Legend

Rd – register destination Rr – register source (can also be designated Rs)

K – constant S – bit locations in either I/O or working registers. P = ports b or a = bit locations

Encoding X's are don't care conditions. There can be either a 1 or 0 in the location. Does not affect the instruction.

X-register – Index register X, made up of XL, and XH. This is considered a pointer register for indirect addressing mode.

Y-register – Index register Y, made up of YL, and YH. This is considered a pointer register for indirect addressing mode.

Z-register – Index register Z, made up of ZL, and ZH. This is considered a pointer register for indirect addressing mode.

Refer to Data Sheet for Specific IC instructions resources. Not all instructions work on every device and more advanced devices may have additional instructions or different labels for registers etc.

Except from: AVR RISC Microcontroller Handbook by Claus Kühnel. Newnes, 1998. ISBN: 0-7506-9963-9

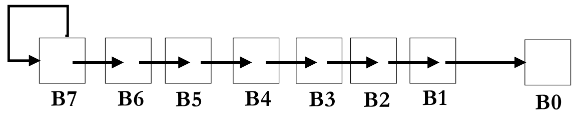
ADC	**Add with Carry**	**Comments**
Syntax	ADC Rd, Rr	Rd = destination; Rr = resident
Operands	0≤d≤31, 0≤r≤31	working registers R0 through R31
Operation	Rd ←Rd+Rr+C	Add Rd to Rr with C and store in Rd
Flags Affected	H, S, V, N, Z, C	In SREG
Encoding	0001 11rd dddd rrrr	
Description	Adds the registers Rd and Rr and the contents of the C flag and places the result in the destination register Rd.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	; Add R1:R0 to R3:R2 add r2, r0 ; Add low bytes adc r3, r1 ; Add high bytes with carry	

ADD	**Add without Carry**	**Comments**
Syntax	ADD Rd, Rr	Rd = destination ; Rr = resident
Operands	0≤d≤31, 0≤r≤31	working registers R0 through R31
Operation	Rd ←Rd+Rr	Add Rd to Rr with C and store in Rd
Flags Affected	H, S, V, N, Z, C	In SREG
Encoding	0000 11rd dddd rrrr	
Description	Adds the registers Rd and Rr and places the result in the destination register Rd.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	; Add R1:R0 to R3:R2 add r1, r2 ; Add r2 to r1 add r28, r28 ; Add r28 to itself	

ADIW	**Add Immediate to Word**	**Comments**
Syntax	ADIW Rd1, K	Rd = destination; K = constant
Operands	d1 ε{24,26,28,30} , 0≤K≤63	registers R24, R26, R28, R30 and K 0 to 63 (immediate value)
Operation	Rdh:Rdl ←Rdh:Rdl+K	Add Rdh to Rdl with K and store in Rdh:Rdl
Flags Affected	S, V, N, Z, C	In SREG
Encoding	1001 0110 KKdd KKKK	
Description	Adds the immediate value (0-63) to a register pair and places the result in the register pair. This instruction operates on the upper four register pairs and is well suited for operations on the pointer registers.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	; Add R1:R0 to R3:R2 add r1, r2 ; Add r2 to r1 add r28, r28 ; Add r28 to itself	

AND	**Logical AND**	**Comments**
Syntax	AND Rd, Rr	Rd = destination; Rr = resident
Operands	0≤d≤31, 0≤r≤31	(d,r) working registers 0 to 31
Operation	Rd←Rd AND Rr	And Rd to Rr and store in Rd
Flags Affected	S, V, N, Z	In SREG
Encoding	0010 00rd dddd rrrr	
Description	Performs a logical AND between the contents of registers Rd and Rr and places the result in the destination register Rd.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	and r2, r3 ;Bitwise AND of r2 and r3, result in r2 ldi r16, 1 ; load 0000 0001 to r16 immediately and r2, r16 ; isolate bit 0 in r2 with a "Mask Byte"	

ANDI	**Logical AND with Immediate**	**Comments**
Syntax	ANDI Rd,k	Rd = destination; K = constant
Operands	0≤d≤31, 0≤K≤255	registers R0-R31 and K 0 to 255 (immediate value)
Operation	Rd←Rd AND k	And immediately Rd to K and store in Rd
Flags Affected	S, V, N, Z	In SREG
Encoding	0011 kkkk dddd kkkk	
Description	Performs a logical AND between the contents of registers Rd and a byte constant and places the result in the destination register Rd.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	andi r17, $0F ;Clear upper nibble of r17 andi r18, $10 ; Isolate bit 4 in r18 andi r19, $AA ; Clear odd bits of r19	$0F represents a hex number of 0F 0b00001111 represents a binary number of 0F 15 represents a decimal number of 15

ASR	**Arithmetic Shift Right**	**Comments**
Syntax	ASR Rd	Rd = destination
Operands	0≤d≤31	registers R0-R31
Operation		Shifts all bits in Register Rd one place to the right. Uses the Carry Flag
Flags Affected	S, V, N, Z, C	In SREG
Encoding	1001 010d dddd 0101	
Description	Shifts all bits in Register Rd one place to the right. Bit 7 is held constant and bit0 is loaded into the C flag.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	ldi r16, $10 ; load 16d into r16 register asr r16 ; r16 = r16/2 because of shift ldi r17, $FC ; load -4d into r17 asr r17 ; r17 = r17/2	$0F represents a hex number of 0F 0b00001111 represents a binary number of 0F 15 represents a decimal number of 15

BCLR	**Bit Clear in SREG**	**Comments**
Syntax	BCLR s	Clears a bit location 0 to 7 in SREG
Operands	0≤s≤7	s = bit location
Operation	SREG(s)←0	And immediately Rd to K and store in Rd
Flags Affected	I, T, H, S, V, N, Z, C	In SREG
Encoding	1001 0100 1sss 1000	
Description	Clears a single flag in SREG	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	bclr 0 ; Clears bit 0 in SREG bclr 7 ; Disables interrupts	

BLD	**Bit Load from T Flag in SREG to a Bit in Register**	**Comments**
Syntax	BLD Rd, b	Rd = destination; b = bit
Operands	0≤d≤31, 0≤b≤7	registers R0-R31 and bits 0 to 7
Operation	Rd(b)←T	And immediately Rd to K and store in Rd
Flags Affected	none	In SREG
Encoding	1111 100d dddd Xbbb	
Description	Copies the T flag in SREG to bit b in Register Rd	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	bst r1, 2 ; store bit2 if r1 in T flag bld r0, 4 ; load T flag in bit 4 of r0	

BRBC	**Branch If Bit in SREG is Cleared**	**Comments**
Syntax	BRBC s, K	S = bit location; k = constant
Operands	0≤s≤7, -64≤K≤63	Bit location 0 to 7; values -64 to +63
Operation	if SREG(s) = 0 then branch k steps	If bit location = 0 then branch k steps
Flags Affected	none	In SREG
Encoding	1111 01KK KKKK Ksss	
Description	Tests a single bit in SREG and branches relatively to PC if the bit is cleared. The parameter k is the offset from PC and is represented in twos complement form	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	cpi r20, 5 ; compare r20 to the value 5 brbc 1,noteq ; branch to label noteq, ; if zero flag is cleared.	

BRBS	**Branch If Bit in SREG is SET**	**Comments**
Syntax	BRBS s, K	S = bit location, k = constant
Operands	0≤s≤7,-64 0≤K≤63	Bit location 0 to 7; values -64 to +63
Operation	If SREG(s) = 1 then branch k steps	If bit location = 1 then branch k steps
Flags Affected	None	In SREG
Encoding	1111 00KK KKKK Ksss	
Description	Tests a single bit in SREG and branches relatively to PC if the bit is set. The parameters k is the offset from PC and is represented in twos complement form.	
Words	1	16 bit instruction
Cycles	1 (true), 2 (false)	1 or 2 machine cycles based on decision
Example	bst r0, 3 ; load T flag with bit3 of r0 brbs 6, bitset ; Branch to label bitset, ; if T flag was set	

BRCC	**Branch if Carry is Cleared**	**Comments**
Syntax	BRCC k	k = constant
Operands	-64≤k≤63	k values from -64 to +63
Operation	if C = 0 then branch k steps	If carry flag is zero then branch k steps
Flags Affected	None	In SREG
Encoding	1111 01kk kkkk k000	
Description	Tests the Carry flag (C) and branches relatively to PC if C is cleared. The parameter k is the offset from PC and is represented in two's complement form (equivalent to BRBC 0,k)	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	add r22, r23 ; Add r23 to r22 brcc nocarry ; Branch to the label nocarry ; if carry flag is cleared	

BRCS	**Branch If Carry is Set**	**Comments**
Syntax	BRCS k	k = constant
Operands	-64≤k≤63	k is a value between -64 and 63
Operation	if C = 1 then branch k steps	Carry bit is set and branch k steps
Flags Affected	None	In SREG
Encoding	1111 00kk kkkk k000	
Description	Tests the Carry flag (C) and branches relatively to PC if C is set. The parameter k is the offset from PC and is represented in two's complement form (equivalent to BRBS 0,k).	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycle(s) based on decision
Example	cpi r26, $56 ; compare r26 with the value $56 brcs carry ; Branch to label carry ; if carry flag is set	

BREQ	**Branch if Equal**	**Comments**
Syntax	BREQ k	K = constant
Operands	-64≤K≤63	K can be between -64 and +63
Operation	if Z = 1 then branch k steps	Z flag set (result is 0) then branch k steps
Flags Affected	None	In SREG
Encoding	1111 00kk kkkk k001	
Description	Tests the Zero flag (Z) and branches relatively to PC if Z is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB, or SUBI, the branch will occur if, and only if, the unsigned or signed binary number represented in Rd was equal to the unsigned or signed binary number represented in Rr. The parameter k is the offset from PC and is represented in two's complement form (equivalent to BRBS 1, k).	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	cp r1, r0 ; compare r1 to r0 breq equal ; Branch to label equal, ; if registers are equal ; (e.g., zero flag is set)	

BRGE	**Branch If Greater or Equal (Signed)**	**Comments**
Syntax	BRGE k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	if S = 0 then branch k steps	If S = 0 then branch k steps
Flags Affected	None	In SREG
Encoding	1111 01kk kkkk k100	
Description	Tests the Signed flag (S) and branches relatively to PC if S is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB, or SUBI, the branch will occur if, and only if, the signed binary number represented in Rd was greater than, or equal to, the signed binary number represented in Rr. The parameter k is the offset from PC and is represented in two's complement form (equivalent to BRBS 1, k)	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	cp r11, r12 ; compare r11 to r12 brge greateq ; Branch to label greateq ; (e.g., signed flag is cleared)	

BRHC	**Branch if Half Carry Flag is Cleared**	**Comments**
Syntax	BRHC k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If H = 0 then branch k steps	H flag clear (no carry from low nibble to high nibble) then branch k steps
Flags Affected	None	In SREG
Encoding	1111 01kk kkkk k101	
Description	Tests the Half Carry flag (H) and branches relatively to PC of H is cleared. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBC 5, k)	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	brhc hclear ; Branch if half carry is cleared …. hclear: nop ; Branch destination (do nothing)	

BRHS	**Branch if Half Carry Flag is Set**	**Comments**
Syntax	BRHS, k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If H = 1 then branch k steps	H flag set (carry from low nibble to high nibble) then branch k steps
Flags Affected	None	In SREG
Encoding	1111 00kk kkkk k101	
Description	Tests the Half Carry flag (H) and branches relatively to PC if H is set. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBS 5, k).	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	brhs hset ; Branch if half carry is set …. hset: nop ; Branch destination (do nothing)	

BRIE	**Branch if Global Interrupt is Enabled**	**Comments**
Syntax	BRIE k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If I = 1 then branch k steps	Interrupt (I) flag set then branch k steps
Flags Affected	None	In SREG
Encoding	1111 00kk kkkk k111	
Description	Tests the Global Interrupt flag (I) and branches relatively to PC if I is set. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instructions BRBS 7, k)	Note: There are local interrupt registers for different resources on AVR devices.
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	brie inten ; Branch if interrupt enabled ….. inten: nop ; Branch destination	

BRLO	**Branch if Lower**	**Comments**
Syntax	BRLO k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If C = 1 then branch k steps	Carry (C) flag set then branch k steps
Flags Affected	None	In SREG
Encoding	1111 00kk kkkk k000	
Description	Tests the Carry flag (C) and branches relatively to PC of C is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB, or SUBI, then branch will occur if, and only if, the unsigned binary number represented in Rd was smaller than the unsigned binary number represented in Rr. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBS 0, k)	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	eor r19, r19 ; clear r19 Loop: inc r19 ; increase r19 …. cpi r19, $10 ;compare r19 to $10 brlo loop ; branch if r19<$10 (unsigned) nop ; exit from loop (do nothing)	

BRLT	**Branch if Less Than (Signed)**	**Comments**
Syntax	BRLT, k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If S = 1 then branch k steps	Signed (S) flag set then branch k steps
Flags Affected	None	In SREG
Encoding	1111 00kk kkkk k100	
Description	Tests the Sign flag (S) and branches relatively to PC if S is set. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBS 2, k).	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	cp r16, r1 ; Compare r16 to r1 brlt less ; Branch if r16 < r1 (signed) …. less: nop ; Branch destination (do nothing)	

BRMI	**Branch if Minus**	**Comments**
Syntax	BRMI, k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If N = 1 then branch k steps	Negative (N) flag is cleared branch k steps
Flags Affected	None	In SREG
Encoding	1111 00kk kkkk k010	
Description	Tests the Negative flag (N) and branches relatively to PC if N is set. This instruction branches relatively to PC in either. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBS 2, k).	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	subi r18, 4 ; subtract 4 from r18 brmi neg ; branch if result negative …. Neg: nop ; branch destination (do nothing)	

BRNE	**Branch if Not Equal**	**Comments**
Syntax	BRNE, k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If Z = 0 then branch k steps	Zero (Z) flag is cleared branch k steps
Flags Affected	None	In SREG
Encoding	1111 01kk kkkk k001	
Description	Tests the Zero flag (Z) and branches relatively to PC if Z is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB, or SUBI, the branch will occur if, and only if, the unsigned or signed binary number represented in Rd was not equal to the unsigned or signed binary number represented in Rr. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBC 1, k).	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	eor r27, r27 ; Clear r27 loop: inc r27 ; increment r27 …. cpi r27, 5 ; compare r27 to 5 immediately brne loop ; branch if r27 < > 5 nop ; loop exit (do nothing)	

BRPL	**Branch if Plus**	**Comments**
Syntax	BRPL, k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If N = 0 then branch k steps	Negative (N) flag cleared branch k steps
Flags Affected	None	In SREG
Encoding	1111 01kk kkkk k010	
Description	Tests the Negative flag (N) and branches relatively to PC if N is cleared. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBC 2, k).	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	subi r26, $50 ; subtract $50 from r26 brpl positive ; branch if r26 is positive …… positive: nop ; branch destination (do nothing).	

BRSH	**Branch if Same or Higher (Unsigned**	**Comments**
Syntax	BRSH, k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If C = 0 then branch k steps	Carry (C) flag cleared branch k steps
Flags Affected	None	In SREG
Encoding	1111 01kk kkkk k000	
Description	Tests the Carry flag (C) and branches relatively to PC if C is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB, or SUBI, the branch will occur if, and only if, the equal to, the unsigned binary number represented in Rr. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBC 0, k).	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	subi r19, 4 ; Subtract 4 from r19 brsch highsm ; branch to label highsm …. ; if r19 ≥ 4 ; (e.g., carry flag is cleared)	

BRTC	**Branch if T flag is Cleared**	**Comments**
Syntax	BRTC, k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If T = 0 then branch k steps	Temporary (T) flag cleared branch k steps (User defined bit)
Flags Affected	None	In SREG
Encoding	1111 01kk kkkk k100	
Description	Tests the T flag and branches relatively to PC if T is cleared. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBC 6, k).	Note: not all microcontrollers have a user defined bit – feel lucky!
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	bst r3, 5 ; store bit 5 of r3 in T flag brtc tclear ; branch if T bit was clear …. tclear: nop ; branch destination (do nothing)	

BRTS	**Branch if T flag is Set**	**Comments**
Syntax	BRTS, k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If T = 1 then branch k steps	Temporary (T) flag set branch k steps (User defined bit)
Flags Affected	None	In SREG
Encoding	1111 00kk kkkk k110	
Description	Tests the T flag and branches relatively to PC if T is set. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBS 6, k).	Note: not all microcontrollers have a user defined bit to use. Feel luck!
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	bst r3, 5 ; store bit 5 of r3 in T flag brts tset ; branch if T bit was set …. tset: nop ; branch destination (do nothing)	

BRVC	**Branch if Overflow Flag is Cleared**	**Comments**
Syntax	BRVC k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If V = 0 then branch k steps	Overflow (V) flag cleared branch k steps
Flags Affected	None	In SREG
Encoding	1111 01kk kkkk k011	
Description	Test the Overflow flag (V) and branches relatively to PC if V is cleared. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBC 3, k)	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	add r3, r4 ; add r3 to r4 brvc noover ; branch if V bit was clear …. noover: nop ; branch destination (do nothing)	

BRVS	**Branch if Overflow Flag is Set**	**Comments**
Syntax	BRVS k	K = constant
Operands	-64≤k≤63	K can be between -64 and +63
Operation	If V = 1 then branch k steps	Overflow (V) flag set branch k steps
Flags Affected	None	In SREG
Encoding	1111 00kk kkkk k011	
Description	Test the Overflow flag (V) and branches relatively to PC if V is set. The parameter k is the offset from PC and is represented in two's complement form (equivalent to instruction BRBS 3, k)	
Words	1	16 bit instruction
Cycles	1 (false), 2 (true)	1 or 2 machine cycles based on decision
Example	add r3, r4 ; add r3 to r4 brvs overfl ; branch if V bit was set …. overfl: nop ; branch destination (do nothing)	

BSET	**Branch Set in SREG**	**Comments**
Syntax	BSET, s	S = flag bit in SREG
Operands	0≤s≤7	S can be between 0 and 7
Operation	SREG(s)←1	Sets a flag bit in the SREG
Flags Affected	Corresponding flag is set	In SREG
Encoding	1001 0100 0sss 1000	
Description	Sets a single flag or bit in the SREG	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	bset 6 ; Set T flag bset 7 ; Enable interrupt	

BST	**Bit Store from Bit in Register to T Flag in SREG**	**Comments**
Syntax	BST Rd, b	Store bit (b) value from register in T flag
Operands	0≤d≤31, 0≤b≤7	(d) Working registers 0-31 and (b) bits 0 to 7
Operation	T🡐0 if bit b in register Rd is cleared, T🡐 1 otherwise	Clears or Sets the T flag based on bit value in working register Rd
Flags Affected	T flag	In SREG
Encoding	1111 101d dddd Xbbb	X is don't care condition
Description	Stores bit b from Rd to the T flag in SREG (status register - bit 6).	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	; Copy bit bst r1, 2 ; Store bit 2 of r1 in T flag bld r0, 4 ; Load T into bit 4 of r0	

CALL	**Long Call to a Subroutine**	**Comments**
Syntax	Call k	K = constant
Operands	0≤k≤64k	Call a subroutine between 0 to 64k away
Operation	PC 🡐 k	The value represented by the subroutine label into PC
Flags Affected	None	In SREG
Encoding	1001 010k kkkk 111k kkkk kkkk kkkk kkkk	
Description	Calls to a subroutine within the entire program memory. The return address (to the instruction CALL) will be stored onto the stack. (See also RCALL).	
Words	2	32 bit instruction
Cycles	4	4 machine cycles
Example	mov r16, r0 ; Copy r0 to r16 call check ; call subroutine "check" nop ; continue (do nothing) …. check: cpi r16, $42 ; Check if r16 = $42 breq error ; branch if equal ret ; return from subroutine ….. error: rjmp error ; Infinite loop to nowhere	Note: Some assemblers are case-sensitive on labels. Also check for restricted names and symbols that cannot be used in the label.

CBI	**Clear Bit in I/O Register**	**Comments**
Syntax	CBI P, b	Clear bit location (b) at I/O register (P)
Operands	0≤P≤31, 0≤b≤7	(P) I/O registers and (b) bits 0 to 7
Operation	I/O (P, b)← 0	Clears the bit location at the I/O register location
Flags Affected	None	In SREG
Encoding	1001 1000 PPPP Pbbb	
Description	Clears a specified bit in an I/O register. This instruction operates on the lower 32 I/O registers 0-31	
Words	1	16 bit instruction
Cycles	2	2 machine cycle
Example	cbi $12, 7 ; Clear bit 7 in PortD	

CBR	**Clear Bits in Register**	**Comments**
Syntax	CBR Rd, k	Rd is working register and k is constant
Operands	0≤d≤31, 0≤k≤255	(d) working registers, k 0 to 255 or $00 to $FF
Operation	Rd←Rd AND ($FF-K)	Clears the bit(s) in working register.
Flags Affected	S, V, N, Z	In SREG
Encoding	0111 kkkk dddd kkkk k=($FF-k)	
Description	Clears the specified bits in register Rd. Performs the logical AND between the contents of register Rd and the complement of the constant mask k. The result will be placed in register Rd.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	cbr r16, $F0 ; clear upper nibble of r16 cbr r18, 1 ; clear bit 0 in r18	

CLC	**Clear Carry Flag**	**Comments**
Syntax	CLC	Clear Carry (C) flag
Operands	None	
Operation	C←0	Clears Carry flag in SREG
Flags Affected	C	In SREG
Encoding	1001 0100 1000 1000	
Description	Clears the Carry flag (C) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	clc ; Clear the Carry Flag	

CLH	**Clear Half Carry Flag**	**Comments**
Syntax	CLH	Clear Half Carry (H) flag
Operands	None	
Operation	H← 0	Clears the H bit
Flags Affected	H	In SREG
Encoding	1001 0100 1101 1000	
Description	Clears the Half Carry flag (H) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	clh ; clear the Half Carry flag	

CLI	**Clear Global Interrupt Flag**	**Comments**
Syntax	CLI	Clear Interrupt (I) flag
Operands	None	
Operation	I← 0	Clears the Interrupt (I) bit in SREG
Flags Affected	I	In SREG
Encoding	1001 0100 1111 1000	
Description	Clears the Global Interrupt flag (I) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	cli ; Disable interrupts in r11, $16 ; Read PortB sei ; Enable interrupts	

CLN	**Clear Negative Flag**	**Comments**
Syntax	CLN	Clear Negative (N) bit
Operands	None	
Operation	N← 0	Clears the Negative (N) flag in SREG
Flags Affected	N	In SREG
Encoding	1001 0100 1010 1000	
Description	Clears the Negative Flag (N) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	cln ; clear negative flag	

CLR	**Clear Register**	**Comments**
Syntax	CLR Rd	Clear a working register
Operands	0≤d≤31	(d) working register from 0 to 31
Operation	Rd← 0	Clears all bits in working register Rd
Flags Affected	S, V, N, Z	In SREG
Encoding	0010 01dd dddd dddd	
Description	Clears a register. This instruction performs an exclusive OR between a register and itself. This will clear all bits in the register	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	clr r18 ; clear r18 loop: inc r18 ; increase r18 by 1 ….. cpi r18, $50 ; compare r18 to $50 brne loop ; if not equal branch to loop	

CLS	**Clear Sign Flag**	**Comments**
Syntax	CLS	Clear Sign (S) bit
Operands	None	
Operation	S←0	Clears Sign (S) Flag in SREG
Flags Affected	S	In SREG
Encoding	1001 0100 1100 1000	
Description	Clears the Sign Flag (S) in the SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	cls ; clear sign flag	

CLT	**Clear Test Flag**	**Comments**
Syntax	CLT	Clear Test (T) bit (user defined)
Operands	None	(P) I/O registers and (b) bits 0 to 7
Operation	T← 0	Clears Test (T) Flag in SREG
Flags Affected	T	In SREG
Encoding	1001 0100 1110 1000	
Description	Clears the T flag in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	clt ; clear T flag	

CLV	Clear Overflow Flag	Comments
Syntax	CLV	Clear Overflow (V) bit
Operands	None	
Operation	V← 0	Clears Overflow (V) flag in SREG
Flags Affected	V	In SREG
Encoding	1001 0100 1011 1000	
Description	Clears the Overflow flag (V) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	clv ; clear V flag	

CLZ	**Clear Zero Flag**	**Comments**
Syntax	CLZ	Clear Zero (Z) bit
Operands	None	
Operation	Z← 0	Clears Zero (Z) flag in SREG
Flags Affected	Z	In SREG
Encoding	1001 0100 1111 1000	
Description	Clears the Zero Flag (Z) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	clz ; clear Z flag	

COM	**One's Complement**	**Comments**
Syntax	COM Rd	One's Complement on Rd; 0 to 1 and 1 to 0 for each value
Operands	0≤d≤31	(d) working registers
Operation	Rd ← $FF – Rd	One's complement on Rd
Flags Affected	S, V, N, Z, C	In SREG
Encoding	1001 010d dddd 0000	
Description	Performs a one's complement of register Rd	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	com r4 ; take one's complement of r4 ; i.e., if Rd had $0F, then r4 = $F0	

CP	**Compare**	**Comments**
Syntax	CP Rd, Rr	Compare Rr to Rd
Operands	0≤d≤31, 0≤r≤31	(d, r) working registers 0 to 31
Operation	Rd-Rr	Compare Rr to Rd
Flags Affected	H, S, V, N, Z, C	In SREG
Encoding	0001 01rd dddd rrrr	
Description	Performs a compare between two registers Rd and Rr. None of the registers are changed. All conditional branches can be used after this instruction.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	cp r4, r19 ; compare r4 with r19	

CPC	**Compare with Carry**	**Comments**
Syntax	CPC Rd, Rr	Compare Rd and Rr including C flag
Operands	0≤d≤31, 0≤r≤31	(d, r) working register 0 to 31
Operation	Rd-Rr-C	Compare Rd and Rr and C together
Flags Affected	H, S, V, N, Z, C	In SREG
Encoding	0000 01rd dddd rrrr	
Description	Performs a compare between two registers Rd and Rr and also takes into account the previous carry. None of the registers are changed. All conditional branches can be used after this instruction	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	; compare r3: r2 with r1:r0 cp r2, r0 ; compare low byte cp r3, r1 ; compare high byte	

CPI	**Compare with Immediate**	**Comments**
Syntax	CPI Rd, k	Rd working register and k is constant
Operands	16≤d≤31, 0≤k≤255	(d) working registers R16 to R31 and (k) from 0 to 255 or $00 to $FF
Operation	Rd-k	Compares immediately k with the value in Rd
Flags Affected	H, S, V, N, Z, C	In SREG
Encoding	0011 kkkk dddd kkkk	
Description	Performs a compare between register Rd and a constant. The register is not changed. All conditional branches can be used after this instruction	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	cpi r19, 3 ; compare r19 with 3	

CPSE	**Compare Skip if Equal**	**Comments**
Syntax	CPSE Rd, Rr	Compare Rd with Rr and skip if equal
Operands	0≤d≤31, 0≤r≤31	(d, r) working registers 0 to 31
Operation	Skips the next instruction if Rd🡐Rr	Skip next instruction if Rd=Rr
Flags Affected	None	In SREG
Encoding	0001 00rd dddd rrrr	
Description	Performs a compare between two register Rd and Rr, and skips the next instruction if Rd🡐Rr	If Rd=Rr then skip next instruction, else continue.
Words	1	16 bit instruction
Cycles	1	1 machine cycle

Example	`cpse r4, r0 ; compare r4 to r0`
	`neg r4 ; only execute if r4<>r0`
	`nop ; continue (do nothing)`	

DEC	**Decrement**	**Comments**
Syntax	DEC Rd	Rd is a working register
Operands	0≤d≤31	(d) working registers 0 to 31
Operation	Rd🡐Rd-1	Decrease Rd by 1and store in Rd
Flags Affected	S, V, N, Z	In SREG
Encoding	1001 010d dddd 1010	
Description	Subtracts 1 from the contents of register Rd and places the result in the destination register Rd. The C flag in SREG is not affected by the operation, thus allowing the DEC instruction to be used on a loop counter in multiple precision computations. When operating on unsigned values, only BREQ and BRNE branches can be expected to perform consistently. When operating on two's complement values, all signed branches are available.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle

Example	`ldi r17, $10 ; load immediately $10 into r17`
	`loop: add r1, r2 ; add r2 to r1`
	`dec r17 ; decrement r17 by 1`
	`brne loop ; branch if r17<>0`
	`nop ; continue (do nothing)`	

EOR	**Exclusive OR**	**Comments**
Syntax	EOR Rd, Rr	Rd and Rr working registers
Operands	0≤d≤31, 0≤r≤31	(d, r) working registers 0 to 31
Operation	Rd←Rd⊕Rr	Exclusive OR between Rd and Rr
Flags Affected	S, V, N, Z	In SREG
Encoding	0010 01rd dddd rrrr	
Description	Performs logical EOR between the contents of register Rd and register Rr and places the result in the destination register Rd.	Note: In digital logic terms XOR and EOR are the same function.
Words	1	16 bit instruction
Cycles	1	1 machine cycle

Example	`eor r4, r4 ; clears r4`
	`eor r0, r22 ; bitwise exclusive OR between r0`
	`; and r22`	

ICALL	**Indirect Call to Subroutine**	**Comments**
Syntax	ICALL	
Operands	None	
Operation	PC(15,0)←Z(15-0)	Program Counter (PC) and Z-pointer register (r30 and r31) (ZL, ZH)
Flags Affected	None	In SREG
Encoding	1001 0101 XXXX 1001	X's are don't care
Description	Indirect call of a subroutine pointed to by the Z (16 bits) pointer register in the register file. The Z pointer register is 16 bits wide and allows call to a subroutine within the current 64k words (128k bytes) section in the program memory space.	Note: X, Y, and Z registers have a low and high byte and are located at the bottom of the working registers (r26-r31) and are called pointer registers for indirect addressing. They can also be used for 16-bit math/logic operations.
Words	1	16 bit instruction
Cycles	3	3 machine cycles

Example	`mov r30, r0 ; set offset to call table`
	`icall ; call routine pointed to by r31:r30`	

IJMP	**Indirect Jump**	**Comments**
Syntax	IJMP	
Operands	None	
Operation	PC(15-0)← Z(15-0)	Program Counter (PC) and Z-pointer register (r30 and r31) (ZL, ZH)
Flags Affected	None	In SREG
Encoding	1001 0100 XXXX 1001	X's are don't cares
Description	Indirect jump to the address pointed to by the Z (16 bits) pointer register file. The Z pointer register is 16 bits wide and allows jump within the current 64k words (128k bytes) section of program memory.	Note: X, Y, and Z registers have a low and high byte and are located at the bottom of the working registers (r26-r31) and are called pointer registers for indirect addressing. They can also be used for 16-bit math/logic operations.
Words	1	16 bit instruction
Cycles	2	2 machine cycles
Example	mov r30, r0 ; set offset to jump table ijmp ; jump to routine pointed to by r31:r30	Note: be careful not to type into assembler software IJUMP (does not work).

IN	**Load an I/O Port to Register**	**Comments**
Syntax	In Rd, P	
Operands	0≤d≤31, 0≤P≤63	(d) working registers 0 to 31, (P) I/0 Ports (registers) 0 to 63
Operation	Rd←P	From P (Port) to Rd working register 8-bits
Flags Affected	None	In SREG
Encoding	1011 0PPd dddd PPPP	
Description	Loads data from the I/O Space (ports, timers, configuration registers, etc.) into register Rd in the register file.	8-bits wide
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	in r25, $16 ; Read PortB at address $16 ; value stored in r25	

INC	**Increment**	**Comments**
Syntax	INC Rd	Rd is working register
Operands	0≤d≤31	
Operation	Rd🡐Rd+1	Increment Rd by 1 store in Rd
Flags Affected	S, V, N, Z	In SREG
Encoding	1001 010d dddd 0011	
Description	Adds 1 to the contents of register Rd and places the results in the destination register Rd. The C flag in SREG is not affected by the operation, thus allowing the INC instruction to be used on a loop counter in multiple precision computations. When operating on unsigned numbers, only BREQ and BRNE branches can be expected to perform consistently. When operating on two's complement values, all signed branches are available.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	clr r22 ; clear r22 to $00 Loop: inc r22 ; increment r22 by 1 …. cpi r22, $4F ; compare r22 to $4F brne loop ; if not equal branch to loop nop ; continue (do nothing)	Note: be careful not to type into assembler software IJUMP (does not work).

JMP	**Jump**	**Comments**
Syntax	JMP k	K is constant
Operands	0≤k≤4M	K between 0 and 4M (words) in program memory – see individual AVR device for range.
Operation	PC🡐 k	Constant (k) into Program Counter (PC)
Flags Affected	None	In SREG
Encoding	1001 010k kkkk 110k kkkk kkkk kkkk kkkk	
Description	Jump to an address within the entire 4M (words) program memory. See also RJMP	
Words	2	32 bit instruction
Cycles	3	3 machine cycles
Example	mov r1, r0 ; copy r0 to r1 jmp farplc ; unconditional jump …. farplc: nop ; jump destination (do nothing)	Note: be careful not to type into assembler software JUMP (does not work). Some assemblers are case-sensitive on labels. Also check for restricted names and symbols that cannot be used in the label.

LD	**Load Indirect from SRAM to Register Using Index X**	**Comments**
Syntax	LD Rd, X LD Rd, X+ LD Rd, -X	X: Unchanged X: Post-incremented X: Pre-incremented
Operands	0≤d≤31	(d) working registers 0 to 31
Operation	X Unchanged Rd🡐 (X) X Post-incremented Rd🡐(X) X🡐X+1 X Pre-decremented X🡐X-1 Rd🡐(X)	X into Rd X into Rd and then X + 1 into X X-1 into X and then X into Rd
Flags Affected	None	In SREG
Encoding	X unchanged 1001 000d dddd 1100 X Post-incremented 1001 000d dddd 1101 X Pre-incremented 1001 000d dddd 1110	
Description	Loads 1 byte indirect from SRAM to register. The SRAM location is pointed to by the X (16 bits) pointer register in the register file. Memory access is limited to the current SRAM page of 64k bytes. To access another SRAM page, the RAMPX in register in the I/O area has to be changed. The X pointer register can be left unchanged after the operation, or it can be incremented or decremented. These features are especially suited for accessing arrays, tables, and stack pointer usage of the X pointer register.	Note: X, Y, and Z registers have a low and high byte and are located at the bottom of the working registers (r26-r31) and are called pointer registers for indirect addressing. They can also be used for 16-bit math/logic operations.
Words	1	16 bit instruction
Cycles	2	2 machine cycles
Example	clr r27 ; clear X high byte (XH) ldi r26, $20 ; set X low byte to $20 ld r0, X+ ; load r0 with SRAM location $20 ; (X post inc) ld r1, X ; load r1 with SRAM location $21 ldi r26, $23 ; set X low byte to $23 ld r2, X ; load r2 with SRAM loc. $23 ld r3, -X ; load r3 with SRAM loc. $22 ; (X pre dec)	

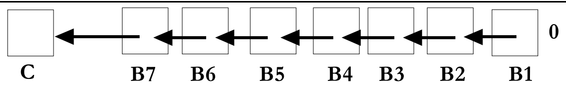
LD (LDD)	**Load Indirect from SRAM to Register Using Index Y**	**Comments**
Syntax	LD Rd, Y LD Rd, Y+ LD Rd, -Y LDD Rd, Y+q	Y: Unchanged Y: Post-incremented Y: Pre-incremented Y: Unchanged, q: Displacement
Operands	0≤d≤31, 0≤q≤63	(d) working registers 0 to 31 and q is displacement value 0 to 63
Operation	Y Unchanged Rd← (Y) Y Post-incremented Rd←(Y) Y←Y+1 Y Pre-decremented Y←Y-1 Rd←(Y) Y Unchanged, q displacement Rd← (Y+q)	Y into Rd Y into Rd and then Y + 1 into Y Y-1 into Y and then Y into Rd Y+q into Rd
Flags Affected	None	In SREG
Encoding	Y unchanged 1000 000d dddd 1000 Y Post-incremented 1001 000d dddd 1001 Y Pre-incremented 1001 000d dddd 1010 Y Unchanged, q displace. 10q0 qq0d dddd 1qqq	
Description	Loads 1 byte indirect with or without displacement from SRAM to register. The SRAM location is pointed to by the Y (16 bits) pointer register in the register file. Memory access is limited to the current SRAM page of 64k bytes. To access another SRAM page, the RAMPY in register in the I/O area has to be changed. The Y pointer register can be left unchanged after the operation, or it can be incremented or decremented. These features are especially suited for accessing arrays, tables, and stack pointer usage of the Y pointer register.	Note: X, Y, and Z registers have a low and high byte and are located at the bottom of the working registers (r26-r31) and are called pointer registers for indirect addressing. They can also be used for 16-bit math/logic operations. This registers are also called Index registers.
Words	1	16 bit instruction
Cycles	2	2 machine cycles
Example	clr r29 ; clear Y high byte (YH) ldi r28, $20 ; set Y low byte to $20 ld r0, Y+ ; load r0 with SRAM location $20 ; (Y post inc) ld r1, Y ; load r1 with SRAM location $21 ldi r28, $23 ; set Y low byte to $23 ld r2, Y ; load r2 with SRAM loc. $23 ld r3, -Y ; load r3 with SRAM loc. $22 ; (Y pre dec) ldd r4, Y+2 ; Load r4 with SRAM loc. $24	

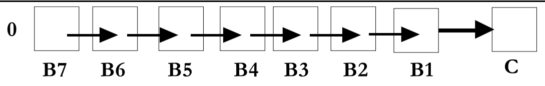
LD (LDD)	**Load Indirect from SRAM to Register Using Index Z**	**Comments**
Syntax	LD Rd, Z LD Rd, Z+ LD Rd, -Z LDD Rd, Z+q	Z: Unchanged Z: Post-incremented Z: Pre-incremented Z: Unchanged, q: Displacement
Operands	0≤d≤31, 0≤q≤63	(d) working registers 0 to 31 and q is displacement value 0 to 63
Operation	Z Unchanged Rd🡐 (Z) Z Post-incremented Rd🡐(Z) Z🡐Z+1 Z Pre-decremented Z🡐Z-1 Rd🡐(Z) Z Unchanged, q displacement Rd🡐 (Z+q)	Z into Rd Z into Rd and then Z + 1 into Z Z-1 into Z and then Z into Rd Z+q into Rd
Flags Affected	None	In SREG
Encoding	Z unchanged 1000 000d dddd 0000 Z Post-incremented 1001 000d dddd 0001 Z Pre-incremented 1001 000d dddd 0010 Z Unchanged, q displace. 10q0 qq0d dddd 0qqq	
Description	Loads 1 byte indirect with or without displacement from SRAM to register. The SRAM location is pointed to by the Z (16 bits) pointer register in the register file. Memory access is limited to the current SRAM page of 64k bytes. To access another SRAM page, the RAMPZ in register in the I/O area has to be changed. The Z pointer register can be left unchanged after the operation, or it can be incremented or decremented. These features are especially suited for stack pointer usage of the Z pointer register; however, because the Z pointer register can be used for indirect subroutine calls, indirect jumps, and table lookup, it is often more convenient to use the X and Y pointer as a dedicated stack pointer. For using the Z pointer for table lookup in program memory, see the LPM instruction.	Note: X, Y, and Z registers have a low and high byte and are located at the bottom of the working registers (r26-r31) and are called pointer registers for indirect addressing. They can also be used for 16-bit math/logic operations. This registers are also called Index registers.
Words	1	16 bit instruction
Cycles	2	2 machine cycles
Example	clr r31 ; clear Z high byte (ZH) ldi r30, $20 ; set Z low byte to $20 ld r0, Z+ ; load r0 with SRAM location $20 ; (Z post inc) ld r1, Z ; load r1 with SRAM location $21 ldi r30, $23 ; set Z low byte to $23 ld r2, Z ; load r2 with SRAM loc. $23 ld r3, -Z ; load r3 with SRAM loc. $22 ; (Z pre dec) ldd r4, Z+2 ; Load r4 with SRAM loc. $24	

LDI	**Load Immediate**	**Comments**
Syntax	LDI Rd, k	Rd is working register, k is constant
Operands	16≤d≤31, 0≤k≤255	(d) working registers 16 to 31 and k is a value between 0 and 255 or $00 and $FF
Operation	Rd←k	Load immediately the value of k into Rd
Flags Affected	None	In SREG
Encoding	1110 kkkk dddd kkkk	
Description	Loads an 8-bit constant directly to registers 16 to 31	Note: command does not work with registers r0 through r15. A typical command to use is the mov r0, r16 to move the contents of r16 into r0 if the lower working registers will be used for temporary storage.
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	clr r31 ; clear Z high byte (ZH) ldi r30, $F0 ; set Z low byte (ZL) to $F0 lpm ; load constant from program ; memory pointed to by Z	

LDS	**Load Direct from SRAM**	**Comments**
Syntax	LDS, Rd, k	Rd is working register, k is constant
Operands	0≤d≤31, 0≤k≤65535	(d) working registers 0 to 31 and k is a value between 0 and 65535 or $00 and $FFFF
Operation	Rd←(k)	Load immediately from SRAM into Rd
Flags Affected	None	In SREG
Encoding	1001 000d dddd 0000 kkkk kkkk kkkk kkkk	
Description	Loads 1 byte from SRAM to a working register. A 16-bit address must be supplied. Memory access is limited to the current SRAM page of 64k bytes. The LDS instruction uses the RAMPZ register to access memory above 64k bytes.	
Words	2	32 bit instruction
Cycles	4	4 machine cycles
Example	lds r2, $FF00 ; load r2 with the contents of ; SRAM location $FF00 add r2, r1 ; add r1 to r2 sts $FF00, r2 ; Write back the new value into ; SRAM location $FF00	

LPM	**Load Program Memory**	**Comments**
Syntax	LPM	
Operands	None	
Operation	R0← Z	Load contents pointed to by Z into r0
Flags Affected	None	In SREG
Encoding	1001 0101 110X 1000	
Description	Z points to program memory and the contents of memory location are loaded into R0	
Words	1	16 bit instruction
Cycles	3	3 machine cycles
Example	clr r31 ; clear Z high byte (ZH) ldi r30, $F0 ; load constant from program lpm ; memory pointed to by Z (r31:r30) ; loc. $00$F0	

LSL	**Logical Shift Left**	**Comments**
Syntax	LSL Rd	Rd is working register
Operands	0≤d≤31	(d) working registers 0 to 31
Operation		Shifts all bits in Rd one, including C bit
Flags Affected	H, S, V, N, C, Z	In SREG
Encoding	0000 11dd dddd dddd	
Description	Shifts all bits in Rd one place to the left. Bit 0 is cleared. Bit 7 is loaded into C flag of the SREG. This operation effectively multiplies an unsigned value by 2.	Note: This will clear Rd after 8 shifts (0-7).
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	lsl r0 ; multiply r0 by 2	

LSR	**Logical Shift Right**	**Comments**
Syntax	LSR Rd	Rd is working register
Operands	0≤d≤31	(d) working registers 0 to 31
Operation		Shifts all bits in Rd one, including C bit
Flags Affected	S, V, N, Z, C	In SREG
Encoding	1001 010d dddd 0110	
Description	Shifts all bits in Rd one place to the right. Bit 7 is cleared. Bit 0 is loaded into C flag of the SREG. This operation effectively divides an unsigned value by 2. The C flag can be used to round the result.	Note: This will clear Rd after 8 shifts (0-7).
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	lsr r0 ; divide r0 by 2	

MOV	**Copy Register**	**Comments**
Syntax	MOV Rd, Rr	Rd and Rr are working registers
Operands	0≤d≤31, 0≤r≤31	(d,r) working registers 0 to 31
Operation	Rd🡐 Rr	Copy value in Rr to Rd; Rr unchanged
Flags Affected	None	In SREG
Encoding	0010 11rd dddd rrrr	
Description	This instruction makes a copy of one register into another. Th source register Rr is left unchanged; the destination register Rd is loaded with the copy of Rr.	Note: Effective command to move values into working registers R0 through R15.
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	mov r16, r0 ; copy r0 to r16	

MUL	**Multiply**	**Comments**
Syntax	MUL Rd, Rr	Rd and Rr are working register
Operands	0≤d≤31, 0≤r≤31	(d, r) working registers 0 to 31
Operation	R1,R0🡐 Rr*Rd	Rd*Rr, results in R1:R0
Flags Affected	C	In SREG
Encoding	1001 11rd dddd rrrr	
Description	The multiplicand Rr and the multiplier Rd are two registers. The 16-bit product is placed in R1 (high byte) and R0 (low byte)	Note: that if the multiplicand and the multiplier are selected from R0 or R1, the result will overwrite those after multiplication.
Words	1	16 bit instruction
Cycles	2	2 machine cycles
Example	…..mul r6, r5 ; multiply r6 and r5 mov r6, r1 ; copy result back in r6:r5 mov r5, r0 ; copy result back into r6:r5	

NEG	**Two's Complement**	**Comments**
Syntax	Neg Rd	Rd is working register
Operands	0≤d≤31	(d) working registers 0 to 31
Operation	Rd 🡐 $00-Rd	Subtracts Rd from $00
Flags Affected	H, S, V, N, Z, C	In SREG
Encoding	1001 010d dddd 0001	
Description	Replaces the contents of register Rd with its two's complement; the value $80 is left unchanged	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	sub r11, r0 ; subtract r0 from r11 brpl positive ; branch if result positive neg r11 ; take two's complement ; of r11 positive: nop ; branch destination	

NOP	**No Operation**	**Comments**
Syntax	NOP	
Operands	none	
Operation	No	
Flags Affected	None	In SREG
Encoding	0000 0000 0000 0000	
Description	Performs a single-cycle No Operation	Note: used to construct delay loop timing if nested loops are constructed
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	clr r16 ; clear r16 r16=$00 ser r17 ; set r17 r17=$FF out $18, r16 ; write zeros to PortB nop ; wait and do nothing for 1 cycle out $18, r17 ; write ones to PortB	

OR	**Logical OR**	**Comments**
Syntax	OR Rd, Rr	Rd and Rr are working registers
Operands	0≤d≤31, 0≤r≤31	(d,r) working registers 0 to 31
Operation	Rd ← Rd OR Rr	Logical OR between Rr and Rd results Rd
Flags Affected	S, V, N, Z	In SREG
Encoding	0010 10rd dddd rrrr	
Description	Performs the logical OR between the contents of register Rd and those of register Rr, and places the result in the destination register Rd.	Note: This is a bitwise operation.
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	or r15, r16 ; do bitwise OR between registers	

ORI	**Logical OR with Immediate**	**Comments**
Syntax	OR Rd, k	Rd working register, k is constant
Operands	16≤d≤31, 0≤k≤255	(d) working registers16 to 31, k 0 to 255 ($00 to $FF)
Operation	Rd ← Rd OR k	Logical OR between k and Rd results Rd
Flags Affected	S, V, N, Z	In SREG
Encoding	0110 kkkk dddd kkkk	
Description	Performs the logical OR between the contents of register Rd and a constant, and places the result in the destination register Rd	Note: This is a bitwise operation.
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	ori r16, $F0 ; set high nibble of r16 ori r17, 1 ; set bit 0 of r17	

OUT	**Store Register to I/O Port**	**Comments**
Syntax	OUT P, Rr	P is a Port and Rr is a working register
Operands	0≤r≤31, 0≤P≤63	(r) working registers 0 to 31, (P) port (I/O register) 0 to 63.
Operation	P ← Rr	Value in source register Rr out to I/O register (Port)
Flags Affected	None	In SREG
Encoding	1011 1PPr rrrr PPPP	
Description	Stores data from register Rr (source register) in the register file to I/O space (ports, timers, configuration registers, etc.).	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	clr r16 ; clear r16 r16=$00 ser r17 ; set r17 r17=$FF out $18, r16 ; write zeros to PortB nop ; wait and do nothing for 1 cycle out $18, r17 ; write ones to PortB	

POP	**Pop Register from Stack**	**Comments**
Syntax	POP Rd	Rd is a working register
Operands	0≤d≤31	(d) working registers 0 to 31
Operation	Rd ← STACK	Value on the STACK into Rd
Flags Affected	None	In SREG
Encoding	1001 000d dddd 1111	
Description	Loads register Rd with a byte from the STACK	Note: To use the STACK it must be initialized in program memory before using. The address of the STACK is identified by the STACK POINTER Register (SPH, SPL) a 16-bit special register in the AVR core.
Words	1	16 bit instruction
Cycles	2	2 machine cycles
Example	call routine ; call subroutine routine: push r14 ; save r14 onto the stack push r13 ; save r13 onto the stack …. pop r13 ; restore r13 pop r14 ; restore r14 ret ; return to main program	Note: The order of the push and pop commands. Because of how the STACK works, you must pop in reverse of how you pushed the values unto the STACK. Plus, for every push executed, there is an equal amount of pop executed to keep the STACK balanced. This is done because the first values placed on the STACK when executing a subroutine or interrupt is the Program Counter.

PUSH	**Push Register on Stack**	**Comments**
Syntax	PSH Rr	Rr is a working register
Operands	0≤r≤31	(r) working registers 0 to 31
Operation	STACK 🡐 Rr	Value in Rr onto the STACK
Flags Affected	None	In SREG
Encoding	1001 001r rrrr 1111	
Description	Loads register Rd with a byte from the STACK	Note: To use the STACK it must be initialized in program memory before using. The address of the STACK is identified by the STACK POINTER Register (SPH, SPL) a 16-bit special register in the AVR core.
Words	1	16 bit instruction
Cycles	2	2 machine cycles

Example	
	```
	          call routine          ; call subroutine
	routine:  push r14        ; save r14 onto the stack
	          push r13        ; save r13 onto the stack
	….
	          pop r13         ; restore r13
	          pop r14         ; restore r14
	          ret             ; return to main program
	```
		Note: The order of the push and pop commands. Because of how the STACK works, you must pop in reverse of how you pushed the values unto the STACK. Plus, for every push executed, there is an equal amount of pop executed to keep the STACK balanced. This is done because the first values placed on the STACK when executing a subroutine or interrupt is the Program Counter.

RCALL	**Relative Call to Subroutine**	**Comments**
Syntax	RCALL, k	K is constant
Operands	-2k≤k≤2k	±2k from the rcall command
Operation	PC 🡐 PC +k + 1	Program Counter (PC) + k + 1 into PC
Flags Affected	None	In SREG
Encoding	1101 kkkk kkkk kkkk	
Description	Calls a subroutine within ±2k words (4k bytes). The return address (the instruction after the RCALL) is stored onto the stack. (See also CALL).	
Words	1	16 bit instruction
Cycles	3	3 machine cycles

Example	
	```
	          rcall routine          ; call subroutine
	…..
	routine:  push r14        ; save r14 onto the stack
	….
	          pop r14         ; restore r14
	          ret             ; return to main program
	```
		Note: use labels that make sense when calling or jumping to a subroutine. The return command (ret) pulls the "old PC" off the STACK and places it into PC so that the program can return from the subroutine. It is crucial to pop all values off the stack before the return.

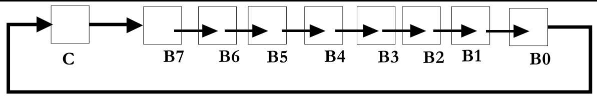
RET	**Return from Subroutine**	**Comments**
Syntax	RET	
Operands	None	
Operation	PC← STACK	Value on the STACK into PC
Flags Affected	None	In SREG
Encoding	1001 0101 0XX0 1000	X's are don't cares
Description	Returns from subroutine. The return address is loaded from the STACK.	Note: To use the STACK it must be initialized in program memory before using. The address of the STACK is identified by the STACK POINTER Register (SPH, SPL) a 16-bit special register in the AVR core.
Words	1	16 bit instruction
Cycles	4	4 machine cycles

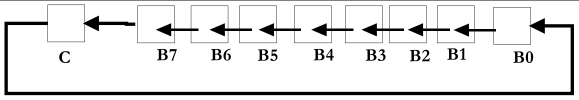
Example	`call routine ; call subroutine`
	`routine: push r14 ; save r14 onto the stack`
	`push r13 ; save r13 onto the stack`
	`….`
	`pop r13 ; restore r13`
	`pop r14 ; restore r14`
	`ret ; return to main program`
		Note: The order of the push and pop commands. Because of how the STACK works, you must pop in reverse of how you pushed the values unto the STACK. Plus, for every push executed, there is an equal amount of pop executed to keep the STACK balanced. The return command (ret) pulls the "old PC" off the STACK and places it into PC so that the program can return from the subroutine. It is crucial to pop all values off the stack before the return.

RETI	**Return from Interrupt**	**Comments**
Syntax	RETI	
Operands	None	
Operation	PC← STACK	Value on the STACK into PC
Flags Affected	I	In SREG
Encoding	1001 0101 0XX1 1000	X's are don't cares
Description	Returns from interrupt. The return address is loaded from the STACK and the global interrupt flag is set.	Note: To use the STACK it must be initialized in program memory before using. The address of the STACK is identified by the STACK POINTER Register (SPH, SPL) a 16-bit special register in the AVR core.
Words	1	16 bit instruction
Cycles	4	4 machine cycles

Example	
	```
	          …..
	extint:   push r0    ; save r0 on the stack
	           ……
	          pop r0     ; restore r0
	          reti       ; return and enable interrupts
	```
		Note: The order of the push and pop commands. Because of how the STACK works, you must pop in reverse of how you pushed the values unto the STACK. Plus, for every push executed, there is an equal amount of pop executed to keep the STACK balanced. The return command (ret) pulls the "old PC" off the STACK and places it into PC so that the program can return from the subroutine. It is crucial to pop all values off the stack before the return.

RJMP	**Relative Jump**	**Comments**
Syntax	RJMP k	K is a constant
Operands	-2k≤k≤2k	±2k relative to jump instruction
Operation	PC🡐 PC + k + 1	Program Counter (PC) + K + 1 into PC
Flags Affected	None	In SREG
Encoding	1100 kkkk kkkk kkkk	
Description	Relative jump to an address within PC-2k and PC+2k (words). In the assembler, labels are used instead of relative operands. For AVR microcontrollers with program memory not exceeding 4k works (8k bytes) this instruction can address the entire memory from every address location.	Note: See individual specification sheets for AVR devices to determine memory map sizes.
Words	1	16 bit instruction
Cycles	2	2 machine cycles
Example	cpi r16, $42 ; compare r16 to $42 brne error ; branch if r16<>$42 rjmp ok ; unconditional branch error: add r16, r17 ; add r17 to r16 inc r16 ; increment r16 by 1 ok: nop ; destination for rjmp ; (do nothing)	Note: Some assemblers are case-sensitive on labels. Also check for restricted names and symbols that cannot be used in the label.

ROL	**Rotate Left Through Carry**	**Comments**
Syntax	ROL Rd	Rd is working register
Operands	0≤d≤31	(d) working registers 0 to 31
Operation		Rotates through C bit left one place
Flags Affected	H, S, V, N, C, Z	In SREG
Encoding	0001 11dd dddd dddd	
Description	Rotates all bits in Rd one place to the left. The C flag is shifted into bit 0 of Rd. Bit 7 is shifted into the C flag. Recirculates bits.	Note: Rotation is different then Shift because the bits are circulating back through the Carry bit.
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	rol r15 ; rotate left	

ROR	**Rotate Right Through Carry**	**Comments**
Syntax	ROR Rd	Rd is working register
Operands	0≤d≤31	(d) working registers 0 to 31
Operation		Rotates through C bit right one place
Flags Affected	S, V, N, Z, C	In SREG
Encoding	1001 010d dddd 0111	
Description	Rotates all bits in Rd one place to the right. The C flag is shifted into bit 7 of Rd. Bit 0 is shifted into the C flag. Recirculates bits.	Note: Rotation is different then Shift because the bits are circulating back through the Carry bit.
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	ror r15 ; rotate right	

SBC	**Subtract with Carry**	**Comments**
Syntax	SBC Rd, Rr	Rd and Rr are working registers
Operands	0≤d≤31, 0≤r≤31	(d,r) working registers 0 to 31
Operation	Rd←Rd-Rr-C	Subtract Rd-Rr-C, result in Rd
Flags Affected	H, S, V, N, C, Z	In SREG
Encoding	0000 10rd dddd rrrr	
Description	Subtracts two registers and subtracts with the C flag and places the result in the destination register Rd	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	;Subtract r1:r0 from r3:r2 sub r2,r0 ; subtract low byte sbc r3, r1 ; subtract with carry high byte ; result in r3:r2	

SBCI	**Subtract Immediate with Carry**	**Comments**
Syntax	SBCI Rd, K	Rd is a working register, k is constant
Operands	16≤d≤31, 0≤k≤255	(d) working register 16 to 31, k 0 to 255
Operation	Rd←Rd-k-C	Subtract Rd-k-C, result in Rd
Flags Affected	H, S, V, N, C, Z	In SREG
Encoding	0100 kkkk dddd kkkk	
Description	Subtracts a constant from a register and subtracts with the C flag and places the result in the destination register Rd	Note: Only works with R16 through R31
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	;Subtract $4F23 from r17:r16 subi r16,$23 ; subtract low byte sbci r17, $4F ; subtract with carry high byte ; result in r17:r16	

SBI	**Set Bit in I/O Register**	**Comments**
Syntax	SBI P, b	Clear bit location (b) at I/O register (P)
Operands	0≤P≤31, 0≤b≤7	(P) I/O registers and (b) bits 0 to 7
Operation	I/O (P, b)← 1	Sets the bit location at the I/O register location
Flags Affected	None	In SREG
Encoding	1001 1010 PPPP Pbbb	
Description	Sets a specified bit in an I/O register. This instruction operates on the lower 32 I/O registers 0-31. Addresses 0-31.	
Words	1	16 bit instruction
Cycles	2	2 machine cycles
Example	out $1E, r0 ; write EEPROM address sbi $1C, 0 ; set read bit in EECR in r1, $1D ; read EEPROM data	

SBIC	**Skip if Bit in I/O Register is Cleared**	**Comments**
Syntax	SBIC P, b	(P) Port and b is a bit location on I/O register.
Operands	0≤P≤31, 0≤b≤7	(P) I/O registers and (b) bits 0 to 7
Operation	Skip next instruction if I/O(P, b) = 0	Check for zero on pin at Port and skip next instruction if true, else do next instruction.
Flags Affected	None	In SREG
Encoding	1001 1001 PPPP Pbbb	
Description	Tests a single bit in an I/O register and skips the next instruction if the bit is cleared. This instruction operates on the lower 32 I/O registers – addresses 0-31	Note: These I/O registers are the Port, timer, configuration registers, etc.
Words	1	16 bit instruction
Cycles	2 (false); 3 (true)	2 or 3 machine cycles based on decision
Example	e2wait: sbic $1C, 1 ; skip next inst. If EEWE ; cleared rjmp e2wait ; EEPROM write not finished nop ; continue (do nothing)	

SBIS	**Skip if Bit in I/O Register is Set**	**Comments**
Syntax	SBIS P, b	(P) Port and b is a bit location on Port
Operands	0≤P≤31, 0≤b≤7	(P) I/O registers and (b) bits 0 to 7
Operation	Skip next instruction if I/O(P, b) = 1	Check for one on pin at Port and skip next instruction if true, else do next instruction.
Flags Affected	None	In SREG
Encoding	1001 1011 PPPP Pbbb	
Description	Tests a single bit in an I/O register and skips the next instruction if the bit is set. This instruction operates on the lower 32 I/O registers – addresses 0-31	Note: These I/O registers are the Port, timer, configuration registers, etc.
Words	1	16 bit instruction
Cycles	2 (false); 3 (true)	2 or 3 machine cycles based on decision
Example	waitset: sbis $10, 0 ; skip next inst. If bit 0 ; in PortD set rjmp waitset ; bit not set nop ; continue (do nothing)	

SBIW	**Subtract Immediate from Word**	**Comments**
Syntax	SBIW Rdl, k	Rdl lower byte working registers for X, Y, Z and R24; k is constant
Operands	dl ∈{24,26,28,30} ≤31, 0≤k≤63	(dl) working register 24, 26, 28, 30, k 0 to 255.
Operation	Rdh:Rdl← Rdh:Rdl-k	Subtract Rdh:Rdl-k, result in Rdh:Rdl
Flags Affected	S, V, N, Z, C	In SREG
Encoding	1001 0111 kkdd kkkk	
Description	Subtracts an immediate value (0-63) from a register pair and places the result in the register pair. This instruction operates on the upper four register pairs and is well suited for operations on the pointer registers.	Note: dl∈{24,26,28,30} Values stored in R25:R24; R27:R26; R29:R28; R31:R30. The last three pairs are the X, Y, Z pointer or index registers.
Words	1	16 bit instruction
Cycles	2	2 machine cycles
Example	sbiw r24, 1 ; subtract 1 from r25:r24 sbiw r28, 63 ; subtract 63 from the Y pointer ; (r29:r28)	

SBR	**Set Bits in Register**	**Comments**
Syntax	SBR Rd, k	Rd working registers, k is constant
Operands	16≤d≤31, 0≤k≤255	(d) working registers 16 to 31, k 0 to 255
Operation	Rd← Rd OR k	Rd is ORed with k result in Rd
Flags Affected	S, V, N, Z	In SREG
Encoding	0110 kkkk dddd kkkkk	
Description	Sets specified bits in register Rd. Performs the logical ORI between the contents of register Rd and a constant mask k and places the result in the destination register Rd.	Note: Similar to Rd←Rd OR k Example: ori r16, $F0 ; sets high nibble
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	sbr r16, 3 ; sets bit 0 and 1 in r16 sbr r17, $F0 ; set 4 MSB in R17	

SBRC	**Skip if Bit in Register is Cleared**	**Comments**
Syntax	SBRC Rr, b	Rr working registers, b is bit
Operands	0≤d≤31, 0≤b≤7	(r) working registers 0 to 31, b 0 to 7
Operation	Skip next instruction if Rr(b) = 0	Rr bit location check for zero; true skip next instruction; else continue
Flags Affected	None	In SREG
Encoding	1111 110r rrrr Xbbb	X is a don't care
Description	Tests a single bit in a register and skips the next instruction if the bit is cleared	Note: Similar to the SBIC command for checking an I/O register pin location.
Words	1	16 bit instruction
Cycles	1 (false); 2 (true)	1 or 2 machine cycles based on decision
Example	sub r0, r1 ; subtract r1 from r0 sbrc r0, 7 ; skip if bit 7 in r0 cleared sub r0, r1 ; only execute if bit 7 in r0 not ; cleared nop ; continue (do nothing)	

SBRS	**Skip if Bit in Register is Set**	**Comments**
Syntax	SBRS Rr, b	Rr working registers, b is bit
Operands	0≤d≤31, 0≤b≤7	(r) working registers 0 to 31, b 0 to 7
Operation	Skip next instruction if Rr(b) = 1	Rr bit location check for one; true skip next instruction; else continue
Flags Affected	None	In SREG
Encoding	1111 111r rrrr Xbbb	X is a don't care
Description	Tests a single bit in a register and skips the next instruction if the bit is set	Note: Similar to the SBIS command for checking an I/O register pin location.
Words	1	16 bit instruction
Cycles	1 (false); 2 (true)	1 or 2 machine cycles based on decision
Example	sub r0, r1 ; subtract r1 from r0 sbrs r0, 7 ; skip if bit 7 in r0 set neg r0 ; only execute if bit 7 in r0 not ; set nop ; continue (do nothing)	

SEC	**Set Carry Flag**	**Comments**
Syntax	SEC	
Operands	None	
Operation	C← 1	Set Carry (C) flag bit
Flags Affected	C	In SREG
Encoding	1001 0100 0000 1000	
Description	Sets the Carry Flag (C) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	sec ; set carry flag adc r0, r1 ; r0 = r0 + r1 + 1	

SEH	**Set Half Carry Flag**	**Comments**
Syntax	SHE	
Operands	None	
Operation	H← 1	Set Half Carry (H) flag bit
Flags Affected	H	In SREG
Encoding	1001 0100 0101 1000	
Description	Sets the Half Carry Flag (H) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	seh ; set half carry flag	

SEI	**Set Global Interrupt Flag**	**Comments**
Syntax	SEI	
Operands	None	
Operation	I← 1	Set Global Interrupt (I) flag bit
Flags Affected	I	In SREG
Encoding	1001 0100 0111 1000	
Description	Sets the Global Interrupt Flag (I) in SREG (status register)	Note: There are local registers used for interrupts
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	cli ; disable interrupts I=0 in r13, $16 ; Read PortB sei ; enable interrupts I=1	

SEN	**Set Negative Flag**	**Comments**
Syntax	SEN	
Operands	None	
Operation	N← 1	Set Negative (N) flag bit
Flags Affected	N	In SREG
Encoding	1001 0100 0110 1000	
Description	Sets the Negative Flag (N) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	sen ; set negative flag	

SER	**Set All Bits in Register**	**Comments**
Syntax	SER Rd	Rd is a working register
Operands	16≤d≤31	Rd can be R16 through R31
Operation	Rd←$FF	Set all bits in Rd to one's
Flags Affected	None	In SREG
Encoding	1110 1111 dddd 1111	
Description	Loads $FF directly to register Rd (destination register).	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	clr r16 ; clear r16 ser r17 ; set r17 out $18, r16 ; write zeros to PortB nop ; Delay (do nothing) 1 cycle out $18, r17 ; write ones to PortB	

SES	**Set Signed Flag**	**Comments**
Syntax	SES	
Operands	None	
Operation	S← 1	Set Signed (S) flag bit
Flags Affected	S	In SREG
Encoding	1001 0100 0100 1000	
Description	Sets the Signed Flag (S) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	ses ; set signed flag	

SET	**Set Test Flag**	**Comments**
Syntax	SET	
Operands	None	
Operation	T← 1	Set Test (T) flag bit
Flags Affected	T	In SREG
Encoding	1001 0100 0110 1000	
Description	Sets the Test Flag (T) in SREG (status register)	T Flag is a user defined flag bit
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	set ; set T flag	

SEV	**Set Overflow Flag**	**Comments**
Syntax	SEV	
Operands	None	
Operation	V← 1	Set Overflow (V) flag bit
Flags Affected	V	In SREG
Encoding	1001 0100 0011 1000	
Description	Sets the Overflow flag (V) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	sev ; set overflow flag	

SEZ	**Set Zero Flag**	**Comments**
Syntax	SEZ	
Operands	None	
Operation	Z← 1	Set Zero (Z) flag bit
Flags Affected	Z	In SREG
Encoding	1001 0100 0001 1000	
Description	Sets the Zero Flag (Z) in SREG (status register)	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	sez ; set zero flag	

SLEEP	**Sleep**	**Comments**
Syntax	SLEEP	
Operands	None	
Operation	-----	
Flags Affected	None	In SREG
Encoding	1001 0100 0001 1000	
Description	Sets the circuit in sleep mode defined by the MCU control register. When an interrupt wakes up the MCU from a sleep state, the instruction following the SLEEP instruction will be executed before the interrupt handler is executed.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	sleep ; put MCU in sleep mode	

ST	**Store Indirect from Register to SRAM Using Index X**	**Comments**
Syntax	ST X, Rr ST X+. Rr ST –X, Rr	X: Unchanged X: Post-increment X: Pre-increment
Operands	0≤r≤31	Rr is a working register 0 to 31
Operation	X is unchanged (X)← Rr X Post-increment (X)←Rr X← X+1 X Pre-decrement X←X-1 (X)←Rr	X is Unchanged (not inc or dec) After Rr is stored in SRAM, X increments X decrements, Rr value stored in SRAM
Flags Affected	None	In SREG
Encoding	X unchanged 1001 001r rrrr 1100 X Post-increment 1001 001r rrrr 1101 X Pre-decrement 1001 001r rrrr 1110	
Description	Stores one byte indirect from Register to SRAM. The SRAM location is pointed to by the X (16 bits) pointer register in the register file. Memory access is limited to the current SRAM page of 64k bytes. To access another SRAM page the RAMPX register in the I/O area has to be changed. The X pointer (index) register can be left unchanged after the operation, or it can be incremented or decremented. These features are especially suited for stack pointer usage of the X pointer register.	
Words	1	16 bit instruction
Cycles	2	2 machine cycles

Example	`clr r27 ; clear X high byte (XH)`
	`ldi r26, $20 ; set X low byte to $20 (XL)`
	`st X+, r0 ; store r0 in SRAM $20`
	`; (X post inc)`
	`st X, r1 ; store r1 in SRAM loc. $21`
	`ldi r26, $23 ; set X low byte to $23 (XL)`
	`st r2, X ; store r2 in SRAM loc. $23`
	`st r3, -X ; store r3 in SRAM loc. $22`
	`; (X pre dec).`	

ST (STD)	**Store Indirect from Register to SRAM Using Index Y**	**Comments**
Syntax	ST Y, Rr ST Y+. Rr ST –Y, Rr STD Y+q, Rr	Y: Unchanged Y: Post-increment Y: Pre-increment Y: Unchanged, q is displacement
Operands	0≤r≤31, 0≤q≤63	Rr is a working register 0 to 31, q displacement value 0 to 63
Operation	Y is unchanged (Y)← Rr Y Post-increment (Y)←Rr Y← Y+1 Y Pre-decrement Y←Y-1 (Y)←Rr Y added to q (Y+q)←Rr	Y is Unchanged (not inc or dec) After Rr is stored in SRAM, Y increments Y decrements, Rr value stored in SRAM Y added to q, Rr value stored in SRAM
Flags Affected	None	In SREG
Encoding	Y unchanged 1000 001r rrrr 1000 Y Post-increment 1001 001r rrrr 1001 Y Pre-decrement 1001 001r rrrr 1010 Y unchanged, q displace. 10q0 qq1r rrrr 1qqq	
Description	Stores one byte indirect from Register to SRAM. The SRAM location is pointed to by the Y (16 bits) pointer register in the register file. Memory access is limited to the current SRAM page of 64k bytes. To access another SRAM page the RAMPY register in the I/O area has to be changed. The Y pointer (index) register can be left unchanged after the operation, or it can be incremented or decremented. These features are especially suited for stack pointer usage of the Y pointer register.	
Words	1	16 bit instruction
Cycles	2	2 machine cycles
Example	clr r29 ; clear Y high byte (YH) ldi r28, $20 ; set Y low byte to $20 (YL) st Y+, r0 ; store r0 in SRAM $20 ; (Y post inc) st Y, r1 ; store r1 in SRAM loc. $21 ldi r28, $23 ; set Y low byte to $23 (YL) st r2, Y ; store r2 in SRAM loc. $23 st r3, -Y ; store r3 in SRAM loc. $22 ; (Y pre dec).	

ST (STD)	**Store Indirect from Register to SRAM Using Index Z**	**Comments**
Syntax	ST Z, Rr ST Z+. Rr ST –Z, Rr STD Z+q, Rr	Z: Unchanged Z: Post-increment Z: Pre-increment Z: Unchanged, q is displacement
Operands	0≤r≤31, 0≤q≤63	Rr is a working register 0 to 31, q displacement value 0 to 63
Operation	Z is unchanged (Z)← Rr Z Post-increment (Z)←Rr Z← Z+1 Z Pre-decrement Z←Z-1 (Z)←Rr Z added to q (Z+q)←Rr	Z is Unchanged (not inc or dec) After Rr is stored in SRAM, Z increments Z decrements, Rr value stored in SRAM Z added to q, Rr value stored in SRAM
Flags Affected	None	In SREG
Encoding	Z unchanged 1000 001r rrrr 0000 Z Post-increment 1001 001r rrrr 0001 Z Pre-decrement 1001 001r rrrr 0010 Z unchanged, q displace. 10q0 qq1r rrrr 0qqq	
Description	Stores one byte indirect from Register to SRAM. The SRAM location is pointed to by the Z (16 bits) pointer register in the register file. Memory access is limited to the current SRAM page of 64k bytes. To access another SRAM page the RAMPZ register in the I/O area has to be changed. The Y pointer (index) register can be left unchanged after the operation, or it can be incremented or decremented. These features are especially suited for stack pointer usage of the Z pointer register, but because the Z pointer register can be used for indirect subroutine calls, indirect jumps and table lookup, it is often more convenient to use the X and Y pointer as a dedicated stack pointer.	
Words	1	16 bit instruction
Cycles	2	2 machine cycles
Example	clr r31 ; clear Z high byte (ZH) ldi r30, $20 ; set Y low byte to $20 (ZL) st Z+, r0 ; store r0 in SRAM $20 ; (Z post inc) st Z, r1 ; store r1 in SRAM loc. $21 ldi r30, $23 ; set Z low byte to $23 (ZL) st r2, Z ; store r2 in SRAM loc. $23 st r3, -Z ; store r3 in SRAM loc. $22 ; (Z pre dec).	

STS	**Store Direct to SRAM**	**Comments**
Syntax	STS k, Rr	K is the address, and Rr is a working register (source register)
Operands	0≤r≤31, 0≤k≤65535	Rr is a working register 0 to 31, k address location 0 and 65535
Operation	(k)←Rr	Rr value stored at memory location k
Flags Affected	None	In SREG
Encoding	1001 001r rrrr 0000 kkkk kkkk kkkk kkkk	
Description	Stores one byte from a register to the SRAM. A 16-bit address must be supplied. Memory access is limited to the current SRAM page of 64k bytes. The SDS instruction uses the RAMPZ register to access memory above 64k bytes	Note: check the data sheets and memory map for individual AVR devices for specific resources.
Words	2	32 bit instruction
Cycles	3	3 machine cycles

Example	`lds r2, $FF00 ; load r2 with the contents of`
	`               ; SRAM`
	`add r2, r1     ; add r1 to r2`
	`sts $FF00, r2  ; write new value back to`
	`               ; SRAM loc. $FF00`	

SUB	**Subtract without Carry**	**Comments**
Syntax	SUB Rd, Rr	Rd and Rr working registers
Operands	0≤d≤31, 0≤r≤31	(d,r) registers 0 to 31
Operation	Rd← Rd-Rr	Subtract Rd-Rr store results in Rd
Flags Affected	H, S, V, N, Z, C	In SREG
Encoding	0001 10rd dddd rrrr	
Description	Subtracts two registers and places the result in the destination register Rd.	Note: does not use Carry Flag (C)
Words	1	16 bit instruction
Cycles	1	1 machine cycle

Example	`sub r13, r12    ; subtract r12 from r13`
	`                ; results in r13`	

SUBI	**Subtract Immediate**	**Comments**
Syntax	SUBI Rd, k	Rd working register, k constant
Operands	16≤d≤31, 0≤k≤255	(d) registers 16 to 31, k 0 to 255
Operation	Rd← Rd-k	Subtract Rd-k, store results in Rd
Flags Affected	H, S, V, N, Z, C	In SREG
Encoding	0101 kkkk dddd kkkk	
Description	Subtracts a register and a constant and places the result in the destination register Rd. This instruction is working on registers R16 to R31 and is very well suited for operations on the X, Y, and Z pointers.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	subi r22, $11 ; subtract $11 from r22 ; results in r22.	

SWAP	**Swap Nibbles**	**Comments**
Syntax	SWAP Rd	Rd working register
Operands	0≤d≤31	(d) registers 0 to 31
Operation	R(7-4)←Rd(3-0), R(3-0)←Rd(7-4)	Switch High nibble value to low nibble value
Flags Affected	None	In SREG
Encoding	1001 010d dddd 0010	
Description	Swaps high and low nibble in a register	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	ldi r16, $0F ; load r16 with $0F mov r1, r16 ; move value in r16 to r1 swap r1 ; swap high and low nibble r1 ; r1 results are $F0	

TST	**Test for Zero or Minus**	**Comments**
Syntax	TST Rd	Rd working register
Operands	0≤d≤31	(d) registers 0
Operation	Rd ← Rd AND Rd	AND Rd with Rd
Flags Affected	S, V, N, Z	In SREG
Encoding	0010 00dd dddd dddd	
Description	Tests if register is zero or negative. Performs a logical AND between a register and itself. The register will remain unchanged.	
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	tst r0 ; tst r0	

WDR	**Watchdog Reset**	**Comments**
Syntax	WDR	
Operands	None	
Operation	Watchdog Timer Reset	
Flags Affected	None	In SREG
Encoding	1001 0101 101X 1000	X is don't care
Description	Resets the watchdog timer. This instruction must be executed within a limited time given by the WD prescaler. See the Watchdog Timer hardware specifications	Note: Check individual specification sheets for each AVR device.
Words	1	16 bit instruction
Cycles	1	1 machine cycle
Example	wdr ; reset watchdog timer	