Complete Instruction Set Summary

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
Arithmetic and Logic Instructions						
ADD	Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,S,H	1	
ADC	Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1	
ADIW(1)	Rd, K	Add Immediate to Word	Rd ← Rd + 1:Rd + K	Z,C,N,V,S	2	
SUB	Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,S,H	1	
SUBI	Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,S,H	1	
SBC	Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,S,H	1	
SBCI	Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,S,H	1	
SBIW(1)	Rd, K	Subtract Immediate from Word	Rd + 1:Rd ← Rd + 1:Rd - K	Z,C,N,V,S	2	
AND	Rd, Rr	Logical AND	Rd ← Rd • Rr	Z,N,V,S	1	
ANDI	Rd, K	Logical AND with Immediate	Rd ← Rd • K	Z,N,V,S	1	
OR	Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V,S	1	
ORI	Rd, K	Logical OR with Immediate	Rd ← Rd v K	Z,N,V,S	1	
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ 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 ← $00 - Rd	Z,C,N,V,S,H	1	
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V,S	1	
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FFh - K)	Z,N,V,S	1	
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1	
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V,S	1	
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V,S	1	
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V,S	1	
SER	Rd	Set Register	Rd ← $FF	None	1	
MUL(1)	Rd,Rr	Multiply Unsigned	R1:R0 ← Rd x Rr (UU)	Z,C	2	
MULS(1)	Rd,Rr	Multiply Signed	R1:R0 ← Rd x Rr (SS)	Z,C	2	
MULSU(1)	Rd,Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr (SU)	Z,C	2	
FMUL(1)	Rd,Rr	Fractional Multiply Unsigned	R1:R0 ← Rd x Rr<<1 (UU)	Z,C	2	
FMULS(1)	Rd,Rr	Fractional Multiply Signed	R1:R0 ← Rd x Rr<<1 (SS)	Z,C	2	
FMULSU(1)	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← Rd x Rr<<1 (SU)	Z,C	2	

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)
						1/2

Branch Instructions						
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2	
IJMP(1)		Indirect Jump to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	2	
EIJMP(1)		Extended Indirect Jump to (Z)	PC(15:0) ← Z, PC(21:16) ← EIND	None	2	
JMP(1)	k	Jump	PC ← k	None	3	

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
RCALL	k	Relative Call Subroutine	PC ← PC + k + 1	None	3 / 4[3][5]	2 / 3[3]
ICALL[1]		Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	3 / 4[3]	2 / 3[3]
EICALL[1]		Extended Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← EIND	None	4 [3]	3 [3]
CALL[1]	k	call Subroutine	PC ← k	None	4 / 5[3]	3 / 4[3]
RET		Subroutine Return	PC ← STACK	None	4 / 5[3]	
RETI		Interrupt Return	PC ← STACK	I	4 / 5[3]	
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	2 / 3 / 4
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	2 / 3 / 4
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[1]	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1	
LDI	Rd, K	Load Immediate	Rd ← K	None	1	
LDS[1]	Rd, k	Load Direct from data space	Rd ← (k)	None	1[5]/2[3]	2[3][4]
LD[2]	Rd, X	Load Indirect	Rd ← (X)	None	1[5]2[3]	1[3][4]

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
LD[2]	Rd, X+	Load Indirect and Post-Increment	Rd ← (X) X ← X + 1	None	2[3]	1[3][4]
LD[2]	Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1, ← X - 1 Rd ← (X) ← (X)	None	2[3]/3[5]	2[3][4]
LD[2]	Rd, Y	Load Indirect	Rd ← (Y) ← (Y)	None	1[5]/2[3]	1[3][4]
LD[2]	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y) Y ← Y + 1	None	2[3]	1[3][4]
LD[2]	Rd, -Y	Load Indirect and Pre-Decrement	Y ← Y - 1 Rd ← (Y)	None	2[3]/3[5]	2[3][4]
LDD[1]	Rd, Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2[3]	2[3][4]
LD[2]	Rd, Z	Load Indirect	Rd ← (Z)	None	1[5]/2[3]	1[3][4]
LD[2]	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2[3]	1[3][4]
LD[2]	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2[3]/3[5]	2[3][4]
LDD[1]	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2[3]	2[3][4]
STS[1]	k, Rr	Store Direct to Data Space	(k) ← Rd	None	1[5]/2[3]	2[3]
ST[2]	X, Rr	Store Indirect	(X) ← Rr	None	1[5]/2[3]	1[3]
ST[2]	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	1[5]/2[3]	1[3]
ST[2]	-X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	2[3]	2[3]
ST[2]	Y, Rr	Store Indirect	(Y) ← Rr	None	1[5]/2[3]	1[3]
ST[2]	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr, Y ← Y + 1	None	1[5]/2[3]	1[3]
ST[2]	-Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1, (Y) ← Rr	None	2[3]	2[3]
STD[1]	Y+q, Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2[3]	2[3]
ST[2]	Z, Rr	Store Indirect	(Z) ← Rr	None	1[5]/2[3]	1[3]
ST[2]	Z+, Rr	Store Indirect and Post-Increment	(Z) ← Rr Z ← Z + 1	None	1[5]/2[3]	1[3]
ST[2]	-Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1	None	2[3]	2[3]
STD[1]	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2[3]	2[3]
LPM[1][2]		Load Program Memory	R0 ← (Z)	None	3	3
LPM[1][2]	Rd, Z	Load Program Memory	Rd ← (Z)	None	3	3
LPM[1][2]	Rd, Z+	Load Program Memory and Post-Increment	Rd ← (Z), Z ← Z + 1	None	3	3
ELPM[1]		Extended Load Program Memory	R0 ← (RAMPZ:Z)	None	3	
ELPM[1]	Rd, Z	Extended Load Program Memory	Rd ← (RAMPZ:Z)	None	3	
ELPM[1]	Rd, Z+	Extended Load Program Memory and Post-Increment	Rd ← (RAMPZ:Z), Z ← Z + 1	None	3	
SPM[1]		Store Program Memory	(RAMPZ:Z) ← R1:R0	None	-	-
SPM[1]	Z+	Store Program Memory and Post-Increment by 2	(RAMPZ:Z) ← R1:R0, Z ← Z + 2	None	-	-
IN	Rd, A	In From I/O Location	Rd ← I/O(A)	None	1	
OUT	A, Rr	Out To I/O Location	I/O(A) ← Rr	None	1	
PUSH[1]	Rr	Push Register on Stack	STACK ← Rr	None	2	1[3]
POP[1]	Rd	Pop Register from Stack	Rd ← STACK	None	2	2[3]

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
XCH	Z, Rd	Exchange	(Z) ← Rd, Rd ← (Z)	None	1	
LAS	Z, Rd	Load and Set	(Z) ← Rd v (Z) Rd ← (Z)	None	1	
LAC	Z, Rd	Load and Clear	(Z) ← ($FF – Rd) • (Z) Rd ← (Z)	None	1	
LAT	Z, Rd	Load and Toggle	(Z) ← Rd ⊕ (Z) Rd ← (Z)	None	1	
Bit and Bit-test Instructions						
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0, C ← Rd(7)	Z,C,N,V,H	1	
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0, C ← Rd(0)	Z,C,N,V	1	
ROL	Rd	Rotate Left Through Carry	Rd(0) ← C, Rd(n+1) ← Rd(n), C ← Rd(7)	Z,C,N,V,H	1	
ROR	Rd	Rotate Right Through Carry	Rd(7) ← C, Rd(n) ← Rd(n+1), C ← Rd(0)	Z,C,N,V	1	
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1	
SWAP	Rd	Swap Nibbles	Rd(3..0) ↔ Rd(7..4)	None	1	
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1	
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1	
SBI	A, b	Set Bit in I/O Register	I/O(A, b) ← 1	None	1[5]2	1
CBI	A, b	Clear Bit in I/O Register	I/O(A, b) ← 0	None	1[5]/2	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1	
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1	
SEC		Set Carry	C ← 1	C	1	
CLC		Clear Carry	C ← 0	C	1	
SEN		Set Negative Flag	N ← 1	N	1	
CLN		Clear Negative Flag	N ← 0	N	1	
SEZ		Set Zero Flag	Z ← 1	Z	1	
CLZ		Clear Zero Flag	Z ← 0	Z	1	
SEI		Global Interrupt Enable	I ← 1	I	1	
CLI		Global Interrupt Disable	I ← 0	I	1	
SES		Set Signed Test Flag	S ← 1	S	1	
CLS		Clear Signed Test Flag	S ← 0	S	1	
SEV		Set Two’s Complement Overflow	V ← 1	V	1	
CLV		Clear Two’s Complement Overflow	V ← 0	V	1	
SET		Set T in SREG	T ← 1	T	1	
CLT		Clear T in SREG	T ← 0	T	1	
SEH		Set Half Carry Flag in SREG	H ← 1	H	1	
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1	
MCU Control Instructions						
BREAK[1]		Break	(See specific descr. for BREAK)	None	1	

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
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. Cycle times for Data memory accesses assume internal memory accesses, and are not valid for accesses via the external RAM interface.
4. One extra cycle must be added when accessing Internal SRAM.
5. Number of clock cycles for Reduced Core tinyAVR.