3.3.4. Address alignment

An aligned access is an operation where a word-aligned address is used for a word, or multiple word access, or where a halfword-aligned address is used for a halfword access. Byte accesses are always aligned.

There is no support for unaligned accesses on the Cortex-M0 processor. Any attempt to perform an unaligned memory access operation results in a HardFault exception.

3.4.2. LDR and STR, immediate offset

Load and Store with immediate offset.

Syntax

```
LDR Rt, [<Rn | SP> {, #imm}]

LDR<B|H> Rt, [Rn {, #imm}]

STR Rt, [<Rn | SP>, {,#imm}]

STR<B|H> Rt, [Rn {,#imm}]
```

where:

Rt

is the register to load or store.

Rn

is the register on which the memory address is based.

imm

is an offset from *Rn*. If *imm* is omitted, it is assumed to be zero.

Operation

LDR, LDRB and LDRH instructions load the register specified by *Rt* with either a word, byte or halfword data value from memory. Sizes less than word are zero extended to 32-bits before being written to the register specified by *Rt*.

STR, STRB and STRH instructions store the word, least-significant byte or lower halfword contained in the single register specified by *Rt* in to memory. The memory address to load from or store to is the sum of the value in the register specified by either *Rn* or SP and the immediate value *imm*.

Restrictions

In these instructions:

- *Rt* and *Rn* must only specify R0-R7.
- *imm* must be between:
  - 0 and 1020 and an integer multiple of four for LDR and STR using SP as the base register
  - 0 and 124 and an integer multiple of four for LDR and STR using R0-R7 as the base register
  - 0 and 62 and an integer multiple of two for LDRH and STRH
  - 0 and 31 for LDRB and STRB.
- The computed address must be divisible by the number of bytes in the transaction, see *Address alignment*.

Condition flags

These instructions do not change the flags.

Examples

```
    LDR     R4, [R7                 ; Loads R4 from the address in R7.
    STR     R2, [R0,#const-struc]   ; const-struc is an expression
evaluating
                                ; to a constant in the range 0-1020.
```

3.4.3. LDR and STR, register offset

Load and Store with register offset.

Syntax

`LDR` *Rt*, [*Rn*, *Rm*]

`LDR<B|H>` *Rt*, [*Rn*, *Rm*]

`LDR<SB|SH>` *Rt*, [*Rn*, *Rm*]

`STR` *Rt*, [*Rn*, *Rm*]

`STR<B|H>` *Rt*, [*Rn*, *Rm*]

where:

Rt

is the register to load or store.

Rn

is the register on which the memory address is based.

Rm

is a register containing a value to be used as the offset.

Operation

`LDR`, `LDRB`, `LDRH`, `LDRSB` and `LDRSH` load the register specified by *Rt* with either a word, zero extended byte, zero extended halfword, sign extended byte or sign extended halfword value from memory.

`STR`, `STRB` and `STRH` store the word, least-significant byte or lower halfword contained in the single register specified by *Rt* into memory.

The memory address to load from or store to is the sum of the values in the registers specified by *Rn* and *Rm*.

Restrictions

In these instructions:

- *Rt*, *Rn*, and *Rm* must only specify R0-R7.
- the computed memory address must be divisible by the number of bytes in the load or store, see *Address alignment*.

Condition flags

These instructions do not change the flags.

Examples

```
    STR    R0, [R5, R1]          ; Store value of R0 into an address
equal to
                              ; sum of R5 and R1

    LDRSH  R1, [R2, R3]          ; Load a halfword from the memory
address
                              ; specified by (R2 + R3), sign extend to
32-bits

                              ; and write to R1.
```


3.4.1. ADR

Generates a PC-relative address.

Syntax

`ADR` *Rd*, *label*

where:

Rd

is the destination register.

label

is a PC-relative expression. See *PC-relative expressions*.

Operation

`ADR` generates an address by adding an immediate value to the PC, and writes the result to the destination register.

`ADR` facilitates the generation of position-independent code, because the address is PC-relative.

If you use `ADR` to generate a target address for a `BX` or `BLX` instruction, you must ensure that bit[0] of the address you generate is set to 1 for correct execution.

Restrictions

In this instruction *Rd* must specify R0-R7. The data-value addressed must be word

aligned and within 1020 bytes of the current PC.

Condition flags

This instruction does not change the flags.

Examples

```
    ADR    R1, TextMessage   ; Write address value of a location
labelled as
                             ; TextMessage to R1

    ADR    R3, [PC,#996]     ; Set R3 to value of PC + 996.
```