Arithmetic and Logic Instructions

Arithmetic Instructions

Mnemonic	Description	Operation	Flags	Clocks(1)
ADD Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,S,H	1 | 1 | 1 |1
ADC Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1 | 1 | 1 |1
ADIW Rd, K	Add Immediate to Word	Rd ← Rd+1:Rd + K	Z,C,N,V,S	2 | 2 | 2 | N/A
SUB Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,S,H	1 | 1 | 1 |1
SUBI Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,S,H	1 | 1 | 1 |1
SBC Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,S,H	1 | 1 | 1 |1
SBCI Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,S,H	1 | 1 | 1 |1
SBIW Rd, K	Subtract Immediate from Word	Rd+1:Rd ← Rd+1:Rd - K	Z,C,N,V,S	2 | 2 | 2 | N/A
NEG Rd	Two’s Complement	Rd ← $00 - Rd	Z,C,N,V,S,H	1 | 1 | 1 |1
INC Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1 | 1 | 1 |1
DEC Rd	Decrement	Rd ← Rd - 1	Z,N,V,S	1 | 1 | 1 |1
MUL Rd,Rr	Multiply Unsigned	R1:R0 ← Rd x Rr (UU)	Z,C	2 | 2 | 2 | N/A
MULS Rd,Rr	Multiply Signed	R1:R0 ← Rd x Rr (SS)	Z,C	2 | 2 | 2 | N/A
MULSU Rd,Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr (SU)	Z,C	2 | 2 | 2 | N/A
FMUL Rd,Rr	Fractional Multiply Unsigned	R1:R0 ← Rd x Rr<<1 (UU)	Z,C	2 | 2 | 2 | N/A
FMULS Rd,Rr	Fractional Multiply Signed	R1:R0 ← Rd x Rr<<1 (SS)	Z,C	2 | 2 | 2 | N/A
FMULSU Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← Rd x Rr<<1 (SU)	Z,C	2 | 2 | 2 | N/A

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

Logic Instructions

Mnemonic	Description	Operation	Flags	Clocks AVR(1)
AND Rd, Rr	Logical AND	Rd ← Rd • Rr	Z,N,V,S	1 | 1 | 1 |1

Logic Instructions (cont)

ANDI Rd, K	Logical AND with Immediate	Rd ← Rd • K	Z,N,V,S	1 | 1 | 1 |1
OR Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V,S	1 | 1 | 1 |1
ORI Rd, K	Logical OR with Immediate	Rd ← Rd v K	Z,N,V,S	1 | 1 | 1 |1
EOR Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V,S	1 | 1 | 1 |1
COM Rd	One's Complement	Rd ← $FF - Rd	Z,C,N,V,S	1 | 1 | 1 |1
SBR Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V,S	1 | 1 | 1 |1
CBR Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FFh - K)	Z,N,V,S	1 | 1 | 1 |1
TST Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V,S	1 | 1 | 1 |1
CLR Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V,S	1 | 1 | 1 |1
SER Rd	Set Register	Rd ← $FF	Z,N,V,S	1 | 1 | 1 |1
DES K	Data Encryption	if (H = 0) then R15:R0 ← Encrypt(R15: R0, K) else if (H = 1) then R15:R0 ← Decrypt(R15: R0, K)	NONE	N/A | 1/2 | N/A | N/A

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

Flow Control Instructions

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

Jump and Call Instructions

Mnemonic	Description	Operation	Flags	Clocks(1)
CALL k	Call Subroutine	PC ← k	NONE	4/5 | 3/4 | 3/4 | N/A
JMP k	Jump	PC ← k	NONE	3 | 3 | 3 | N/A
RCALL k	Relative Call Subroutine	PC ← PC + k + 1	NONE	3/4 | 2/3 | 2/3 | 3
ICALL	Indirect Call to (Z)	PC(15:0) ← Z PC(21:16) ← 0	NONE	3/4 | 2/3 | 2/3 | 3
RJMP k	Relative Jump	PC ← PC + k + 1	NONE	2 | 2 | 2 | 2
IJMP	Indirect Jump to (Z)	PC(15:0) ← Z PC(21:16) ← 0	NONE	2 | 2 | 2 | 2
EICALL	Extended Indirect Call to (Z)	PC(15:0) ← Z PC(21:16) ← EIND	NONE	4 | 3 | 2/3 | N/A
EIJMP	Extended Indirect Jump to (Z)	PC(15:0) ← Z PC(21:16) ← EIND	NONE	2 | 2 | 2 | N/A
RET	Subroutine Return	PC ← STACK	NONE	4/5 | 4/5 | 4/5 | 6
RETI	Interrupt Return	PC ← STACK	I	4/5 | 4/5 | 4/5 | 6

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

Compare and Skip Instructions

Mnemonic	Description	Operation	Flags	Clocks(1)

Compare and Skip Instructions (cont)				
CPSE Rd, Rr	Compare, skip if Equal	if (Rd = Rr) then PC ← PC + 2/3 else PC ← PC + 1	NONE	1/2/3 | 1/2/3 | 1/2/3 | 1/2
CP Rd, Rr	Compare	Rd - Rr	Z,C,N,V,S,H	1 | 1 | 1 | 1
CPC Rd, Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,S,H	1 | 1 | 1 | 1
CPI Rd, K	Compare with Immediate	Rd - K	Z,C,N,V,S,H	1 | 1 | 1 | 1
SBRC Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) then PC ← PC + 2/3 else PC ← PC + 1	NONE	1/2/3 | 1/2/3 | 1/2/3 | 1/2
SBRS Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) then PC ← PC + 2/3 else PC ← PC + 1	NONE	1/2/3 | 1/2/3 | 1/2/3 | 1/2
SBIC A, b	Skip if Bit in I/O Register Cleared	if (I/O(a,b) = 0) then PC ← PC + 2/3 else PC ← PC + 1	NONE	1/2/3 | 2/3/4 | 1/2/3 | 1/2
SBIS A, b	Skip if Bit in I/O Register Set	if (I/O(a,b) = 1) then PC ← PC + 2/3 else PC ← PC + 1	NONE	1/2/3 | 2/3/4 | 1/2/3 | 1/2

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

Branch Instructions				
Mnemonic	**Description**	**Operation**	**Flags**	**Clocks(1)**
BRBS s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRBC s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BREQ k	Branch if Equal (Zero)	if (Z = 1) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRNE k	Branch if Not Equal (Zero)	if (Z = 0) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRCS k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2

Branch Instructions (cont)				
BRCC k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRSH k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRLO k	Branch if Lower	if (C = 1) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRMI k	Branch if Minus	if (N = 1) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRPL k	Branch if Plus	if (N = 0) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRGE k	Branch if Greater or Equal, Signed	if (N ⊕ V = 0) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRLT k	Branch if Less Than, Signed	if (N ⊕ V = 1) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRHS k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRHC k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRTS k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRTC k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRVS k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRVC k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2

Branch Instructions (cont)

BRIE k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2
BRID k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1 else PC ← PC + 1	NONE	1/2 | 1/2 | 1/2 | 1/2

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

Data Transfer Instructions

Flash Memory Instructions

Mnemonic	Description	Operation	Flags	Clocks(1)
LPM	Load Program Memory	R0 ← (Z)	NONE	3 | 3 | 3 | N/A
LPM Rd, Z	Load Program Memory	Rd ← (Z)	NONE	3 | 3 | 3 | N/A
LPM Rd, Z+	Load Program Memory and Post-Increment	Rd ← (Z) Z ← Z + 1	NONE	3 | 3 | 3 | N/A
ELPM	Extended Load Program Memory	R0 ← (RAMPZ:Z)	NONE	3 | 3 | 3 | N/A
ELPM Rd, Z	Extended Load Program Memory	Rd ← (RAMPZ:Z)	NONE	3 | 3 | 3 | N/A
ELPM Rd, Z+	Extended Load Program Memory and Post-Increment	Rd ← (RAMPZ:Z) RAMPZ:Z ← RAMPZ:Z + 1	NONE	3 | 3 | 3 | N/A
SPM	Store Program Memory	(RAMPZ:Z) ← R1:R0	NONE	D/D | D/D | 4 | N/A
SPM Z+	Store Program Memory and Post-Increment by 2	(RAMPZ:Z) ← R1:R0 Z ← Z + 2	NONE	D/D | D/D | 4 | N/A

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

Data Memory Instructions

Mnemonic	Description	Operation	Flags	Clocks(1)
LDS Rd, k	Load Direct from data space	Rd ← (k)	NONE	2 | 2 | 3 | 2
LD Rd, X	Load Indirect	Rd ← (X)	NONE	2 | 1 | 2 | 1/2
LD Rd, X+	Load Indirect and Post-Increment	Rd ← (X) X ← X + 1	NONE	2 | 1 | 2 | 2/3
LD Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1 Rd ← (X)	NONE	2 | 2 | 2 | 2/3
LD Rd, Y	Load Indirect	Rd ← (Y)	NONE	2 | 1 | 2 | 1/2

Data Memory Instructions (cont)				
LD Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y) Y ← Y + 1	NONE	2 | 1 | 2 | 2/3
LD Rd, -Y	Load Indirect and Pre-Decrement	Y ← Y - 1 Rd ← (Y)	NONE	2 | 2 | 2 | 2/3
LDD Rd, Y + q	Load Indirect with Displacement	Rd ← (Y + q)	NONE	2 | 2 | 2 | N/A
LD Rd, Z	Load Indirect	Rd ← (Z)	NONE	2 | 1 | 2 | 1/2
LD Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z) Z ← Z + 1	NONE	2 | 1 | 2 | 2/3
LD Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1 Rd ← (Z)	NONE	2 | 2 | 2 | 2/3
LDD Rd, Z + q	Load Indirect with Displacement	Rd ← (Z + q)	NONE	2 | 2 | 2 | N/A
STS k, Rr	Store Direct to Data Space	(k) ← Rr	NONE	2 | 2 | 2 | 1
ST X, Rr	Store Indirect	(X) ← Rr	NONE	1 | 1 | 1 | 1
ST X+, Rr	Store Indirect and Post-Increment	(X) ← Rr X ← X + 1	NONE	1 | 1 | 1 | 1
ST -X, Rr	Store Indirect and Pre-Decrement	X ← X - 1 (X) ← Rr	NONE	2 | 2 | 1 | 2
ST Y, Rr	Store Indirect	(Y) ← Rr	NONE	2 | 1 | 1 | 1
ST Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr Y ← Y + 1	NONE	2 | 1 | 1 | 1
ST -Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1 (Y) ← Rr	NONE	2 | 2 | 1 | 2
STD Y + q, Rr	Store Indirect with Displacement	(Y + q) ← Rr	NONE	2 | 2 | 1 | N/A
ST Z, Rr	Store Indirect	(Z) ← Rr	NONE	2 | 1 | 1 | 1
ST Z+, Rr	Store Indirect and Post-Increment	(Z) ← Rr Z ← Z + 1	NONE	2 | 1 | 1 | 1
ST -Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1 (Z) ← Rr	NONE	2 | 2 | 1 | 2
STD Z + q, Rr	Store Indirect with Displacement	(Z + q) ← Rr	NONE	2 | 2 | 1 | N/A
PUSH Rr	Push Register on Stack	STACK ← Rr	NONE	2 | 1 | 1 | 1
POP Rd	Pop Register from Stack	Rd ← STACK	NONE	2 | 2 | 2 | 3
XCH Z, Rd	Exchange	(Z) ← Rd Rd ← (Z)	NONE	N/A | 1 | N/A | N/A
LAS Z, Rd	Load and Set	(Z) ← Rd v (Z) Rd ← (Z)	NONE	N/A | 1 | N/A | N/A

Data Memory Instructions (cont)

LAC Z, Rd	Load and Clear	(Z) ← ($FF - Rd) • (Z) Rd ← (Z)	NONE	N/A | 1 | N/A | N/A
LAT Z, Rd	Load and Toggle	(Z) ← Rd ⊕ (Z) Rd ← (Z)	NONE	N/A | 1 | N/A | N/A

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

Register Transfer Instructions

Mnemonic	Description	Operation	Flags	Clocks(1)
MOV Rd, Rr	Copy Register	Rd ← Rr	NONE	1 | 1 | 1 | 1
MOVW Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	NONE	1 | 1 | 1 | N/A
LDI Rd, K	Load Immediate	Rd ← K	NONE	1 | 1 | 1 | 1
IN Rd, A	In From I/O Location	Rd ← I/O(A)	NONE	1 | 1 | 1 | 1
OUT A, Rr	Out To I/O Location	I/O(A) ← Rr	NONE	1 | 1 | 1 | 1

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

Bit and Bit-Test Instructions

Shift and Rotation Instructions

Mnemonic	Description	Operation	Flags	Clocks(1)
LSL Rd	Logical Shift Left	Rd(n + 1) ← Rd(n) Rd(0) ← 0 C ← Rd(7)	Z,C,N,V,H	1 | 1 | 1 | 1
LSR Rd	Logical Shift Right	Rd(n) ← Rd(n + 1) Rd(7) ← 0 C ← Rd(0)	Z,C,N,V	1 | 1 | 1 | 1
ROL Rd	Rotate Left Through Carry	Rd(0) ← C Rd(n + 1) ← Rd(n) C ← Rd(7)	Z,C,N,V,H	1 | 1 | 1 | 1
ROR Rd	Rotate Right Through Carry	Rd(7) ← C Rd(n) ← Rd(n + 1) C ← Rd(0)	Z,C,N,V	1 | 1 | 1 | 1
ASR Rd	Arithmetic Shift Right	Rd(n) ← Rd(n + 1), n = 0..6	Z,C,N,V	1 | 1 | 1 | 1
SWAP Rd	Swap Nibbles	R(3..0) ↔ R(7..4)	NONE	1 | 1 | 1 | 1
SBI A, b	Set Bit in I/O Register	I/O(A,b) ← 1	NONE	2 | 1 | 1 | 1
CBI A, b	Clear Bit in I/O Register	I/O(A,b) ← 0	NONE	2 | 1 | 1 | 1
BST Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1 | 1 | 1 | 1

Shift and Rotation Instructions (cont)

BLD Rd, b	Bit load from T to Register	Rd(b) ← T	NONE	1 | 1 | 1 | 1

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

Status Register Bit Instructions

Mnemonic	Description	Operation	Flags	Clocks(1)
BSET s	Flag Set	SREG(s) ← 1	SREG(s)	1 | 1 | 1 | 1
BCLR s	Flag Clear	SREG(s) ← 0	SREG(s)	1 | 1 | 1 | 1
SEC	Set Carry	C ← 1	C	1 | 1 | 1 | 1
CLC	Clear Carry	C ← 0	C	1 | 1 | 1 | 1
SEN	Set Negative Flag	N ← 1	N	1 | 1 | 1 | 1
CLN	Clear Negative Flag	N ← 0	N	1 | 1 | 1 | 1
SEZ	Set Zero Flag	Z ← 1	Z	1 | 1 | 1 | 1
CLZ	Clear Zero Flag	Z ← 0	Z	1 | 1 | 1 | 1
SEI	Global Interrupt Enable	I ← 1	I	1 | 1 | 1 | 1
CLI	Global Interrupt Disable	I ← 0	II	1 | 1 | 1 | 1
SES	Set Signed Test Flag	S ← 1	S	1 | 1 | 1 | 1
CLS	Clear Signed Test Flag	S ← 0	S	1 | 1 | 1 | 1
SEV	Set Two’s Complement Overflow	V ← 1	V	1 | 1 | 1 | 1
CLV	Clear Two’s Complement Overflow	V ← 0	V	1 | 1 | 1 | 1
SET	Set T in SREG	T ← 1	T	1 | 1 | 1 | 1
CLT	Clear T in SREG	T ← 0	T	1 | 1 | 1 | 1
SEH	Set Half Carry Flag in SREG	H ← 1	H	1 | 1 | 1 | 1
CLH	Clear Half Carry Flag in SREG	H ← 0	H	1 | 1 | 1 | 1

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

MCU Control Instructions

MCU Control Instructions

Mnemonic	Description	Operation	Flags	Clocks(1)
BREAK	Break	-	NONE	1 | 1 | 1 | 1
NOP	No Operation	-	NONE	1 | 1 | 1 | 1
SLEEP	Sleep	-	NONE	1 | 1 | 1 | 1
WDR	Watchdog Reset	-	NONE	1 | 1 | 1 | 1

1. Number of clocks in AVR 8-bit CPU version (AVR | AVRxm | AVRxt | AVRrc)

Assembler Directives

Definição de Segmentos

.CSEG	Defines the start of a Code Segment.
.CSEGSIZE[1]	Specifies the size of the program memory block.
.DSEG	Defines the start of a Data Segment.
.ESEG	Defines the start of an EEPROM Segment.
.ORG	Sets the location counter to an absolute value.
.OVERL-AP/.NO-OVERLAP[2]	Marks a section that will be allowed to overlap code/data with code/data defined elsewhere, without any error or warning messages being generated.

1. This directive is specific to AT94K devices, since they have a user configurable memory partition between the AVR Program memory and the data memory.
2. These directives are for projects with special needs and should normally not be used.

Definição de Símbolos

.DEF	Defines a symbol to a register.
.EQU	Assigns a value to a label. This value cannot be reassigned later in the program.
.SET	Assigns a value to a label. This value can be reassigned later in the program.
.UNDEF	Undefines a symbol previously defined with the .DEF directive.

Definição de Variáveis

.BYTE	Reserves blocks of 1 byte of memory resources in the SRAM or EEPROM.
.DB	Reserves blocks of 1 byte of memory resources in the SRAM or EEPROM.
.DD	
.DQ	
.DW	