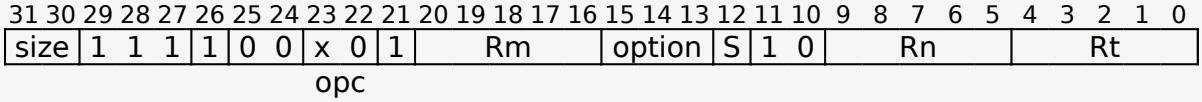
STR (register, SIMD&FP)

Store SIMD&FP register (register 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 offset register value. The offset can be optionally shifted and extended.

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 (size == 00 && opc == 00 && option != 011)

```
STR <Bt>, [<Xn|SP>, (<Wm>|<Xm>), <extend> {<amount>}]
```

8-bit (size == 00 && opc == 00 && option == 011)

```
STR <Bt>, [<Xn|SP>, <Xm>{, LSL <amount>}]
```

16-bit (size == 01 && opc == 00)

```
STR <Ht>, [<Xn|SP>, (<Wm>|<Xm>){, <extend> {<amount>}}]
```

32-bit (size == 10 && opc == 00)

```
STR <St>, [<Xn|SP>, (<Wm>|<Xm>){, <extend> {<amount>}}]
```

64-bit (size == 11 && opc == 00)

```
STR <Dt>, [<Xn|SP>, (<Wm>|<Xm>){, <extend> {<amount>}}]
```

128-bit (size == 00 && opc == 10)

```
STR <Qt>, [<Xn|SP>, (<Wm>|<Xm>){, <extend> {<amount>}}]
```

```
integer scale = UInt(opc<1>:size);
if scale > 4 then UNDEFINED;
if option<1> == '0' then UNDEFINED;    // sub-word index
ExtendType extend_type = DecodeRegExtend(option);
integer shift = if S == '1' then scale else 0;
```

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.

<Wm> When option<0> is set to 0, is the 32-bit name of the general-purpose index register, encoded in the "Rm" field.

<Xm> When option<0> is set to 1, is the 64-bit name of the general-purpose index register, encoded in the "Rm" field.

<extend>

For the 8-bit variant: is the index extend specifier, encoded in “option”:

option	<extend>
010	UXTW
110	SXTW
111	SXTX

For the 128-bit, 16-bit, 32-bit and 64-bit variant: is the index extend/shift specifier, defaulting to LSL, and which must be omitted for the LSL option when <amount> is omitted. encoded in “option”:

option	<extend>
010	UXTW
011	LSL
110	SXTW
111	SXTX

<amount> For the 8-bit variant: is the index shift amount, it must be #0, encoded in "S" as 0 if omitted, or as 1 if present.

For the 16-bit variant: is the index shift amount, optional only when <extend> is not LSL. Where it is permitted to be optional, it defaults to #0. It is encoded in “S”:

S	<amount>
0	#0
1	#1

For the 32-bit variant: is the index shift amount, optional only when <extend> is not LSL. Where it is permitted to be optional, it defaults to #0. It is encoded in “S”:

S	<amount>
0	#0
1	#2

For the 64-bit variant: is the index shift amount, optional only when <extend> is not LSL. Where it is permitted to be optional, it defaults to #0. It is encoded in “S”:

S	<amount>
0	#0
1	#3

For the 128-bit variant: is the index shift amount, optional only when <extend> is not LSL. Where it is permitted to be optional, it defaults to #0. It is encoded in “S”:

S	<amount>
0	#0
1	#4

Shared Decode

```
integer n = UInt(Rn);
integer t = UInt(Rt);
integer m = UInt(Rm);
MemOp memop = if opc<0> == '1' then MemOp_LOAD else MemOp_STORE;
constant integer datasize = 8 << scale;
boolean tagchecked = memop != MemOp_PREFETCH;
```

Operation

```
bits(64) offset = ExtendReg(m, extend_type, shift, 64);
CheckFPEnabled64();
bits(64) address;
bits(datasize) data;

AccessDescriptor accdesc = CreateAccDescASIMD(memop, FALSE, tagchecked)

if n == 31 then
    CheckSPAlignment();
    address = SP[];
else
    address = X[n, 64];

address = address + offset;

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

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

Operational information

If PSTATE.DIT is 1, the timing of this instruction is insensitive to the value of the data being loaded or stored.

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