## C7.2.291 STUR (SIMD&FP)

Store SIMD&FP register (unscaled offset). This instruction stores a single SIMD&FP register to memory. The address that is used for the store is calculated from a base register value and an optional immediate offset.

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.


### 8-bit variant

Applies when size == 00 && opc == 00.

```
STUR <Bt>, [<Xn|SP>{, #<simm>}]
```

### 16-bit variant

Applies when size == 01 && opc == 00.

```
STUR <Ht>, [<Xn|SP>{, #<simm>}]
```

### 32-bit variant

Applies when size == 10 && opc == 00.

```
STUR <St>, [<Xn|SP>{, #<simm>}]
```

### 64-bit variant

Applies when size == 11 && opc == 00.

```
STUR <Dt>, [<Xn|SP>{, #<simm>}]
```

### 128-bit variant

Applies when size == 00 && opc == 10.

```
STUR <Qt>, [<Xn|SP>{, #<simm>}]
```

### Decode for all variants of this encoding

```
boolean wback = FALSE;
boolean postindex = FALSE;
integer scale = UInt(opc<1>:size);
if scale > 4 then UnallocatedEncoding();
bits(64) offset = SignExtend(imm9, 64);
```

## Assembler symbols

| | |
|---|---|
| <Bt> | Is the 8-bit name of the SIMD&FP register to be transferred, encoded in the "Rt" field. |
| <Dt> | Is the 64-bit name of the SIMD&FP register to be transferred, encoded in the "Rt" field. |
| <Ht> | Is the 16-bit name of the SIMD&FP register to be transferred, encoded in the "Rt" field. |
| <Qt> | Is the 128-bit name of the SIMD&FP register to be transferred, encoded in the "Rt" field. |
| <St> | Is the 32-bit name of the SIMD&FP register to be transferred, encoded in the "Rt" field. |

<Xn|SP> Is the 64-bit name of the general-purpose base register or stack pointer, encoded in the "Rn" field.

<simm> Is the optional signed immediate byte offset, in the range -256 to 255, defaulting to 0 and encoded in the "imm9" field.

## Shared decode for all encodings

```
integer n = UInt(Rn);
integer t = UInt(Rt);
AccType acctype = AccType_VEC;
MemOp memop = if opc<0> == '1' then MemOp_LOAD else MemOp_STORE;
integer datasize = 8 << scale;
```

## Operation

```
CheckFPAdvSIMDEnabled64();

bits(64) address;
bits(datasize) data;

if n == 31 then
    CheckSPAlignment();
    address = SP[];
else
    address = X[n];

if !postindex then
    address = address + offset;

case memop of
    when MemOp_STORE
        data = V[t];
        Mem[address, datasize DIV 8, acctype] = data;

    when MemOp_LOAD
        data = Mem[address, datasize DIV 8, acctype];
        V[t] = data;

if wback then
    if postindex then
        address = address + offset;
    if n == 31 then
        SP[] = address;
    else
        X[n] = address;
```