① Arithmetic Instructions

ADD Rd, Rn, Operand 2 // Rd ← Rn + Operand 2 General Form

e.g: ADD R8, R0, R1 // R8 ← R0 + R1

ADD R8, [R8], #4 // if Rd and Rn are same, can omit Rn

SUB Rd, Rn, Operand 2 // Rd ← Rn - Operand 2

e.g: SUB R2, R3, R4 // R2 ← R3 - R4

- Recall that any of these instructions, without extra "S" at the end do not affect the condition codes Z, N, V, C

But ADDS, SUBS, ANDS do affect condition code.

MUL R1, R2, R3 // $R_1 \leftarrow R_2 * R_3$ multiply

MULA R1, R2, R3, R4 // $R_1 \leftarrow R_2 * R_3 + R_4$ multiply-accumulate

- Integer operations: floating point computation is handled separately
- No divide instruction

② Logic Instructions: AND, ORR, EOR

- Important to deal with individual bits.
- Bitwise logic operations.

e.g: AND R0, R1, R2 // Rd ← R1 · R2 (bitwise)

ORR R3, R4, #0x1

EOR (Exclusive OR)

- can also add "S" to affect condition code

③ Shift / Rotate Instructions

- Used to "move" bits into other bit positions

LSR Rd, Rn, Operand 2 "logic shift right"

// Shift register Rn right by Operand 2 bits → Rd (0 comes into bit 31)

e.g: LSR R2, R3, #1

ASR "Arithmetic Shift Right"

// similar, except the most significant bit (sign bit) stays the same as it was before shift

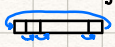
this makes a divide by 2 done by shift right correct for 2's compliement (signed numbers)

LSL logical shift left // multiply by 2

LSL R1, R1, #1 // $R_1 \leftarrow R_1 * 2$

LSL R1, #3 // $R_1 \leftarrow R_1 * 8$

ROR rotate right // shifts right & "wraps around"



ROR R1, R2, #8 // $R_1 \leftarrow R_2$ rotated 8 times

ROR R1, #31 // $R_1 \leftarrow R_1$ rotated left by 1