LDPSW

Load Pair of Registers Signed Word calculates an address from a base register value and an immediate offset, loads two 32-bit words from memory, sign-extends them, and writes them to two registers. For information about memory accesses, see *Load/Store addressing modes*.

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

LDPSW <Xt1>, <Xt2>, [<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
0	1	1	0	1	0	0	1	1	1	imm7	Rt2	Rn	Rt
opc									L				

LDPSW <Xt1>, <Xt2>, [<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
0	1	1	0	1	0	0	1	0	1	imm7	Rt2	Rn	Rt
opc									L				

LDPSW <Xt1>, <Xt2>, [<Xn|SP>{, #<imm>}]

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

For information about the constrained unpredictable behavior of this instruction, see *Architectural Constraints on UNPREDICTABLE behaviors*, and particularly *LDPSW*.

Assembler Symbols

<Xt1>	Is the 64-bit name of the first general-purpose register to be transferred, encoded in the "Rt" field.

<Xt2>	Is the 64-bit name of the second general-purpose 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 post-index and 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 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.

Shared Decode

```
integer n = UInt(Rn);
integer t = UInt(Rt);
integer t2 = UInt(Rt2);
bits(64) offset = LSL(SignExtend(imm7, 64), 2);
boolean tagchecked = wback || n != 31;

boolean rt_unknown = FALSE;
boolean wb_unknown = FALSE;

if wback && (t == n || t2 == n) && n != 31 then
    Constraint c = ConstrainUnpredictable(Unpredictable_WBOVERLAPLD);
    assert c IN {Constraint_WBSUPPRESS, Constraint_UNKNOWN, Constraint_
    case c of
        when Constraint_WBSUPPRESS wback = FALSE;     // writeback is su
        when Constraint_UNKNOWN    wb_unknown = TRUE;     // writeback i
        when Constraint_UNDEF      UNDEFINED;
        when Constraint_NOP        EndOfInstruction();

if t == t2 then
    Constraint c = ConstrainUnpredictable(Unpredictable_LDPOVERLAP);
    assert c IN {Constraint_UNKNOWN, Constraint_UNDEF, Constraint_NOP};
    case c of
        when Constraint_UNKNOWN rt_unknown = TRUE;     // result is UNKN
        when Constraint_UNDEF   UNDEFINED;
        when Constraint_NOP     EndOfInstruction();
```

Operation

```
bits(64) address;
bits(32) data1;
bits(32) data2;
boolean privileged = PSTATE.EL != EL0;

AccessDescriptor accdesc = CreateAccDescGPR(MemOp_LOAD, FALSE, privileg

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

```
    address = X[n, 64];

if !postindex then
    address = address + offset;

if IsFeatureImplemented(FEAT_LSE2) && FALSE then
    bits(64) full_data;
    accdesc.ispair = TRUE;
    full_data = Mem[address, 8, accdesc];
    if BigEndian(accdesc.acctype) then
        data2 = full_data<31:0>;
        data1 = full_data<63:32>;
    else
        data1 = full_data<31:0>;
        data2 = full_data<63:32>;
else
    data1 = Mem[address, 4, accdesc];
    data2 = Mem[address+4, 4, accdesc];
if rt_unknown then
    data1 = bits(32) UNKNOWN;
    data2 = bits(32) UNKNOWN;
X[t, 64] = SignExtend(data1, 64);
X[t2, 64] = SignExtend(data2, 64);
if wback then
    if wb_unknown then
        address = bits(64) UNKNOWN;
    elsif 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.

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