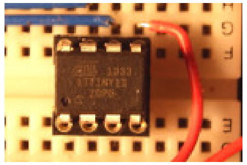

Instruction list in AVR assembler

Rx: result register; Ry: any register; Rh: Register R16 to R31; RdL/H: Register pair low/High (R25:R24, R27:R26, R29:R28, R31:R30); P: Port; PL: Lower port; X: R27:R26; Y: R29:R28; Z: R31:R30; SP: Stack pointer; C: Carry flag; I: Interrupt flag; K: Constant

Instruction list

Mnem.	P1	P2	Description	Action	Flags affected	Clk	Limitations	Words
Arithmetical and logical operations								
ADD	Rx	Ry	Add register	Rx ← Rx + Ry	Z,C,N,V,S,H	1		1
ADC	Rx	Ry	Add register and carry	Rx ← Rx + Ry + C	Z,C,N,V,S,H	1		1
ADIW	RdL	K	Add constant to register pair	RdH:RdL ← RdH:RdL + K	Z,C,N,V,S	2	RdL=24/26/28/30, K: 0 to 63	1
SUB	Rx	Ry	Subtract register	Rx ← Rx - Ry	Z,C,N,V,S,H	1		1
SUBI	Rh	K	Subtract constant	Rh ← Rh - K	Z,C,N,V,S,H	1	R: 16 to 31	1
SBC	Rx	Ry	Subtract register and carry	Rx ← Rx - Ry - C	Z[1],C,N,V,S,H	1		1
SBCI	Rh	K	Subtract constant and carry	Rh ← Rh - K - C	Z[1],C,N,V,S,H	1	R: 16 to 31	1
CP	Ry1	Ry2	Compare register	Rx - Ry	Z,C,N,V,S,H	1		1
CPC	Ry1	Ry2	Compare register and carry	Ry1 - Ry2 - C	Z[1],C,N,V,S,H	1		1
CPI	Rh	K	Compare with constant	Rx - K	Z,C,N,V,S,H	1	R: 16 to 31, K: 0 to 255	1
SBIW	RdL	K	Subtract constant from register pair	RdH:RdL ← RdH:RdL - K	Z,C,N,V,S	2	RdL=24/26/28/30, K: 0 to 63	1
AND	Rx	Ry	Binary AND register	Rx ← Rx AND Ry	Z,N,V,S	1		1
ANDI	Rh	K	Binary AND with constant	Rh ← Rh UND K	Z,N,V,S	1	R: 16 to 31, K: 0 to 255	1
OR	Rx	Ry	Binary OR	Rx ← Rx OR Ry	Z,N,V,S	1		1
ORI	Rh	K	Binary OR with constant	Rh ← Rh OR K	Z,N,V,S	1	R: 16 to 31, K: 0 to 255	1
EOR	Rx	Ry	Exclusive-OR	Rx ← Rx XOR Ry	Z,N,V,S	1		1
COM	Rx		One's complement	Rx ← 255 - Rx	Z,C,N,V,S	1		1
NEG	Rx		Two's complement	Rx ← 256 - Rx	Z,C,N,V,S,H	1		1
SBR	Rh	K	Set bits in constant K	Rh ← Rh OR K	Z,N,V,S	1	R: 16 to 31, K: 0 to 255	1
CBR	Rh	K	Clear bits in constant K	Rh ← Rh AND (NEG K)	Z,N,V,S	1	R: 16 to 31, K: 0 to 255	1
INC	Rx		Increase by one	Rx ← Rx + 1	Z,N,V,S	1		1
DEC	Rx		Decrease by one	Rx ← Rx - 1	Z,N,V,S	1		1
TST	Ry		Compare with zero	Rx OR Rx	Z,N,V,S	1		1
CLR	Rx		Clear all bits	Rx ← 0	Z,N,V,S	1		1
SER	Rh		Set all bits	Rh ← 255	-	1	R: 16 to 31	1
MUL	Ry1	Ry2	Multiply 8 bits	R1:R0 ← Ry1 * Ry2	Z,C	2		1

MULS	Ry1	Ry2	Multiply signed	R1:R0 ← Ry1 * Ry2	Z,C	2	Ry1,Ry2: 16 to 31	1
MULSU	Rx	Ry	Multiplipy unsigned and signed	R1:R0 ← Ry1 * Ry2	Z,C	2	Ry1,Ry2: 16 to 31	1
FMUL	Ry1	Ry2	Floating point multiplication	R1:R0 ← Ry1 * Ry2	Z,C	2	Ry1,Ry2: 16 to 23	1
FMULS	Ry1	Ry2	Floating point multiplication signed	R1:R0 ← Ry1 * Ry2	Z,C	2	Ry1,Ry2: 16 to 23	1
FMULSU	Ry1	Ry2	Floating point multiplication signed and unsigned	R1:R0 ← Ry1 * Ry2	Z,C	2	Ry1,Ry2: 16 to 23	1
DES	K		Data encoding and decoding	(R7:R0, R15:R8)	-	1/2	(MEGA/XMEGA only), K<16	1
The Z-flag is set to one if this instruction yielded zero AND if it was set by the previous instruction. This enables 16-bit comparisons.								
Jump instructions								
RJMP	K		Relative jump	(PC) ← (PC) +/- K	-	2	K: -2048 to 2047	1
IJMP			Indirect jump	(PC) ← Z	-	2		1
EIJMP			Extended indirect jump	(PC) ← EIND + Z	-	2	(XMEGA only)	1
JMP	K		Direct (wide) jump	(PC) ← K	-	3	K: 0 to 65535	2
RCALL	K		Relative call	(Stack) ← (PC), (PC) ← (PC) +/- K	-	2/3/4	K: -2048 to 2047	1
ICALL			Indirect call	(Stack) ← (PC), (PC) ← Z	-	2/3/4		1
EICALL			Extended indirect call	(Stack) ← (PC), (PC) ← EIND+Z	-	3/4		1
CALL	K		Wide call	(Stack) ← (PC), (PC) ← K	-	3/4/5		2
RET			Return from call	(PC) ← (Stack)	-	4		1
RETI			Return from interrupt service routine	(PC) ← (Stack), I ← 1	-	4		1
CPSE	Ry1	Ry2	Jump over next instruction if equal	Ry1=Ry2: (PC) ← (PC + 2)	-	2/3		1
SBRC	Ry	B	Jump over next instruction if bit clear	(Bit)=0: (PC) ← (PC+2)	-	2/3		1
SBRS	Ry	B	Jump over next instruction if bit set	(Bit)=1: (PC) ← (PC+2)	-	2/3		1
SBIC	PL	B	Jump over next instruction if portbit clear	(Bit)=0: (PC) ← (PC+1)	-	2/3		1
SBIS	PL	B	Jump over next instruction if portbit set	(Bit)=1: (PC) ← (PC+1)	-	2/3		1
BRBS	K	B	Jump relative if bit in SREG set	(SREG-Bit=1): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRBC	K	B	Jump relative if bit in SREG clear	(SREG-Bit=0): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BREQ	K		Jump relative if Z in SREG set	(SREG-Z=1): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRNE	K		Jump relative if Z in SREG clear	(SREG-Z=0): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRCS	K		Jump relative if C in SREG set	(SREG-C=1): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRCC	K		Jump relative if C in SREG clear	(SREG-C=0): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRSH	K		Jump relative if C in SREG clear	(SREG-C=0): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRLO	K		Jump relative if C in SREG set	(SREG-C=1): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRMI	K		Jump relative if N in SREG set	(SREG-N=1): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRPL	K		Jump relative if Z in SREG clear	(SREG-N=1): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRGE	K		Jump relative if S in SREG clear	(SREG-S=0): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRLT	K		Jump relative if S in SREG set	(SREG-S=1): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRHS	K		Jump relative if H in SREG set	(SREG-H=1): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRHC	K		Jump relative if H in SREG clear	(SREG-H=0): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRTS	K		Jump relative if T in SREG set	(SREG-T=1): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRTC	K		Jump relative if T in SREG clear	(SREG-T=0): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRVS	K		Jump relative if V in SREG set	(SREG-V=1): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRVC	K		Jump relative if V in SREG clear	(SREG-V=0): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1

BRIE	K		Jump relative if I in SREG set	(SREG-I=1): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
BRID	K		Jump relative if I in SREG clear	(SREG-I=0): (PC) ← (PC) +/- K	-	1/2	K: -63 to + 64	1
Data copy and load instructions								
MOV	Rx	Ry	Copy register	Rx ← Ry	-	1		1
MOVW	Rx	Ry	Copy register pair	Rx+1:Rx ← Ry+1:Ry	-	1	Rx, Ry: Even	1
LDI	Rh	K	Load constant	Rh ← K	-	1	R: 16 to 31, K: 0 to 255	1
LDS	Rh	A	Copy SRAM byte	Rx ← (SRAM-A)	-	2/3/4	R: 16 to 31	2
LD	Rx	X	Copy SRAM byte on address X	Rx ← (X)	-	2/3/4		1
LD	Rx	X+	Copy SRAM byte on address X and increment address	Rx ← (X), X = X + 1	-	2/3		1
LD	Rx	-X	Decrement X and copy SRAM byte from address X	X = X - 1, Rx ← (X)	-	2/3/4		1
LD	Rx	Y	Copy SRAM byte on address Y	Rx ← (Y)	-	2/3/4		1
LD	Rx	Y+	Copy SRAM byte on address Y and increment address	Rx ← (Y), Y = Y + 1	-	2/3		1
LD	Rx	-Y	Decrement Y and copy SRAM byte at address Y	Y = Y - 1, Rx ← (Y)	-	2/3/4		1
LDD	Rx	Y+K	Copy SRAM byte from address (Y+K)	Rx ← (Y+K)	-	2/3	K: 0 to 63	1
LD	Rx	Z	Copy SRAM byte from address Z	Rx ← (Z)	-	2/3/4		1
LD	Rx	Z+	Copy SRAM byte from address Z and increment address	Rx ← (Z), Z = Z + 1	-	2/3		1
LD	Rx	-Z	Decrement Z and copy SRAM byte from address Z	Z = Z - 1, Rx ← (Z)	-	2/3/4		1
LDD	Rx	Z+K	Copy SRAM byte from address (Z+K)	Rx ← (Z+K)	-	2/3	K: 0 to 63	1
STS	A	Rh	Copy to SRAM	(SRAM-A) ← Rx	-	2/3/4	R: 16 to 31	2
ST	X	Rx	Copy to SRAM address in X	(X) ← Rx	-	2/3/4		1
ST	X+	Rx	Copy to SRAM address in X and increment address	(X) ← Rx, X = X + 1	-	2/3		1
ST	-X	Rx	Decrement X and copy to SRAM address X	X = X - 1, (X) ← Rx	-	2/3/4		1
ST	Y	Rx	Copy to SRAM address Y	(Y) ← Rx	-	2/3/4		1
ST	Rx	Y+	Copy to address Y and increment Y	(Y) ← Rx, Y = Y + 1	-	2/3		1
ST	Rx	-Y	Decrement Y and copy to SRAM address Y	Y = Y - 1, (Y) ← Rx	-	2/3/4		1
STD	Y+K	Rx	Copy to SRAM address (Y+K)	(Y+K) ← Rx	-	2/3	K: 0 to 63	1
ST	Z	Rx	Copy to SRAM address Z	(Z) ← Rx	-	2/3/4		1
ST	Z+	Rx	Copy to SRAM address Z and increment Z	(Z) ← Rx, Z = Z + 1	-	2/3		1
ST	-Z	Rx	Decrement Z and copy to SRAM address Z	Z = Z - 1, (Z) ← Rx	-	2/3/4		1
STD	Z+K	Rx	Copy to SRAM address (Z+K)	(Z+K) ← Rx	-	2/3	K: 0 to 63	1
LPM			Copy from program memory address (Z) to R0	R0 ← (Flash Z)	-	3		1
LPM	Rx	Z	Copy from program memory address (Z) to register	Rx ← (Flash Z)	-	3		1
LPM	Rx	Z+	Copy from program memory address (Z) to register and increment Z	Rx ← (Flash Z), Z = Z + 1	-	3		1
ELPM			Copy from extended program memory address (EIND+Z) to register R0	R0 ← (Flash Z)	-	3		1
ELPM	Rx	Z	Copy from extended program memory address (EIND+Z) to register	Rx ← (Flash Z)	-	3		1
ELPM	Rx	Z+	Copy from extended program memory address (EIND+Z) to register and increment Z	Rx ← (Flash Z), Z = Z + 1	-	3		1
SPM			Copy word R1:R0 to program memory address (Z)	(Flash Z) ← R1:R0	-	N		1
SPM	Z+		Copy word R1:R0 to program memory address (Z) and increment Z	(Flash Z) ← R1:R0, Z = Z+ 1	-	N		1
IN	Rx	P	Copy port byte	Rx ← P	-	1	P: 0 to 63	1
OUT	P	Rx	Copy byte to port	P ← Rx	-	1	P: 0 to 63	1
PUSH	Rx		Copy to stack and dekrement (SP)	(Stack) ← Rx, SP = SP - 1	-	2		1

POP	Rx		Copy from stack and increment (SP)	Rx ← (Stack), SP = SP + 1	-	2		1
XCH	Z	Rx	Exchange register with SRAM address (Z)	Rx ↔ (Z)	-	1		1
LAS	Z	Rx	OR register and SRAM (Z) und exchange	Rx ← Rx ODER (Z), (Z) ↔ Rx	-	1		1
LAC	Z	Rx	AND complement register with SRAM (Z) and copy to SRAM (Z)	Rx ← (255-Rx) UND (Z), (Z) ↔ Rx	-	1		1
LAT	Z,	Rd	XOR register and SRAM (Z) and exchange	Rx EXOR (Z), Rx ↔ (Z)	-	1		1
Bit operations								
LSL	Rx		Logical shift left	Rx ← Rx * 2	Z,C,N,V,H	1		1
LSR	Rx		Logical shift right	Rx ← Rx / 2	Z,C,N,V	1		1
ROL	Rx		Binary rotate left with C	Rx ← Rx * 2 with Bit 0 = C/C = Bit 7	Z,C,N,V,H	1		1
ROR	Rx		Binary rotate right with C	Rx ← Rx / 2 with Bit 7 = C/C = Bit 0	Z,C,N,V	1		1
ASR	Rx		Arithmetical shift right	Rx ← Rx(6:0) / 2, Bit 6 = 0	Z,C,N,V	1		1
SWAP	Rx		Exchange upper and lower nibble	Rx ← (7:4) ↔ (3:0)	-	1		1
BSET	B		Set bit in SREG	SREG ← SREG OR (1<<B)	-	1	B: 0 to 7	1
BCLR	B		Clear bit in SREG	SREG ← SREG AND (255-(1<<B))	-	1	B: 0 to 7	1
SBI	PL	B	Set bit in port	PL ← PL OR (1<<B)	-	2	PL: 0 to 31, B: 0 to 7	1
CBI	PL	B	Clear bit in port	PL ← PL AND (255-(1<<B))	-	2	PL: 0 to 31, B: 0 to 7	1
BST	Rx	B	Copy register bit to T	SREG-T ← Rx-Bit B	-	1	B: 0 to 7	1
BLD	Rx	B	Copy T to register bit	Rx-Bit B ← T	-	1	B: 0 to 7	1
SEC			Set SREG C	SREG-Bit C ← 1	-	1		1
CLC			Clear SREG C	SREG-Bit C ← 0	-	1		1
SEN			Set SREG N	SREG-Bit N ← 1	-	1		1
CLN			Clear SREG N	SREG-Bit N ← 0	-	1		1
SEZ			Set SREG Z	SREG-Bit Z ← 1	-	1		1
CLZ			Clear SREG Z	SREG-Bit Z ← 0	-	1		1
SEI			Set SREG I	SREG-Bit I ← 1	-	1		1
CLI			Clear SREG I	SREG-Bit I ← 0	-	1		1
SES			Set SREG S	SREG-Bit S ← 1	-	1		1
CLS			Clear SREG S	SREG-Bit S ← 0	-	1		1
SEV			Set SREG V	SREG-Bit V ← 1	-	1		1
CLV			Clear SREG V	SREG-Bit V ← 0	-	1		1
SET			Set SREG T	SREG-Bit T ← 1	-	1		1
CLT			Clear SREG T	SREG-Bit T ← 0	-	1		1
SEH			Set SREG H	SREG-Bit H ← 1	-	1		1
CLH			Clear SREG H	SREG-Bit H ← 0	-	1		1
Controller instructions								
BREAK			Stop execution, control to debugger		-	1		1
NOP			Do nothing		-	1		1
SLEEP			Sleep		-	1		1
WDR			Clear watchdog	WDR counter ← 0	-	1		1