FCVTZS (vector, fixed-point)

Floating-point Convert to Signed fixed-point, rounding toward Zero (vector). This instruction converts a scalar or each element in a vector from floating-point to fixed-point signed integer using the Round towards Zero 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 Security state and Exception level in which the instruction is executed, 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	1	0	!= 0000	immb	1	1	1	1	1	1	Rn	Rd
		U							immh									

FCVTZS <V><d>, <V><n>, #<fbits>

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

if immh IN {'000x'} || (immh IN {'001x'} && !IsFeatureImplemented(FEAT_
constant integer esize = if immh IN {'1xxx'} then 64 else if immh IN {'
constant integer datasize = esize;
integer elements = 1;

integer fracbits = (esize * 2) - UInt(immh:immb);
boolean unsigned = (U == '1');
FPRounding rounding = FPRounding_ZERO;
```

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

FCVTZS <Vd>.<T>, <Vn>.<T>, #<fbits>

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

if immh == '0000' then SEE(asimdimm);
if immh IN {'000x'} || (immh IN {'001x'} && !IsFeatureImplemented(FEAT_
if immh<3>:Q == '10' then UNDEFINED;
constant integer esize = if immh IN {'1xxx'} then 64 else if immh IN {'
constant integer datasize = 64 << UInt(Q);
```

```
integer elements = datasize DIV esize;

integer fracbits = (esize * 2) - UInt(immh:immb);
boolean unsigned = (U == '1');
FPRounding rounding = FPRounding_ZERO;
```

Assembler Symbols

<V>

Is a width specifier, encoded in “immh”:

immh	<V>
000x	RESERVED
001x	H
01xx	S
1xxx	D

<d>

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

<n>

Is the number of the first 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>

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

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

<Vn>

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

<fbits>

For the scalar variant: is the number of fractional bits, in the range 1 to the operand width, encoded in “immh:immb”:

immh	<fbits>
000x	RESERVED
001x	(32-Uint(immh:immb))
01xx	(64-UInt(immh:immb))
1xxx	(128-UInt(immh:immb))

For the vector variant: is the number of fractional bits, in the range 1 to the element width, encoded in “immh:immb”:

immh	<fbits>
0000	SEE Advanced SIMD modified immediate
0001	RESERVED
001x	(32-Uint(immh:immb))
01xx	(64-UInt(immh:immb))
1xxx	(128-UInt(immh:immb))

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, fracbits, unsigned, fpc

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