

ATmega16/32, ATmega406 Instruction Set

The Assembler accepts mnemonic instructions from the instruction set. A summary of the instruction set mnemonics and their parameters is given here. For a detailed description of the Instruction set, refer to the AVR Data Book.

Arithmetic and Logic Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
ADD	Rd,Rr	Add without Carry	$Rd = Rd + Rr$	Z,C,N,V,H,S	1
ADC	Rd,Rr	Add with Carry	$Rd = Rd + Rr + C$	Z,C,N,V,H,S	1
ADIW	Rd, K	Add Immediate To Word	$Rd+1 : Rd, K$	Z,C,N,V,S	2
SUB	Rd,Rr	Subtract without Carry	$Rd = Rd - Rr$	Z,C,N,V,H,S	1
SUBI	Rd,K8	Subtract Immediate	$Rd = Rd - K8$	Z,C,N,V,H,S	1
SBC	Rd,Rr	Subtract with Carry	$Rd = Rd - Rr - C$	Z,C,N,V,H,S	1
SBCI	Rd,K8	Subtract with Carry Immediate	$Rd = Rd - K8 - C$	Z,C,N,V,H,S	1
AND	Rd,Rr	Logical AND	$Rd = Rd \cdot Rr$	Z,N,V,S	1
ANDI	Rd,K8	Logical AND with Immediate	$Rd = Rd \cdot K8$	Z,N,V,S	1
OR	Rd,Rr	Logical OR	$Rd = Rd \vee Rr$	Z,N,V,S	1
ORI	Rd,K8	Logical OR with Immediate	$Rd = Rd \vee K8$	Z,N,V,S	1
EOR	Rd,Rr	Logical Exclusive OR	$Rd = Rd \oplus Rr$	Z,N,V,S	1
COM	Rd	One's Complement	$Rd = \$FF - Rd$	Z,C,N,V,S	1
NEG	Rd	Two's Complement	$Rd = \$00 - Rd$	Z,C,N,V,H,S	1
SBR	Rd,K8	Set Bit(s) in Register	$Rd = Rd \vee K8$	Z,C,N,V,S	1
CBR	Rd,K8	Clear Bit(s) in Register	$Rd = Rd \cdot (\$FF - K8)$	Z,C,N,V,S	1
INC	Rd	Increment Register	$Rd = Rd + 1$	Z,N,V,S	1
DEC	Rd	Decrement Register	$Rd = Rd - 1$	Z,N,V,S	1
TST	Rd	Test for Zero or Negative	$Rd = Rd \cdot Rd$	Z,C,N,V,S	1
CLR	Rd	Clear Register	$Rd = 0$	Z,C,N,V,S	1
SER	Rd	Set Register	$Rd = \$FF$	None	1
SBIW	Rd,K6	Subtract Immediate from Word	$Rdh:Rdl = Rdh:Rdl - K6$	Z,C,N,V,S	2
MUL	Rd,Rr	Multiply Unsigned	$R1:R0 = Rd * Rr$	Z,C	2
MULS	Rd,Rr	Multiply Signed	$R1:R0 = Rd * Rr$	Z,C	2
MULSU	Rd,Rr	Multiply Signed with Unsigned	$R1:R0 = Rd * Rr$	Z,C	2
FMUL	Rd,Rr	Fractional Multiply Unsigned	$R1:R0 = (Rd * Rr) << 1$	Z,C	2
FMULS	Rd,Rr	Fractional Multiply Signed	$R1:R0 = (Rd * Rr) << 1$	Z,C	2
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	$R1:R0 = (Rd * Rr) << 1$	Z,C	2

Branch Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
RJMP	k	Relative Jump	$PC = PC + k + 1$	None	2
IJMP	None	Indirect Jump to (Z)	$PC = Z$	None	2
JMP	k	Jump	$PC = k$	None	3
RCALL	k	Relative Call Subroutine	$STACK = PC+1, PC = PC + k + 1$	None	3/4*

ICALL	None	Indirect Call to (Z)	STACK = PC+1, PC = Z	None	3/4*
CALL	k	Call Subroutine	STACK = PC+2, PC = k	None	4/5*
RET	None	Subroutine Return	PC = STACK	None	4/5*
RETI	None	Interrupt Return	PC = STACK	I	4/5*
CPSE	Rd,Rr	Compare, Skip if equal	if (Rd == Rr) PC = PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd - Rr	Z,C,N,V,H,S	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,H,S	1
CPI	Rd,K8	Compare with Immediate	Rd - K	Z,C,N,V,H,S	1
SBRC	Rr,b	Skip if bit in register cleared	if(Rr(b)==0) PC = PC + 2 or 3	None	1/2/3
SBRSC	Rr,b	Skip if bit in register set	if(Rr(b)==1) PC = PC + 2 or 3	None	1/2/3
SBIC	P,b	Skip if bit in I/O register cleared	if(I/O(P,b)==0) PC = PC + 2 or 3	None	1/2/3
SBIS	P,b	Skip if bit in I/O register set	if(I/O(P,b)==1) PC = PC + 2 or 3	None	1/2/3
BRBC	s,k	Branch if Status flag cleared	if(SREG(s)==0) PC = PC + k + 1	None	1/2
BRBS	s,k	Branch if Status flag set	if(SREG(s)==1) PC = PC + k + 1	None	1/2
BREQ	k	Branch if equal	if(Z==1) PC = PC + k + 1	None	1/2
BRNE	k	Branch if not equal	if(Z==0) PC = PC + k + 1	None	1/2
BRCS	k	Branch if carry set	if(C==1) PC = PC + k + 1	None	1/2
BRCC	k	Branch if carry cleared	if(C==0) PC = PC + k + 1	None	1/2
BRSH	k	Branch if same or higher	if(C==0) PC = PC + k + 1	None	1/2
BRLO	k	Branch if lower	if(C==1) PC = PC + k + 1	None	1/2
BRMI	k	Branch if minus	if(N==1) PC = PC + k + 1	None	1/2
BRPL	k	Branch if plus	if(N==0) PC = PC + k + 1	None	1/2
BRGE	k	Branch if greater than or equal (signed)	if(S==0) PC = PC + k + 1	None	1/2
BRLT	k	Branch if less than (signed)	if(S==1) PC = PC + k + 1	None	1/2
BRHS	k	Branch if half carry flag set	if(H==1) PC = PC + k + 1	None	1/2
BRHC	k	Branch if half carry flag cleared	if(H==0) PC = PC + k + 1	None	1/2
BRIS	k	Branch if T flag set	if(T==1) PC = PC + k + 1	None	1/2
BRIC	k	Branch if T flag cleared	if(T==0) PC = PC + k + 1	None	1/2
BRVS	k	Branch if overflow flag set	if(V==1) PC = PC + k + 1	None	1/2
BRVC	k	Branch if overflow flag cleared	if(V==0) PC = PC + k + 1	None	1/2
BRIE	k	Branch if interrupt enabled	if(I==1) PC = PC + k + 1	None	1/2
BRID	k	Branch if interrupt disabled	if(I==0) PC = PC + k + 1	None	1/2

Data Transfer Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
MOV	Rd,Rr	Copy register	Rd = Rr	None	1
MOVW	Rd,Rr	Copy register pair	Rd+1:Rd = Rr+1:Rr, r,d even	None	1
LDI	Rd,K8	Load Immediate	Rd = K	None	1
LDS	Rd,k	Load Direct	Rd = (k)	None	2*
LD	Rd,X	Load Indirect	Rd = (X)	None	2*
LD	Rd,X+	Load Indirect and Post-Increment	Rd = (X), X=X+1	None	2*
LD	Rd,-X	Load Indirect and Pre-Decrement	X=X-1, Rd = (X)	None	2*
LD	Rd,Y	Load Indirect	Rd = (Y)	None	2*

LD	Rd,Y+	Load Indirect and Post-Increment	$Rd = (Y), Y=Y+1$	None	2*
LD	Rd,-Y	Load Indirect and Pre-Decrement	$Y=Y-1, Rd = (Y)$	None	2*
LDD	Rd,Y+q	Load Indirect with displacement	$Rd = (Y+q)$	None	2*
LD	Rd,Z	Load Indirect	$Rd = (Z)$	None	2*
LD	Rd,Z+	Load Indirect and Post-Increment	$Rd = (Z), Z=Z+1$	None	2*
LD	Rd,-Z	Load Indirect and Pre-Decrement	$Z=Z-1, Rd = (Z)$	None	2*
LDD	Rd,Z+q	Load Indirect with displacement	$Rd = (Z+q)$	None	2*
STS	k,Rr	Store Direct	$(k) = Rr$	None	2*
ST	X,Rr	Store Indirect	$(X) = Rr$	None	2*
ST	X+,Rr	Store Indirect and Post-Increment	$(X) = Rr, X=X+1$	None	2*
ST	-X,Rr	Store Indirect and Pre-Decrement	$X=X-1, (X)=Rr$	None	2*
ST	Y,Rr	Store Indirect	$(Y) = Rr$	None	2*
ST	Y+,Rr	Store Indirect and Post-Increment	$(Y) = Rr, Y=Y+1$	None	2
ST	-Y,Rr	Store Indirect and Pre-Decrement	$Y=Y-1, (Y) = Rr$	None	2
ST	Y+q,Rr	Store Indirect with displacement	$(Y+q) = Rr$	None	2
ST	Z,Rr	Store Indirect	$(Z) = Rr$	None	2
ST	Z+,Rr	Store Indirect and Post-Increment	$(Z) = Rr, Z=Z+1$	None	2
ST	-Z,Rr	Store Indirect and Pre-Decrement	$Z=Z-1, (Z) = Rr$	None	2
ST	Z+q,Rr	Store Indirect with displacement	$(Z+q) = Rr$	None	2
LPM	None	Load Program Memory	$R0 = (Z)$	None	3
LPM	Rd,Z	Load Program Memory	$Rd = (Z)$	None	3
LPM	Rd,Z+	Load Program Memory and Post-Increment	$Rd = (Z), Z=Z+1$	None	3
SPM	None	Store Program Memory	$(Z) = R1:R0$	None	-
IN	Rd,P	In Port	$Rd = P$	None	1
OUT	P,Rr	Out Port	$P = Rr$	None	1
PUSH	Rr	Push register on Stack	$STACK = Rr$	None	2
POP	Rd	Pop register from Stack	$Rd = STACK$	None	2

Bit and Bit-test Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
LSL	Rd	Logical shift left	$Rd(n+1)=Rd(n), Rd(0)=0, C=Rd(7)$	Z,C,N,V,H,S	1
LSR	Rd	Logical shift right	$Rd(n)=Rd(n+1), Rd(7)=0, C=Rd(0)$	Z,C,N,V,S	1
ROL	Rd	Rotate left through carry	$Rd(0)=C, Rd(n+1)=Rd(n), C=Rd(7)$	Z,C,N,V,H,S	1
ROR	Rd	Rotate right through carry	$Rd(7)=C, Rd(n)=Rd(n+1), C=Rd(0)$	Z,C,N,V,S	1
ASR	Rd	Arithmetic shift right	$Rd(n)=Rd(n+1), n=0,...,6$	Z,C,N,V,S	1
SWAP	Rd	Swap nibbles	$Rd(3..0) = Rd(7..4), Rd(7..4) = Rd(3..0)$	None	1
BSET	s	Set flag	$SREG(s) = 1$	SREG(s)	1
BCLR	s	Clear flag	$SREG(s) = 0$	SREG(s)	1
SBI	P,b	Set bit in I/O register	$I/O(P,b) = 1$	None	2
CBI	P,b	Clear bit in I/O register	$I/O(P,b) = 0$	None	2
BST	Rr,b	Bit store from register to T	$T = Rr(b)$	T	1
BLD	Rd,b	Bit load from register to T	$Rd(b) = T$	None	1
SEC	None	Set carry flag	$C = 1$	C	1

CLC	None	Clear carry flag	C = 0	C	1
SEN	None	Set negative flag	N = 1	N	1
CLN	None	Clear negative flag	N = 0	N	1
SEZ	None	Set zero flag	Z = 1	Z	1
CLZ	None	Clear zero flag	Z = 0	Z	1
SEI	None	Set interrupt flag	I = 1	I	1
CLI	None	Clear interrupt flag	I = 0	I	1
SES	None	Set signed flag	S = 1	S	1
CLN	None	Clear signed flag	S = 0	S	1
SEV	None	Set overflow flag	V = 1	V	1
CLV	None	Clear overflow flag	V = 0	V	1
SET	None	Set T-flag	T = 1	T	1
CLT	None	Clear T-flag	T = 0	T	1
SEH	None	Set half carry flag	H = 1	H	1
CLH	None	Clear half carry flag	H = 0	H	1
NOP	None	No operation	None	None	1
SLEEP	None	Sleep	See instruction manual	None	1
WDR	None	Watchdog Reset	See instruction manual	None	1

The Assembler is not case sensitive.

The operands have the following forms:

Rd: Destination (and source) register in the register file

Rr: Source register in the register file

b: Constant (0-7), can be a constant expression

s: Constant (0-7), can be a constant expression

P: Constant (0-31/63), can be a constant expression

K6: Constant (0-63), can be a constant expression

K8: Constant (0-255), can be a constant expression

k: Constant, value range depending on instruction. Can be a constant expression

q: Constant (0-63), can be a constant expression

Rdl: R24, R26, R28, R30. For ADIW and SBIW instructions

X,Y,Z: Indirect address registers (X=R27:R26, Y=R29:R28, Z=R31:R30)