

APPENDIX : RESET & INTERRUPT VECTOR TABLE

Vector No.	Program Address⁽²⁾	Source	Interrupt Definition
1	\$000 ⁽¹⁾	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	\$002	INT0	External Interrupt Request 0
3	\$004	INT1	External Interrupt Request 1
4	\$006	INT2	External Interrupt Request 2
5	\$008	TIMER2 COMP	Timer/Counter2 Compare Match
6	\$00A	TIMER2 OVF	Timer/Counter2 Overflow
7	\$00C	TIMER1 CAPT	Timer/Counter1 Capture Event
8	\$00E	TIMER1 COMPA	Timer/Counter1 Compare Match A
9	\$010	TIMER1 COMPB	Timer/Counter1 Compare Match B
10	\$012	TIMER1 OVF	Timer/Counter1 Overflow
11	\$014	TIMER0 COMP	Timer/Counter0 Compare Match
12	\$016	TIMER0 OVF	Timer/Counter0 Overflow
13	\$018	SPI, STC	Serial Transfer Complete
14	\$01A	USART, RXC	USART, Rx Complete
15	\$01C	USART, UDRE	USART Data Register Empty
16	\$01E	USART, TXC	USART, Tx Complete
17	\$020	ADC	ADC Conversion Complete
18	\$022	EE_RDY	EEPROM Ready
19	\$024	ANA_COMP	Analog Comparator
20	\$026	TWI	Two-wire Serial Interface
21	\$028	SPM_RDY	Store Program Memory Ready

AVR Assembler Directive

Segment	Directive	Description
Header	.DEVICE	Defines the type of the target processor and the applicable set of instructions (illegal instructions for that type trigger an error message, syntax: .DEVICE AT90S8515)
	.DEF	Defines a synonym for a register (e.g. .DEF MyReg = R16)
	.EQU	Defines a symbol and sets its value (later changes of this value remain possible, syntax: .EQU test = 1234567, internal storage of the value is 4-byte- Integer)
	.SET	Fixes the value of a symbol (later redefinition is not possible)
	.INCLUDE	Includes a file and assembles its content, just like its content would be part of the calling file (typical e.g. including the header file: .INCLUDE "C:\avrtools\appnotes\8515def.inc")
Code	.CSEG	Start of the code segment (all that follows is assembled to the code segment and will go to the program space)
	.DB	Inserts one or more constant bytes in the code segment (could be numbers from 0..255, an ASCII-character like 'c', a string like 'abcde' or a combination like 1,2,3,'abc'. The number of inserted bytes must be even, otherwise an additional zero byte will be inserted by the assembler.)
	.DW	Insert a binary word in the code segment (e.g. produces a table within the code)
	.LISTMAC	Macros will be listed in the .LST-file. (Default is that macros are not listed)
	.MACRO	Beginning of a macro (no code will be produced, call of the macro later produces code, syntax: .MACRO macroname parameters, calling by: macroname parameters)
	.ENDMACRO	End of the macro
EEPROM	.ESEG	Assemble to the EEPROM-segment (the code produced will go to the EEPROM section, the code produces an .EEP-file)
	.DB	Inserts one or more constant bytes in the EEPROM segment (could be numbers from 0..255, an ASCII-character like 'c', a string like 'abcde' or a combination like 1,2,3,'abc'.)
	.DW	Inserts a binary word to the EEPROM segment (the lower byte goes to the next address, the higher byte follows on the incremented address)
SRAM	.DSEG	Assemble to the data segment (here only BYTE directives and labels are valid, during assembly only the labels are used)
	.BYTE	Reserves one or more bytes space in the data segment (only used to produce correct labels, does not insert any values!)
Everywhere	.ORG	Defines the address within the respective segment, where the assembler assembles to (e.g. .ORG 0x0000)
	.LIST	Switches the listing to the .LST-file on (the assembled code will be listed in a readable text file .LST)
	.NOLIST	Switches the output to the .LST-file off, suppresses listing.
	.INCLUDE	Inserts the content of another source code file, as if its content would be part of the source file (typical e.g. including the header file: .INCLUDE "C:\avrtools\appnotes\8515def.inc")
	.EXIT	End of the assembler-source code (stops the assembling process)

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
\$3F (\$5F)	SREG	I	T	H	S	V	N	Z	C	10
\$3E (\$5E)	SPH	–	–	–	–	SP11	SP10	SP9	SP8	12
\$3D (\$5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
\$3C (\$5C)	OCR0	Timer/Counter0 Output Compare Register								82
\$3B (\$5B)	GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	47, 67
\$3A (\$5A)	GIFR	INTF1	INTF0	INTF2	–	–	–	–	–	68
\$39 (\$59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	82, 112, 130
\$38 (\$58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	83, 112, 130
\$37 (\$57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	248
\$36 (\$56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	177
\$35 (\$55)	MCUCR	SE	SM2	SM1	SMD	ISC11	ISC10	ISC01	ISC00	32, 66
\$34 (\$54)	MCUCSR	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	40, 67, 228
\$33 (\$53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	80
\$32 (\$52)	TCNT0	Timer/Counter0 (8 Bits)								82
\$31 ⁽¹⁾ (\$51) ⁽¹⁾	OSCCAL	Oscillator Calibration Register								30
	OCDR	On-Chip Debug Register								224
\$30 (\$50)	SFIOR	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	56, 85, 131, 198, 218
\$2F (\$4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	107
\$2E (\$4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	110
\$2D (\$4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								111
\$2C (\$4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								111
\$2B (\$4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								111
\$2A (\$4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								111
\$29 (\$49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								111
\$28 (\$48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								111
\$27 (\$47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								111
\$26 (\$46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								111
\$25 (\$45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	125
\$24 (\$44)	TCNT2	Timer/Counter2 (8 Bits)								127
\$23 (\$43)	OCR2	Timer/Counter2 Output Compare Register								127
\$22 (\$42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	128
\$21 (\$41)	WDTCSR	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	42
\$20 ⁽²⁾ (\$40) ⁽²⁾	UBRRH	URSEL	–	–	–	UBRR[11:8]				164
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	162
\$1F (\$3F)	EEARH	–	–	–	–	–	–	EEAR9	EEAR8	19
\$1E (\$3E)	EEARL	EEPROM Address Register Low Byte								19
\$1D (\$3D)	EEDR	EEPROM Data Register								19
\$1C (\$3C)	EEDCR	–	–	–	–	EERIE	EEMWE	EWE	EERE	19
\$1B (\$3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	64
\$1A (\$3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	64
\$19 (\$39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	64
\$18 (\$38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	64
\$17 (\$37)	DDRB	DOB7	DOB6	DOB5	DOB4	DOB3	DOB2	DOB1	DOB0	64
\$16 (\$36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	65
\$15 (\$35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	65
\$14 (\$34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	65
\$13 (\$33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	65
\$12 (\$32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	65
\$11 (\$31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	65
\$10 (\$30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	65
\$0F (\$2F)	SPDR	SPI Data Register								138
\$0E (\$2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	138
\$0D (\$2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	136
\$0C (\$2C)	UDR	USART I/O Data Register								159
\$0B (\$2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	160
\$0A (\$2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	161
\$09 (\$29)	UBRRL	USART Baud Rate Register Low Byte								164
\$08 (\$28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	199
\$07 (\$27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	214
\$06 (\$26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	216
\$05 (\$25)	ADCH	ADC Data Register High Byte								217
\$04 (\$24)	ADCL	ADC Data Register Low Byte								217
\$03 (\$23)	TWDR	Two-wire Serial Interface Data Register								179
\$02 (\$22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	179
\$01 (\$21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	178
\$00 (\$20)	TWBR	Two-wire Serial Interface Bit Rate Register								177

Complete Instruction Set Summary

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
Arithmetic and Logic Instructions					
ADD	Rd, Rr	Add without Carry	$Rd \leftarrow Rd + Rr$	Z,C,N,V,S,H	1
ADC	Rd, Rr	Add with Carry	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,S,H	1
ADIW	Rd, K	Add Immediate to Word	$Rd+1:Rd \leftarrow Rd+1:Rd + K$	Z,C,N,V,S	2 ⁽¹⁾
SUB	Rd, Rr	Subtract without Carry	$Rd \leftarrow Rd - Rr$	Z,C,N,V,S,H	1
SUBI	Rd, K	Subtract Immediate	$Rd \leftarrow Rd - K$	Z,C,N,V,S,H	1
SBC	Rd, Rr	Subtract with Carry	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,S,H	1
SBCI	Rd, K	Subtract Immediate with Carry	$Rd \leftarrow Rd - K - C$	Z,C,N,V,S,H	1
SBIW	Rd, K	Subtract Immediate from Word	$Rd+1:Rd \leftarrow Rd+1:Rd - K$	Z,C,N,V,S	2 ⁽¹⁾
AND	Rd, Rr	Logical AND	$Rd \leftarrow Rd \bullet Rr$	Z,N,V,S	1
ANDI	Rd, K	Logical AND with Immediate	$Rd \leftarrow Rd \bullet K$	Z,N,V,S	1
OR	Rd, Rr	Logical OR	$Rd \leftarrow Rd \vee Rr$	Z,N,V,S	1
ORI	Rd, K	Logical OR with Immediate	$Rd \leftarrow Rd \vee K$	Z,N,V,S	1
EOR	Rd, Rr	Exclusive OR	$Rd \leftarrow Rd \oplus Rr$	Z,N,V,S	1
COM	Rd	One's Complement	$Rd \leftarrow \$FF - Rd$	Z,C,N,V,S	1
NEG	Rd	Two's Complement	$Rd \leftarrow \$00 - Rd$	Z,C,N,V,S,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V,S	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \bullet (\$FFh - K)$	Z,N,V,S	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V,S	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V,S	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z,N,V,S	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V,S	1
SER	Rd	Set Register	$Rd \leftarrow \$FF$	None	1
MUL	Rd,Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$ (UU)	Z,C	2 ⁽¹⁾
MULS	Rd,Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$ (SS)	Z,C	2 ⁽¹⁾
MULSU	Rd,Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr$ (SU)	Z,C	2 ⁽¹⁾
FMUL	Rd,Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$ (UU)	Z,C	2 ⁽¹⁾
FMULS	Rd,Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$ (SS)	Z,C	2 ⁽¹⁾
FMULSU	Rd,Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$ (SU)	Z,C	2 ⁽¹⁾
Branch Instructions					
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
IJMP		Indirect Jump to (Z)	$PC(15:0) \leftarrow Z, PC(21:16) \leftarrow 0$	None	2 ⁽¹⁾
EIJMP		Extended Indirect Jump to (Z)	$PC(15:0) \leftarrow Z, PC(21:16) \leftarrow EIND$	None	2 ⁽¹⁾
JMP	k	Jump	$PC \leftarrow k$	None	3 ⁽¹⁾
RCALL	k	Relative Call Subroutine	$PC \leftarrow PC + k + 1$	None	3 / 4 ⁽⁴⁾
ICALL		Indirect Call to (Z)	$PC(15:0) \leftarrow Z, PC(21:16) \leftarrow 0$	None	3 / 4 ⁽¹⁾⁽⁴⁾
EICALL		Extended Indirect Call to (Z)	$PC(15:0) \leftarrow Z, PC(21:16) \leftarrow EIND$	None	4 ⁽¹⁾⁽⁴⁾
CALL	k	Call Subroutine	$PC \leftarrow k$	None	4 / 5 ⁽¹⁾⁽⁴⁾
RET		Subroutine Return	$PC \leftarrow STACK$	None	4 / 5 ⁽⁴⁾
RETI		Interrupt Return	$PC \leftarrow STACK$	I	4 / 5 ⁽⁴⁾

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
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,S,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,S,H	1
CPI	Rd,K	Compare with Immediate	Rd - K	Z,C,N,V,S,H	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	A, b	Skip if Bit in I/O Register Cleared	if(I/O(A,b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIS	A, b	Skip if Bit in I/O Register Set	if(I/O(A,b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC+k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC+k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V = 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V = 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
Data Transfer Instructions					
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1 (1)
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LDS	Rd, k	Load Direct from data space	Rd ← (k)	None	2 (1)(4)
LD	Rd, X	Load Indirect	Rd ← (X)	None	2 (2)(4)
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X), X ← X + 1	None	2 (2)(4)
LD	Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1, Rd ← (X)	None	2 (2)(4)
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2 (2)(4)
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y), Y ← Y + 1	None	2 (2)(4)
LD	Rd, -Y	Load Indirect and Pre-Decrement	Y ← Y - 1, Rd ← (Y)	None	2 (2)(4)
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2 (1)(4)
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2 (2)(4)
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2 (2)(4)
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2 (2)(4)
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2 (1)(4)
STS	k, Rr	Store Direct to data space	(k) ← Rr	None	2 (1)(4)
ST	X, Rr	Store Indirect	(X) ← Rr	None	2 (2)(4)

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
ST	X+, Rr	Store Indirect and Post-Increment	$(X) \leftarrow Rr, X \leftarrow X + 1$	None	2 ⁽²⁾⁽⁴⁾
ST	-X, Rr	Store Indirect and Pre-Decrement	$X \leftarrow X - 1, (X) \leftarrow Rr$	None	2 ⁽²⁾⁽⁴⁾
ST	Y, Rr	Store Indirect	$(Y) \leftarrow Rr$	None	2 ⁽²⁾⁽⁴⁾
ST	Y+, Rr	Store Indirect and Post-Increment	$(Y) \leftarrow Rr, Y \leftarrow Y + 1$	None	2 ⁽²⁾⁽⁴⁾
ST	-Y, Rr	Store Indirect and Pre-Decrement	$Y \leftarrow Y - 1, (Y) \leftarrow Rr$	None	2 ⁽²⁾⁽⁴⁾
STD	Y+q,Rr	Store Indirect with Displacement	$(Y + q) \leftarrow Rr$	None	2 ⁽¹⁾⁽⁴⁾
ST	Z, Rr	Store Indirect	$(Z) \leftarrow Rr$	None	2 ⁽²⁾⁽⁴⁾
ST	Z+, Rr	Store Indirect and Post-Increment	$(Z) \leftarrow Rr, Z \leftarrow Z + 1$	None	2 ⁽²⁾⁽⁴⁾
ST	-Z, Rr	Store Indirect and Pre-Decrement	$Z \leftarrow Z - 1, (Z) \leftarrow Rr$	None	2 ⁽²⁾⁽⁴⁾
STD	Z+q,Rr	Store Indirect with Displacement	$(Z + q) \leftarrow Rr$	None	2 ⁽¹⁾⁽⁴⁾
LPM		Load Program Memory	$R0 \leftarrow (Z)$	None	3 ⁽³⁾
LPM	Rd, Z	Load Program Memory	$Rd \leftarrow (Z)$	None	3 ⁽³⁾
LPM	Rd, Z+	Load Program Memory and Post-Increment	$Rd \leftarrow (Z), Z \leftarrow Z + 1$	None	3 ⁽³⁾
ELPM		Extended Load Program Memory	$R0 \leftarrow (RAMPZ:Z)$	None	3 ⁽¹⁾
ELPM	Rd, Z	Extended Load Program Memory	$Rd \leftarrow (RAMPZ:Z)$	None	3 ⁽¹⁾
ELPM	Rd, Z+	Extended Load Program Memory and Post-Increment	$Rd \leftarrow (RAMPZ:Z), Z \leftarrow Z + 1$	None	3 ⁽¹⁾
SPM		Store Program Memory	$(Z) \leftarrow R1:R0$	None	- ⁽¹⁾
IN	Rd, A	In From I/O Location	$Rd \leftarrow I/O(A)$	None	1
OUT	A, Rr	Out To I/O Location	$I/O(A) \leftarrow Rr$	None	1
PUSH	Rr	Push Register on Stack	$STACK \leftarrow Rr$	None	2 ⁽¹⁾
POP	Rd	Pop Register from Stack	$Rd \leftarrow STACK$	None	2 ⁽¹⁾
Bit and Bit-test Instructions					
LSL	Rd	Logical Shift Left	$Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0, C \leftarrow Rd(7)$	Z,C,N,V,H	1
LSR	Rd	Logical Shift Right	$Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0, C \leftarrow Rd(0)$	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	$Rd(0) \leftarrow C, Rd(n+1) \leftarrow Rd(n), C \leftarrow Rd(7)$	Z,C,N,V,H	1
ROR	Rd	Rotate Right Through Carry	$Rd(7) \leftarrow C, Rd(n) \leftarrow Rd(n+1), C \leftarrow Rd(0)$	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	$Rd(n) \leftarrow Rd(n+1), n=0..6$	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	$Rd(3..0) \leftrightarrow Rd(7..4)$	None	1
BSET	s	Flag Set	$SREG(s) \leftarrow 1$	SREG(s)	1
BCLR	s	Flag Clear	$SREG(s) \leftarrow 0$	SREG(s)	1
SBI	A, b	Set Bit in I/O Register	$I/O(A, b) \leftarrow 1$	None	2
CBI	A, b	Clear Bit in I/O Register	$I/O(A, b) \leftarrow 0$	None	2
BST	Rr, b	Bit Store from Register to T	$T \leftarrow Rr(b)$	T	1
BLD	Rd, b	Bit load from T to Register	$Rd(b) \leftarrow T$	None	1
SEC		Set Carry	$C \leftarrow 1$	C	1
CLC		Clear Carry	$C \leftarrow 0$	C	1
SEN		Set Negative Flag	$N \leftarrow 1$	N	1
CLN		Clear Negative Flag	$N \leftarrow 0$	N	1
SEZ		Set Zero Flag	$Z \leftarrow 1$	Z	1
CLZ		Clear Zero Flag	$Z \leftarrow 0$	Z	1
SEI		Global Interrupt Enable	$I \leftarrow 1$	I	1
CLI		Global Interrupt Disable	$I \leftarrow 0$	I	1
SES		Set Signed Test Flag	$S \leftarrow 1$	S	1
CLS		Clear Signed Test Flag	$S \leftarrow 0$	S	1

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
SEV		Set Two's Complement Overflow	$V \leftarrow 1$	V	1
CLV		Clear Two's Complement Overflow	$V \leftarrow 0$	V	1
SET		Set T in SREG	$T \leftarrow 1$	T	1
CLT		Clear T in SREG	$T \leftarrow 0$	T	1
SEH		Set Half Carry Flag in SREG	$H \leftarrow 1$	H	1
CLH		Clear Half Carry Flag in SREG	$H \leftarrow 0$	H	1
MCU Control Instructions					
BREAK		Break	(See specific descr. for BREAK)	None	1 ⁽¹⁾
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR)	None	1

- Notes:
1. This instruction is not available in all devices. Refer to the device specific instruction set summary.
 2. Not all variants of this instruction are available in all devices. Refer to the device specific instruction set summary.
 3. Not all variants of the LPM instruction are available in all devices. Refer to the device specific instruction set summary. The LPM instruction is not implemented at all in the AT90S1200 device.
 4. Cycle times for Data memory accesses assume internal memory accesses, and are not valid for accesses via the external RAM interface. For LD, ST, LDS, STS, PUSH, POP, add one cycle plus one cycle for each wait state. For CALL, ICALL, EICALL, RCALL, RET, RETI in devices with 16-bit PC, add three cycles plus two cycles for each wait state. For CALL, ICALL, EICALL, RCALL, RET, RETI in devices with 22-bit PC, add five cycles plus three cycles for each wait state.

Conditional Branch Summary

Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
$Rd > Rr$	$Z \bullet (N \oplus V) = 0$	BRLT ⁽¹⁾	$Rd \leq Rr$	$Z + (N \oplus V) = 1$	BRGE*	Signed
$Rd \geq Rr$	$(N \oplus V) = 0$	BRGE	$Rd < Rr$	$(N \oplus V) = 1$	BRLT	Signed
$Rd = Rr$	$Z = 1$	BREQ	$Rd \neq Rr$	$Z = 0$	BRNE	Signed
$Rd \leq Rr$	$Z + (N \oplus V) = 1$	BRGE ⁽¹⁾	$Rd > Rr$	$Z \bullet (N \oplus V) = 0$	BRLT*	Signed
$Rd < Rr$	$(N \oplus V) = 1$	BRLT	$Rd \geq Rr$	$(N \oplus V) = 0$	BRGE	Signed
$Rd > Rr$	$C + Z = 0$	BRLO ⁽¹⁾	$Rd \leq Rr$	$C + Z = 1$	BRSH*	Unsigned
$Rd \geq Rr$	$C = 0$	BRSH/BRCC	$Rd < Rr$	$C = 1$	BRLO/BRCS	Unsigned
$Rd = Rr$	$Z = 1$	BREQ	$Rd \neq Rr$	$Z = 0$	BRNE	Unsigned
$Rd \leq Rr$	$C + Z = 1$	BRSH ⁽¹⁾	$Rd > Rr$	$C + Z = 0$	BRLO*	Unsigned
$Rd < Rr$	$C = 1$	BRLO/BRCS	$Rd \geq Rr$	$C = 0$	BRSH/BRCC	Unsigned
Carry	$C = 1$	BRCS	No carry	$C = 0$	BRCC	Simple
Negative	$N = 1$	BRMI	Positive	$N = 0$	BRPL	Simple
Overflow	$V = 1$	BRVS	No overflow	$V = 0$	BRVC	Simple
Zero	$Z = 1$	BREQ	Not zero	$Z = 0$	BRNE	Simple

Note: 1. Interchange Rd and Rr in the operation before the test, i.e., CP Rd,Rr → CP Rr,Rd