Assembler directives

Directive	Description
BYTE	Reserve byte(s) to a variable.
CSEG	Code Segment
CSEGSIZE	Program memory size
DB	Define constant byte(s)
DEF	Define a symbolic name on a register
DSEG	Data Segment
DW	Define Constant word(s)
ENDM, ENDMACRO	EndMacro
EQU	Set a symbol equal to an expression
ESEG	EEPROM Segment
EXIT	Exit from file
INCLUDE	Read source from another file
LIST	Turn listfile generation on
LISTMAC	Turn Macro expansion in list file on
MACRO	Begin Macro
NOLIST	Turn listfile generation off
ORG	Set program origin
SET	Set a symbol to an expression
ELSE,ELIF	Conditional assembly
ENDIF	Conditional assembly
ERROR	Outputs an error message
IF,IFDEF,IFNDEF	Conditional assembly
MESSAGE	Outputs a message string

Directive	Description
DD	Define Doubleword
DQ	Define Quadword
UNDEF	Undefine register symbol
WARNING	Outputs a warning message
OVERLAP/NOOVERLAP	Set up overlapping section

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 Immedtiate	Rd = Rd - K8 - C	Z,C,N,V,H,S	1
AND	Rd,Rr	Logical AND	Rd = Rd · Rr	Z,N,V,S	1
ANDI	Rd,K8	Logical AND with Immediate	Rd = Rd · K8	Z,N,V,S	1
OR	Rd,Rr	Logical OR	Rd = Rd V Rr	Z,N,V,S	1
ORI	Rd,K8	Logical OR with Immediate	Rd = Rd V K8	Z,N,V,S	1
EOR	Rd,Rr	Logical Exclusive OR	Rd = Rd EOR 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 = $00 - Rd	Z,C,N,V,H,S	1
CBR	Rd,K8	Set Bit(s) in Register	Rd = Rd V K8	Z,C,N,V,S	1
CBR	Rd,K8	Clear Bit(s) in Register	Rd = Rd · ($FF - K8)	Z,C,N,V,S	1
INC	Rd	Increment Register	Rd = Rd + 1	Z,N,V,S	1

Mnemonic	Operands	Description	Operation	Flags	Cycles
DEC	Rd	Decrement Register	Rd = Rd -1	Z,N,V,S	1
TST	Rd	Test for Zero or Negative	Rd = Rd · Rd	Z,C,N,V,S	1
CLR	Rd	Clear Register	Rd = 0	Z,N,V,S	1
SER	Rd	Set Register	Rd = $FF	None	1
SBIW	Rdl,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

Test/Branch Instructions

Mnemonic	Operands	Description	Operation	Flags	Cycles
RJMP	k	Relative Jump	PC = PC + k +1	None	2
IJMP	None	Indirect Jump to (X,Y,Z)	PC = Z	None	2
EIJMP	None	Extended Indirect Jump (X,Y,Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) = EIND	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 (X,Y,Z)	STACK = PC+1, PC = Z	None	3/4*
EICALL	None	Extended Indirect Call to (X,Y,Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) =EIND	None	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*

Mnemonic	Operands	Description	Operation	Flags	Cycles
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
SBRS	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
BRTS	k	Branch if T flag set	if(T==1) PC = PC + k + 1	None	1/2
BRTC	k	Branch if T flag cleared	if(T==0) PC = PC + k + 1	None	1/2

Mnemonic	Operands	Description	Operation	Flags	Cycles
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,Y,Z	Load Indirect	Rd = (X)	None	2*
LD	Rd,X,Y,Z	Load Indirect and Post-Increment	Rd = (X), X=X+1	None	2*
LD	Rd,X,Y,Z	Load Indirect and Pre-Decrement	X=X-1, Rd = (X)	None	2*
LD	Rd,X,Y,Z	Load Indirect	Rd = (Y)	None	2*
LD	Rd,X,Y,Z	Load Indirect and Post-Increment	Rd = (Y), Y=Y+1	None	2*
LD	Rd,X,Y,Z	Load Indirect and Pre-Decrement	Y=Y-1, Rd = (Y)	None	2*
LD	Rd,X,Y,Z+q	Load Indirect with displacement	Rd = (Y+q)	None	2*
LD	Rd,X,Y,Z	Load Indirect	Rd = (Z)	None	2*
LD	Rd,X,Y,Z	Load Indirect and Post-Increment	Rd = (Z), Z=Z+1	None	2*
LD	Rd,X,Y,Z	Load Indirect and Pre-Decrement	Z=Z-1, Rd = (Z)	None	2*
LAC	Rd,X,Y,Z	Load and Clear	Z = Rd •($FF-Z)	None	2
LAT	Rd,X,Y,Z	Load and Toggle	Z = Rd ⊕ (Z)	None	2
LAS	Rd,X,Y,Z	Load and Set	Z = Rd v (Z)	None	2
XCH	Rd,X,Y,Z	Exchange	Z = Rd, Rd = Z	None	2

Mnemonic	Operands	Description	Operation	Flags	Cycles
LD	Rd,X,Y,Z+q	Load Indirect with displacement	Rd = (Z+q)	None	2*
STS	>k, Rr	Store Direct	(k) = Rr	None	2*
ST	X,Y,Z,Rr	Store Indirect	(X) = Rr	None	2*
ST	X,Y,Z,Rr	Store Indirect and Post-Increment	(X) = Rr, X=X+1	None	2*
ST	X,Y,Z,Rr	Store Indirect and Pre-Decrement	X=X-1, (X)=Rr	None	2*
ST	X,Y,Z,Rr	Store Indirect	(Y) = Rr	None	2*
ST	X,Y,Z,Rr	Store Indirect and Post-Increment	(Y) = Rr, Y=Y+1	None	2
ST	X,Y,Z,Rr	Store Indirect and Pre-Decrement	Y=Y-1, (Y) = Rr	None	2
ST	X,Y,Z+q,Rr	Store Indirect with displacement	(Y+q) = Rr	None	2
ST	X,Y,Z,Rr	Store Indirect	(Z) = Rr	None	2
ST	X,Y,Z,Rr	Store Indirect and Post-Increment	(Z) = Rr, Z=Z+1	None	2
ST	X,Y,Z,Rr	Store Indirect and Pre-Decrement	Z=Z-1, (Z) = Rr	None	2
ST	X,Y,Z+q,Rr	Store Indirect with displacement	(Z+q) = Rr	None	2
LPM	None	Load Program Memory	R0 = (X,Y,Z)	None	3
LPM	Rd,X,Y,Z	Load Program Memory	Rd = (X,Y,Z)	None	3
LPM	Rd,X,Y,Z	Load Program Memory and Post-Increment	Rd = (X,Y,Z), Z=Z+1	None	3
ELPM	None	Extended Load Program Memory	R0 = (RAMPZ:X,Y,Z)	None	3
ELPM	Rd,X,Y,Z	Extended Load Program Memory	Rd = (RAMPZ:X,Y,Z)	None	3
ELPM	Rd,X,Y,Z	Extended Load Program Memory and Post Increment	Rd = (RAMPZ:X,Y,Z), Z = Z+1	None	3
SPM	None	Store Program Memory	(X,Y,Z) = R1:R0	None	-
ESPM	None	Extended Store Program Memory	(RAMPZ:X,Y,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

Mnemonic	Operands	Description	Operation	Flags	Cycles
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
OR	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

Mnemonic	Operands	Description	Operation	Flags	Cycles
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
BREAK	None	Execution Break	See instruction manual	None	1

Notes:

The Assembler is not case sensitive.

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)

SREG : Status register

C : Carry flag in status register
Z : Zero flag in status register
N : Negative flag in status register
V : Two's complement overflow indicator
S : N [+] V, For signed tests
H : Half Carry flag in the status register
T : Transfer bit used by BLD and BST instructions
I : Global interrupt enable/disable flag