CMTST

Compare bitwise Test bits nonzero (vector). This instruction reads each vector element in the first source SIMD&FP register, performs an AND with the corresponding vector element in the second source SIMD&FP register, and if the result is not zero, sets every bit of the corresponding vector element in the destination SIMD&FP register to one, otherwise sets every bit of the corresponding vector element in the destination SIMD&FP register to zero.

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	Rm	1 0 0 0 1	1	Rn	Rd
	U								

```
        CMTST <V><d>, <V><n>, <V><m>
```

```
    integer d = UInt(Rd);
    integer n = UInt(Rn);
    integer m = UInt(Rm);
    if size != '11' then UNDEFINED;
    constant integer esize = 8 << UInt(size);
    constant integer datasize = esize;
    integer elements = 1;
    boolean and_test = (U == '0');
```

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	Rm	1 0 0 0 1	1	Rn	Rd
		U								

```
        CMTST <Vd>.<T>, <Vn>.<T>, <Vm>.<T>
```

```
    integer d = UInt(Rd);
    integer n = UInt(Rn);
    integer m = UInt(Rm);
    if size:Q == '110' then UNDEFINED;
    constant integer esize = 8 << UInt(size);
    constant integer datasize = 64 << UInt(Q);
    integer elements = datasize DIV esize;
    boolean and_test = (U == '0');
```

Assembler Symbols

\<V\>

Is a width specifier, encoded in "size":

size	\<V\>
0x	RESERVED
10	RESERVED
11	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.

\<m\>
Is the number of the second SIMD&FP source register, encoded in the "Rm" field.

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

\<T\>

Is an arrangement specifier, encoded in "size:Q":

size	Q	\<T\>
00	0	8B
00	1	16B
01	0	4H
01	1	8H
10	0	2S
10	1	4S
11	0	RESERVED
11	1	2D

\<Vn\>
Is the name of the first SIMD&FP source register, encoded in the "Rn" field.

\<Vm\>
Is the name of the second SIMD&FP source register, encoded in the "Rm" field.

Operation

```
CheckFPAdvSIMDEnabled64();
bits(datasize) operand1 = V[n, datasize];
bits(datasize) operand2 = V[m, datasize];
bits(datasize) result;
bits(esize) element1;
bits(esize) element2;
boolean test_passed;

for e = 0 to elements-1
    element1 = Elem[operand1, e, esize];
    element2 = Elem[operand2, e, esize];
```

```
        if and_test then
            test_passed = !IsZero(element1 AND element2);
        else
            test_passed = (element1 == element2);
        Elem[result, e, esize] = if test_passed then Ones(esize) else Zeros

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

Operational information

If PSTATE.DIT is 1:

- The execution time of this instruction is independent of:
  ◦ The values of the data supplied in any of its registers.
  ◦ The values of the NZCV flags.
- The response of this instruction to asynchronous exceptions does not vary based on:
  ◦ The values of the data supplied in any of its registers.
  ◦ The values of the NZCV flags.