4. Advanced Arithmetic Operations

As a refresher, here are some fundamental AVR arithmetic and logical instructions.

Mnemonic	Operand	Description	Example	Notes
ADD	Rd, Rr	Add	`ADD R1, R2`	Rd = Rd + Rr
ADC	Rd, Rr	Add with Carry	`ADC R1, R2`	Rd = Rd + Rr + C (Carry flag)
ADIW	Rd, K	Add Immediate to Word	`ADIW R24, 40`	R[d+1]:Rd = R[d+1]:Rd + K
SUB	Rd, Rr	Subtract	`SUB R16, R17`	Rd = Rd - Rr
SBC	Rd, Rr	Subtract with Carry	`SBC R16, R17`	Rd = Rd - Rr - C
SUBI	Rd, K	Subtract Immediate	`SUBI R16, 67`	Rd = Rd - K
SBIW	Rd, K	Substract Immediate from Word	`SBIW R24, 40`	R[d+1]:Rd = R[d+1]:Rd - K
SBCI	Rd, K	Subtract Immediate with Carry	`SBCI R17, 0`	Rd = Rd - K - C
INC	Rd	Increment	`INC R16`	Rd = Rd + 1
DEC	Rd	Decrement	`DEC R16`	Rd = Rd - 1
MUL	Rd, Rr	Multiply Unsigned	`MUL R16, R17`	R1:R0 = Rd × Rr (16-bit result)
MULS	Rd, Rr	Multiply Signed	`MULS R16, R17`	R1:R0 = Rd × Rr (signed)
NEG	Rd	Negate (Two's Complement)	`NEG R16`	Rd = 0x00 - Rd
AND	Rd, Rr	Logical AND	`AND R1, R2`	Rd = Rd AND Rr
ANDI	Rd, K	AND Immediate	`ANDI R16, 0x0F`	Rd = Rd AND K
OR	Rd, Rr	Logical OR	`OR R1, R2`	Rd = Rd OR Rr
ORI	Rd, K	OR Immediate	`ORI R16, 0x80`	Rd = Rd OR K
EOR	Rd, Rr	Exclusive OR	`EOR R16, R17`	Rd = Rd XOR Rr
COM	Rd	One's Complement	`COM R16`	Rd = 0xFF - Rd (inverts all bits)

Mnemonic	Operand	Description	Example	Notes
NEG	Rd	Two's Complement	`NEG R16`	Rd = -Rd (Signed)
CLR	Rd	Clear Register	`CLR R16`	Rd = 0
SER	Rd	Set Register	`SER R16`	Rd = 0xFF (R16-R31 only)

Register Pair Arithmetic

To perform arithmetic operations with 16 bit words (0 - 65535), we can utilize the Carry Flag to perform ripple carry arithmetic operations.

16 Bit Word Addition

Suppose you want to add two big 16 bit numbers 26983 and 4882 stored in two register pairs R16:R17 and R18:R19. You can add the lower byte first then add the higher byte with a carry addition.

```
; DEC 26983 = HEX 0x6967
LDI R16, 0x69   ; upper byte
LDI R17, 0x67   ; lower byte

; DEC 4882 = HEX 0x1312
LDI R18, 0x13   ; upper byte
LDI R19, 0x12   ; lower byte

ADD R17, R19    ; add lower byte
ADC R16, R18    ; add upper byte + carry from lower

; R16:R17 = 31865
```

16 Bit Word Substraction

A similar thing can be done for substracting two 16 bit numbers.

```
; DEC 26983 = HEX 0x6967
LDI R16, 0x69   ; upper byte
LDI R17, 0x67   ; lower byte
```

```
; DEC 4882 = HEX 0x1312
LDI R18, 0x13   ; upper byte
LDI R19, 0x12   ; lower byte

SUB R17, R19    ; substract lower byte
SBC R16, R18    ; substract upper byte - borrow (C) from lower

; R16:R17 = 22101
```

Immediate Arithmetic

There are 3 immediate arithmetic instructions: `ADIW`, `SBIW`, and `SUBI`. In AVR there is no "Add with immediate" instruction. To do so, you can utilize the `SUBI Rd, K` instruction to substract a negative immediate number to achieve addition.

Suppose you want to add 67 to 61 stored in R16. You can do the following

```
LDI     R16, 61
SUBI    R16, -67 ; - -61 = + 61
; R16 = 128
```

Notice how there are only immediate instructions for word addition and substraction? Recall that word immediate addressing (`ADIW` & `SBIW`) can only operate on 6 bit values (0 - 63). How can we perform immediate arithmetic operations on bigger numbers like 26983?

Word Immediate Arithmetic

To perform addition/substraction operation with immediate numbers greater than 63, we simply split the word operation into two byte immediate operations similar to the previously explained 16 bit operations.

16 Bit Immediate Addition

```
; DEC 4882 = HEX 0x1312
LDI R16, 0x13       ; upper byte
LDI R17, 0x12       ; lower byte

.EQU num, 0x5457    ; immediate constant DEC 21591
```

```
SUBI R17, lo8(-num) ; 0x12 - -0x57 = 0x12 + 0x57         ; add lower byte
SBCI R16, hi8(-num) ; 0x13 - -0x54 - C = 0x13 + 0x54 + C ; add upper byte
; R16:R17 = 26473
```

16 Bit Immediate Substraction

```
LDI R16, 0x67       ; yeah you get the idea atp
LDI R17, 0x69

.EQU num, 0x1312
SUBI R17, lo8(num)  ; this time we simply just substract it
SBCI R16, hi8(num)  ; no need to be negative :)
; R16:R17 = 26983
```

Multiplying Numbers

There are multiple multiplication instructions in AVR, each supporting different data formats such as `MUL Rd, Rr` for unsigned numbers, `MULS Rd, Rr` for signed numbers, and `MULSU Rd, Rr` for multiplying a signed with an unsigned number.

Generally, the way they behave are similar: they multiply Rd with Rr then storing a 16 bit result in R1:R0.

```
LDI R16, 23
LDI R17, 9
MUL R16, R17
; R1:R0 = 23 * 9 = 207 (16 bits wide)
```

Revision #2
Created 2026-02-25 09:07:40 UTC by MF
Updated 2026-02-25 09:08:40 UTC by MF