LDAPURSW

Load-Acquire RCpc Register Signed Word (unscaled) calculates an address from a base register and an immediate offset, loads a signed word from memory, sign-extends it, and writes it to a register.

The instruction has memory ordering semantics as described in *Load-Acquire, Load-AcquirePC, and Store-Release*, except that:

- There is no ordering requirement, separate from the requirements of a Load-AcquirePC or a Store-Release, created by having a Store-Release followed by a Load-AcquirePC instruction.
- The reading of a value written by a Store-Release by a Load-AcquirePC instruction by the same observer does not make the write of the Store-Release globally observed.

This difference in memory ordering is not described in the pseudocode.

For information about memory accesses, see *Load/Store addressing modes*.

Unscaled offset
(FEAT_LRCPC2)

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
1 0	0 1 1 0 0 1 1	0 0	imm9	0 0	Rn	Rt
size	opc					

```
LDAPURSW <Xt>, [<Xn|SP>{, #<simm>}]
```

```
bits(64) offset = SignExtend(imm9, 64);
```

Assembler Symbols

\<Xt>	Is the 64-bit name of the general-purpose 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

```
integer n = UInt(Rn);
integer t = UInt(Rt);

boolean tagchecked = n != 31;
```

Operation

```
bits(64) address;
bits(32) data;

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

address = address + offset;

data = Mem[address, 4, accdesc];
X[t, 64] = SignExtend(data, 64);
```

Operational information

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