TBX

Table vector lookup extension. This instruction reads each value from the vector elements in the index source SIMD&FP register, uses each result as an index to perform a lookup in a table of bytes that is described by one to four source table SIMD&FP registers, places the lookup result in a vector, and writes the vector to the destination SIMD&FP register. If an index is out of range for the table, the existing value in the vector element of the destination register is left unchanged. If more than one source register is used to describe the table, the first source register describes the lowest bytes of the table.

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.

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	0	0	0	Rm					0	len		1	0	0	Rn					Rd				
																			op												

Two register table (len == 01)

```
TBX <Vd>.<Ta>, { <Vn>.16B, <Vn+1>.16B }, <Vm>.<Ta>
```

Three register table (len == 10)

```
TBX <Vd>.<Ta>, { <Vn>.16B, <Vn+1>.16B, <Vn+2>.16B }, <Vm>.<Ta>
```

Four register table (len == 11)

```
TBX <Vd>.<Ta>, { <Vn>.16B, <Vn+1>.16B, <Vn+2>.16B, <Vn+3>.16B }, <V
```

Single register table (len == 00)

```
TBX <Vd>.<Ta>, { <Vn>.16B }, <Vm>.<Ta>
```

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

constant integer datasize = 64 << UInt(Q);
constant integer elements = datasize DIV 8;
constant integer regs = UInt(len) + 1;
boolean is_tbl = (op == '0');
```

Assembler Symbols

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

<Ta>	Is an arrangement specifier, encoded in “Q”:

Q	<Ta>
0	8B
1	16B

<Vn>	For the four register table, three register table and two register table variant: is the name of the first SIMD&FP table register, encoded in the "Rn" field. For the single register table variant: is the name of the SIMD&FP table register, encoded in the "Rn" field.
<Vn+1>	Is the name of the second SIMD&FP table register, encoded as "Rn" plus 1 modulo 32.
<Vn+2>	Is the name of the third SIMD&FP table register, encoded as "Rn" plus 2 modulo 32.
<Vn+3>	Is the name of the fourth SIMD&FP table register, encoded as "Rn" plus 3 modulo 32.
<Vm>	Is the name of the SIMD&FP index register, encoded in the "Rm" field.

Operation

```
CheckFPAdvSIMDEnabled64();
bits(datasize) indices = V[m, datasize];
bits(128*regs) table = Zeros(128 * regs);
bits(datasize) result;
integer index;

// Create table from registers
for i = 0 to regs-1
    table<128*i+127:128*i> = V[n, 128];
    n = (n + 1) MOD 32;

result = if is_tbl then Zeros(datasize) else V[d, datasize];
for i = 0 to elements-1
    index = UInt(Elem[indices, i, 8]);
    if index < 16 * regs then
        Elem[result, i, 8] = Elem[table, index, 8];

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.
