 2*esize];
    (Elem[result, e, esize], sat) = SatQ(Int(element, unsigned), esize,
    if sat then FPSR.QC = '1';

Vpart[d, part, datasize] = result;
```

[Base  
Instructions](#)

[SIMD&FP  
Instructions](#)

[SVE  
Instructions](#)

[SME  
Instructions](#)

[Index by  
Encoding](#)

[Sh  
Pseudocode](#)

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.