

SQSHRUN, SQSHRUN2

Signed saturating Shift Right Unsigned Narrow (immediate). This instruction reads each signed integer value in the vector of the source SIMD&FP register, right shifts each value by an immediate value, saturates the result to an unsigned integer value that is half the original width, places the final result into a vector, and writes the vector to the destination SIMD&FP register. The results are truncated. For rounded results, see [SQRSHRUN](#).

The SQSHRUN instruction writes the vector to the lower half of the destination register and clears the upper half, while the SQSHRUN2 instruction writes the vector to the upper half of the destination register without affecting the other bits of the register.

If saturation occurs, the cumulative saturation bit [FPSR.QC](#) is set.

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	1	1	1	1	1	0	!= 0000				immb		1 0 0 0				0	1	Rn				Rd							
immh										op																					

SQSHRUN <Vb><d>, <Va><n>, #<shift>

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

if immh == '0000' then UNDEFINED;
if immh<3> == '1' then UNDEFINED;
constant integer esize = 8 << HighestSetBit(immh);
constant integer datasize = esize;
integer elements = 1;
integer part = 0;

integer shift = (2 * esize) - UInt(immh:immb);
boolean round = (op == '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	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	Rn						Rd					
immh										op																							

SQSHRUN{2} <Vd>.<Tb>, <Vn>.<Ta>, #<shift>

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

```

if immh == '0000' then SEE\(asimdimm\);
if immh<3> == '1' then UNDEFINED;
constant integer esize = 8 << HighestSetBit(immh);
constant integer datasize = 64;
integer part = UInt(Q);
integer elements = datasize DIV esize;

integer shift = (2 * esize) - UInt(immh:immb);
boolean round = (op == '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 “immh:Q”:

immh	Q	<Tb>
0000	x	SEE Advanced SIMD modified immediate
0001	0	8B
0001	1	16B
001x	0	4H
001x	1	8H
01xx	0	2S
01xx	1	4S
1xxx	x	RESERVED

<Vn>

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

<Ta>

Is an arrangement specifier, encoded in “immh”:

immh	<Ta>
0000	SEE Advanced SIMD modified immediate
0001	8H
001x	4S
01xx	2D
1xxx	RESERVED

<Vb>

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

immh	<Vb>
0000	RESERVED
0001	B
001x	H
01xx	S
1xxx	RESERVED

<d>

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

<Va>

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

immh	<Va>
0000	RESERVED
0001	H
001x	S
01xx	D
1xxx	RESERVED

<n>

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

<shift>

For the scalar variant: is the right shift amount, in the range 1 to the destination operand width in bits, encoded in "immh:immb":

immh	<shift>
0000	RESERVED
0001	(16-UInt (immh:immb))
001x	(32-UInt (immh:immb))
01xx	(64-UInt (immh:immb))
1xxx	RESERVED

For the vector variant: is the right shift amount, in the range 1 to the destination element width in bits, encoded in "immh:immb":

immh	<shift>
0000	SEE Advanced SIMD modified immediate
0001	(16-UInt (immh:immb))
001x	(32-UInt (immh:immb))
01xx	(64-UInt (immh:immb))
1xxx	RESERVED

Operation

```

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

for e = 0 to elements-1
    element = RShr(SInt(Elem[operand, e, 2*esize]), shift, round);
    (Elem[result, e, esize], sat) = UnsignedSatQ(element, 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.