LDAPURSH

Load-Acquire RCpc Register Signed Halfword (unscaled) calculates an address from a base register and an immediate offset, loads a signed halfword 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
0 1	0 1 1 0 0 1 1 x	0		imm9	0 0	Rn	Rt
size	opc						

32-bit (opc == 11)

```
        LDAPURSH <Wt>, [<Xn|SP>{, #<simm>}]
```

64-bit (opc == 10)

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

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

Assembler Symbols

<Wt>   Is the 32-bit name of the general-purpose register to be transferred, encoded in the "Rt" field.

<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);
MemOp memop;
boolean signed;
integer regsize;

if opc<1> == '0' then
    // store or zero-extending load
    memop = if opc<0> == '1' then MemOp_LOAD else MemOp_STORE;
    regsize = 32;
    signed = FALSE;
else
    // sign-extending load
    memop = MemOp_LOAD;
    regsize = if opc<0> == '1' then 32 else 64;
    signed = TRUE;

boolean tagchecked = memop != MemOp_PREFETCH && (n != 31);
```

Operation

```
bits(64) address;
bits(16) data;

AccessDescriptor accdesc;
if memop == MemOp_LOAD then
    accdesc = CreateAccDescLDAcqPC(tagchecked);
elsif memop == MemOp_STORE then
    accdesc = CreateAccDescAcqRel(memop, tagchecked);

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

address = address + offset;

case memop of
    when MemOp_STORE
        data = X[t, 16];
        Mem[address, 2, accdesc] = data;

    when MemOp_LOAD
        data = Mem[address, 2, accdesc];
        if signed then
            X[t, regsize] = SignExtend(data, regsize);
        else
            X[t, regsize] = ZeroExtend(data, regsize);

    when MemOp_PREFETCH
        Prefetch(address, t<4:0>);
```

Operational information

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