# Types of ARM Instructions

- As mentioned, LDRB can be used to read from any address.

Example: `mov R0, #0x1000`

`LDRB R1, [R0, #3]` (*).

- Register R1 will get loaded w/ the byte from addr. 0x1003

As an example, we will assume the following memory content:

| Address | Data (in memory) | | | |
|---|---|---|---|---|
| 0FFC | 00 | 00 | 00 | 00 |
| 1000 | 83 | 82 | 81 | 80 |
| 1004 | 00 | 00 | 00 | 00 |

(*) R1 ← | 00 | 00 | 00 | 83 |

extended w/ 0's

Variant: `LDRSB R1, [R0, #3]`

R1 ← | FF | FF | FF | 83 |

extended w/ the sign bit of 83

neg. → 1000 → 83

Note: The address used w/ LDR must be word aligned (multiple of 4). If the addr. does not end in 00 the ARM processor will take an exception (error) and your program will crash, so....

```
mov R0, #0x1000
LDR R1, [R0, #3]
```

will cause an exception error!

Note: you can also read half words (16-bits) using LDRH and LDRSH. Here the addr. has to be a multiple of 2 (any even addr.)

Example:

```
LDRH    R1, [R0]      R1 ← 0000 8180
LDRSH   R1, [R0, #2]  R1 ← FFFF 8382
```

| Address | Data (in memory) | | | |
|---|---|---|---|---|
| 0FFC | 00 | 00 | 00 | 00 |
| 1000 | 83 | 82 | 81 | 80 |
| 1004 | 00 | 00 | 00 | 00 |

---

Similar varients are available for STR:

| | |
|---|---|
| STR | word |
| STRB | Byte |
| STRH | half word |

| Address | Data (in memory) | | | |
|---|---|---|---|---|
| 0FFC | 00 | 00 | 00 | 00 |
| 1000 | 83 | 82 | 81 | 80 |
| 1004 | 00 | 00 | 00 | 00 |

Example. if R1 = FFFF 8382

```
STRB    R1, [R0, #1]
```

| Address | Data (in memory) | | | |
|---|---|---|---|---|
| 0FFC | 00 | 00 | 00 | 00 |
| 1000 | 83 | 82 | 82 | 80 |
| 1004 | 00 | 00 | 00 | 00 |

---

## Arithmetic Instruction.

```
ADD   Rd, Rn, Operand 2     Rd ← Rn + Operand 2

ADD   R8, R0, R1            R8 ← R0 + R1
```

ADD R8, R8, #4 ⊛ R8 ← R8 + 4

ADD R8, #4 R8 ← R8 + 4 (同上)

⊛ If Rd and Rn are the same register, you can omit Rn

ADC R1, R2, R3 R1 ← R2 + R3 + carry-out (C-flag)

SUB R2, R3, R4 R2 ← R3 - R4

SUB R2, R2, R8 R2 ← R2 - R8

SUB R2, R8 R2 ← R2 - R8

SUB R11, #4 R11 ← R11 - 4

Also, there is a multiply instruction (MUL) instruction and multiply accumulate (MLA):

MUL R1, R2, R3 R1 ← R2 × R3

MLA R1, R2, R3, R4 R1 ← R2 × R3 + R4