STP (SIMD&FP)

Store Pair of SIMD&FP registers. This instruction stores a pair of SIMD&FP registers to memory. The address used for the store is calculated from a base register value and an 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.

It has encodings from 3 classes: Post-index , Pre-index and Signed offset

Post-index

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
opc	1 0 1 1	0	0 1	0	imm7	Rt2	Rn	Rt

L

32-bit (opc == 00)

```
STP <St1>, <St2>, [<Xn|SP>], #<imm>
```

64-bit (opc == 01)

```
STP <Dt1>, <Dt2>, [<Xn|SP>], #<imm>
```

128-bit (opc == 10)

```
STP <Qt1>, <Qt2>, [<Xn|SP>], #<imm>
```

```
boolean wback = TRUE;
boolean postindex = TRUE;
```

Pre-index

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
opc	1 0 1 1	0	1 1	0	imm7	Rt2	Rn	Rt

L

32-bit (opc == 00)

```
STP <St1>, <St2>, [<Xn|SP>, #<imm>]!
```

64-bit (opc == 01)

```
STP <Dt1>, <Dt2>, [<Xn|SP>, #<imm>]!
```

128-bit (opc == 10)

```
        STP <Qt1>, <Qt2>, [<Xn|SP>, #<imm>]!

    boolean wback = TRUE;
    boolean postindex = FALSE;
```

Signed offset

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
opc	1 0 1	1	0	1 0 0	imm7	Rt2	Rn	Rt

L

32-bit (opc == 00)

```
        STP <St1>, <St2>, [<Xn|SP>{, #<imm>}]
```

64-bit (opc == 01)

```
        STP <Dt1>, <Dt2>, [<Xn|SP>{, #<imm>}]
```

128-bit (opc == 10)

```
        STP <Qt1>, <Qt2>, [<Xn|SP>{, #<imm>}]
```

```
    boolean wback = FALSE;
    boolean postindex = FALSE;
```

Assembler Symbols

<Dt1>   Is the 64-bit name of the first SIMD&FP register to be
              transferred, encoded in the "Rt" field.

<Dt2>   Is the 64-bit name of the second SIMD&FP register to be
              transferred, encoded in the "Rt2" field.

<Qt1>   Is the 128-bit name of the first SIMD&FP register to be
              transferred, encoded in the "Rt" field.

<Qt2>   Is the 128-bit name of the second SIMD&FP register to be
              transferred, encoded in the "Rt2" field.

<St1>   Is the 32-bit name of the first SIMD&FP register to be
              transferred, encoded in the "Rt" field.

<St2>   Is the 32-bit name of the second SIMD&FP register to be
              transferred, encoded in the "Rt2" field.

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

<imm>   For the 32-bit post-index and 32-bit pre-index variant: is the
              signed immediate byte offset, a multiple of 4 in the range
              -256 to 252, encoded in the "imm7" field as <imm>/4.

For the 32-bit signed offset variant: is the optional signed immediate byte offset, a multiple of 4 in the range -256 to 252, defaulting to 0 and encoded in the "imm7" field as <imm>/4.

For the 64-bit post-index and 64-bit pre-index variant: is the signed immediate byte offset, a multiple of 8 in the range -512 to 504, encoded in the "imm7" field as <imm>/8.

For the 64-bit signed offset variant: is the optional signed immediate byte offset, a multiple of 8 in the range -512 to 504, defaulting to 0 and encoded in the "imm7" field as <imm>/8.

For the 128-bit post-index and 128-bit pre-index variant: is the signed immediate byte offset, a multiple of 16 in the range -1024 to 1008, encoded in the "imm7" field as <imm>/16.

For the 128-bit signed offset variant: is the optional signed immediate byte offset, a multiple of 16 in the range -1024 to 1008, defaulting to 0 and encoded in the "imm7" field as <imm>/16.

Shared Decode

```
integer n = UInt(Rn);
integer t = UInt(Rt);
integer t2 = UInt(Rt2);
if opc == '11' then UNDEFINED;
integer scale = 2 + UInt(opc);
constant integer datasize = 8 << scale;
bits(64) offset = LSL(SignExtend(imm7, 64), scale);
boolean tagchecked = wback || n != 31;
```

Operation

```
CheckFPEnabled64();
bits(64) address;
bits(datasize) data1;
bits(datasize) data2;
constant integer dbytes = datasize DIV 8;

AccessDescriptor accdesc = CreateAccDescASIMD(MemOp_STORE, FALSE, tagch

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

if !postindex then
    address = address + offset;

data1 = V[t, datasize];
```

```
    data2 = V[t2, datasize];
    Mem[address, dbytes, accdesc] = data1;
    Mem[address+dbytes, dbytes, accdesc] = data2;

if wback then
    if postindex then
        address = address + offset;
    if n == 31 then
        SP[] = address;
    else
        X[n, 64] = address;
```

Operational information

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