

CSSE1000

AVR AT90S8515 Instruction Set Summary

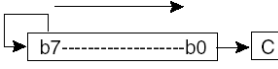
Operand Notation

Operand Notation	Meaning	Replace with	Opcode pattern
Rd	Destination register (could be source also)	r0...r31 r16...r31	dddd dddd
Rr	Source register	r0...r31	rrrrr
Rh:Rl	Pair of registers treated as 16 bit quantity	r25:r24 r27:r26 (or XH:XL) r29:r28 (or YH:YL) r31:r30 (or ZH:ZL)	dd (00) dd (01) dd (10) dd (11)
K	Constant data	0...255 (\$00 to \$FF) 0...63 (\$00 to \$3F)	KKKKKKKK KKKKKK
k	Constant address	-64...63 -2048...2047 0...65535	kkkkkkk kkkk kkkk kkkk kkkk kkkk kkkk kkkk
b	Bit number in register (or I/O register)	0...7	bbb
s	Bit in status register	0...7	sss
W	Y or Z register	Y or Z	W (1 for Y, 0 for Z)
P	I/O register number	0...63 (\$00 to \$3F) 0...31 (\$00 to \$1F)	PPPPPP PPPPP
q	Displacement for direct addressing	0...63	qqqqqq

Notation Used in Operation Description

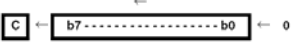

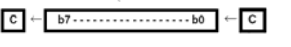
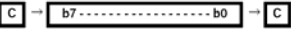
Operation Notation	Meaning
M[a]	Memory (SRAM) cell at address <i>a</i>
PM[a]	Program Memory (Flash) cell at byte address <i>a</i>
IO[a]	I/O register <i>a</i>
PC	Program Counter
SP	Stack Pointer
STACK	M[SP]
←	Assigned value
reg(<i>a</i>)	Bit <i>a</i> of register <i>reg</i> (numbered 7...0, most significant to least significant)
C	Carry bit in status register, same as SREG(0)
Z	Zero bit in status register, same as SREG(1)
N	Negative bit in status register, same as SREG(2)
V	Overflow bit in status register, same as SREG(3)
S	N⊕V bit in status register, same as SREG(4). S = sign of result corrected for any overflow. (Means result is negative (with no overflow) or would have been negative if overflow hadn't occurred.)
H	Half-carry bit in status register, same as SREG(5)
T	Transfer bit in status register, same as SREG(6) (see BLD, BST)
I	Global Interrupt Enable bit in status register, same as SREG(7)

Alphabetic Instruction Set Summary

Mnemonic and Operands	Operation	Description	Flags Affected	No. Ckts	OpCode	Example
ADC Rd, Rr	$Rd \leftarrow Rd + Rr + C$	Add two registers with carry	--HSVNZC	1	0001 11rd dddd rrrr	; add r1:r0 to r3:r2
ADD Rd, Rr	$Rd \leftarrow Rd + Rr$	Add two registers	--HSVNZC	1	0000 11rd dddd rrrr	add r2,r0 ; add low byte adc r3,r1 ; add high byte
ADIW Rh:Rl, K	$Rh:Rl \leftarrow Rh:Rl + K$	Add immediate to Word ($0 \leq K \leq 63$)	---SVNZC	2	1001 0110 KKdd KKKK	adiw ZH:ZL, 7 ; add 7 to Z
AND Rd, Rr	$Rd \leftarrow Rd \bullet Rr$	Logical AND two registers	---SVNZ-, V cleared	1	0010 00rd dddd rrrr	ldi r16, 1 ; set 0000 0001 in r16 and r2, r16 ; isolate bit 0 in r2
ANDI Rd, K	$Rd \leftarrow Rd \bullet K$	Logical AND with immediate ($16 \leq d \leq 31$)	---SVNZ-, V cleared	1	0111 KKKK dddd KKKK	andi r18,\$10 ; isolate bit 4 in r18
ASR Rd	$C \leftarrow Rd(0),$ $Rd(6...0) \leftarrow Rd(7...1),$ $Rd(7) \leftarrow Rd(7)$	Arithmetic shift right 	---SVNZC	1	1001 010d dddd 0101	asr r17 ; r17 = r17 / 2
BCLR s	$SREG(s) \leftarrow 0$	Clear bit in status register	SREG(s)	1	1001 0100 1sss 1000	bclr 7 ; disable interrupts
BLD Rd, b	$Rd(b) \leftarrow T$	Load bit in register from T	-T-----	1	1111 100d dddd 0bbb	bst r1, 2 ; store bit 2 of r1 in T bld r0, 4 ; load T into bit 4 of r0
BRBC s,k	if(SREG(s) = 0) $PC \leftarrow PC + k + 1$	Branch if status register flag cleared	-----	1,2*	1111 01kk kkkk ksss	cpi r20, 5 ; compare r20 to value 5 brbc 1,noteq ; Branch if zero flag 0 ... noteq: nop ; do nothing
BRBS s,k	if(SREG(s) = 1) $PC \leftarrow PC + k + 1$	Branch if status register flag set	-----	1,2*	1111 00kk kkkk ksss	bst r0, 3 ; load T with bit 3 of r0 brbs 6,bitset ; Branch if T was set
BRCC k	if(C=0) then $PC \leftarrow PC + k + 1$	Branch if carry cleared, Same as brbc 0,k and brsh	-----	1,2*	1111 01kk kkkk k000	add r22, r23 ; add r23 to r22 brcc nocarry ; branch if carry ; cleared
BRCS k	if(C=1) then $PC \leftarrow PC + k + 1$	Branch if carry set, Same as brbs 0,k and brlo	-----	1,2*	1111 00kk kkkk k000	cp r26, r25 ; compare r26 with r25 brcs label ; branch if carry set
BREQ k	if(Z=1) then $PC \leftarrow PC + k + 1$	Branch if equal, Same as brbs 1,k	-----	1,2*	1111 00kk kkkk k001	cp r1,r0 ; compare r1 and r0 breq label ; branch if equal
BRGE k	if(S=0) then $PC \leftarrow PC + k + 1$	Branch if greater or equal (signed), Same as brbc 4,k	-----	1,2*	1111 01kk kkkk k100	cp r1, r2 brge label ; branch if r1 ≥ r2 ; (signed)
BRHC k	if(H=0) then $PC \leftarrow PC + k + 1$	Branch if half-carry flag cleared, Same as brbc 5,k	-----	1,2*	1111 01kk kkkk k101	bhrc label
BRHS k	if(H=1) then $PC \leftarrow PC + k + 1$	Branch if half-carry flag set, Same as brbs 5,k	-----	1,2*	1111 00kk kkkk k101	bhrs label
BRID k	if(I=0) then $PC \leftarrow PC + k + 1$	Branch if interrupt disabled, Same as brbc 7,k	-----	1,2*	1111 01kk kkkk k111	brid label
BRIE k	if(I=1) then $PC \leftarrow PC + k + 1$	Branch if interrupts enabled, Same as brbs 7,k	-----	1,2*	1111 00kk kkkk k111	brie label
BRLO k	if(C=1) then $PC \leftarrow PC + k + 1$	Branch if lower, unsigned, Same as brbs 0,k and brcs	-----	1,2*	1111 00kk kkkk k000	cpi r19,\$8 ; compare r19 with \$8 brlo label ; branch if r19 < \$8 (unsigned)
BRLT k	if(S=1) then $PC \leftarrow PC + k + 1$	Branch if less than (signed), Same as brbs 4,k	-----	1,2*	1111 00kk kkkk k100	cpi r19,\$8 ; compare r19 with \$8 brlt label ; branch if r19 < \$8 ;(signed)
BRMI k	if(N=1) then $PC \leftarrow PC + k + 1$	Branch if minus, Same as brbs 2,k	-----	1,2*	1111 00kk kkkk k010	subi r18, 4 ; subtract 4 from r18 brim label ; branch if result ; negative
BRNE k	if(Z=0) then $PC \leftarrow PC + k + 1$	Branch if not equal, Same as brbc 1,k	-----	1,2*	1111 01kk kkkk k001	cpi r27, 5 ; compare r27 to 5 brne label ; branch if r27 ≠ 5
BRPL k	if(N=0) then $PC \leftarrow PC + k + 1$	Branch if plus, Same as brbc 2,k	-----	1,2*	1111 01kk kkkk k010	subi r26,\$50 ; subtract \$50 from r26 brpl pos ; branch if result positive
BRSH k	if(C=0) then $PC \leftarrow PC + k + 1$	Branch if same or higher, unsigned, Same as brbc 0,k and brc	-----	1,2*	1111 01kk kkkk k000	cp r1, r2 brsh label ; branch if r1 ≥ r2 ; (unsigned)
BRTC k	if(T=0) then $PC \leftarrow PC + k + 1$	Branch if T flag cleared, Same as brbc 6,k	-----	1,2*	1111 01kk kkkk k110	bst r3, 5 ; store bit 5 of r3 in T brtc label ; branch if bit was 0
BRTS k	if(T=1) then $PC \leftarrow PC + k + 1$	Branch if T flag set, Same as brbs 6,k	-----	1,2*	1111 00kk kkkk k110	bst r3, 5 ; store bit 5 of r3 in T brts label ; branch if bit was 1
BRVC k	if(V=0) then $PC \leftarrow PC + k + 1$	Branch if overflow flag is cleared, Same as brbc 3,k	-----	1,2*	1111 01kk kkkk k011	add r3, r4 ; add r4 to r3 brvc noover ; branch if no overflow

AVR AT90S8515 Instruction Set Summary

Mnemonic and Operands	Operation	Description	Flags Affected	No. Cycles	OpCode	Example
BRVS k	if(V=1) then PC←PC + k + 1	Branch if overflow flag is set, Same as brbs 3,k	-----	1,2*	1111 00kk kkkk k011	add r3, r4 ; add r4 to r3 brvs over ; branch if overflow
BSET s	SREG(s) ← 1	Set bit in status register	SREG(s)	1	1001 0100 0sss 1000	bset 6 ; set T flag
BST Rr,b	T←Rr(b)	Bit store from register to T	-T-----	1	1111 101d dddd 0bbb	bst r1, 2 ; store bit 2 of r1 in T bld r0, 4 ; load T into bit 4 of r0
CBI P, b	IO[P](b) ← 0	Clear bit in I/O register, only for 0≤P≤31	-----	2	1001 1000 PPPP Pbbb	cbi \$12, 7 ; Clear bit 7 in port D
CBR Rd, K	Rd←Rd• \overline{K}	Clear bit(s) in register, only for 16≤d≤31	---SVNZ-, V cleared	1	As per ANDi with K Complemented	cbr r16,\$F0 ; clr upper nibble of r16 cbr r18, 1 ; clear bit 0 in r18
CLC	C←0	Clear carry flag, Same as bclr 0	-----C	1	1001 0100 1000 1000	clc
CLH	H←0	Clear half-carry flag, Same as bclr 5	--H-----	1	1001 0100 1101 1000	clh
CLI	I←0	Clear global interrupt flag (disable interrupts), Same as bclr 7	I-----	1	1001 0100 1111 1000	cli
CLN	N←0	Clear negative flag, Same as bclr 2	-----N--	1	1001 0100 1010 1000	cln
CLR Rd	Rd←Rd ⊕ Rd	Clear register, Same as eor Rd, Rd	---SVNZ-, Z set; S,V,N cleared	1	0010 01Dd dddd DDDD (DDDDD=dddd)	clr r18 ; clear r18
CLS	S←0	Clear signed flag, Same as bclr 4	---S----	1	1001 0100 1100 1000	cls
CLT	T←0	Clear T flag, Same as bclr 6	-T-----	1	1001 0100 1110 1000	clt
CLV	V←0	Clear overflow flag, Same as bclr 3	---V----	1	1001 0100 1011 1000	clv
CLZ	Z←0	Clear zero flag, Same as bclr 1	-----Z-	1	1001 0100 1001 1000	clz
COM Rd	Rd← \overline{Rd} or Rd←\$FF - Rd	One's complement (inversion)	---SVNZC, V cleared, C set	1	1001 010d dddd 0000	com r4 ; invert bits in r4
CP Rd, Rr	Rd - Rr	Compare	--HSVNZC	1	0001 01rd dddd rrrr	cp r4, r19 ; compare r4 with r19 brne noteq ; branch if r4 ≠ r19
CPC Rd, Rr	Rd - Rr - C	Compare with Carry	--HSVNZC	1	0000 01rd dddd rrrr	; compare r3:r2 with r1:r0 cp r2, r0 ; compare low byte cpc r3, r1 ; compare high byte brne noteq ; branch if not equal
CPI Rd, K	Rd - K	Compare with immediate, 16≤d ≤31	--HSVNZC	1	0011 kkkk dddd kkkk	cpi r19, 3 ; compare r19 with 3 brne error ; branch if r19 ≠ 3
CPSE Rd, Rr	if(Rd=Rr) PC←PC + 2 (or 3)	Compare, skip if equal.	-----	1,2, 3†	0001 00rd dddd rrrr	cpse r4,r0 ; compare r4 to r0 neg r4 ; only executed if r4 ≠ r0 ... ; continue here
DEC Rd	Rd←Rd - 1	Decrement register	---SVNZ-	1	1001 010d dddd 1010	dec r17
EOR Rd, Rr	Rd←Rd ⊕ Rr	Exclusive OR two registers	---SVNZ-, V cleared	1	0010 01rd dddd rrrr	eor r0, r22 ; bitwise exclusive or
ICALL	PC←Z; Stack←PC + 1; SP←SP-2	Indirect call to [Z] (High bits of Z discarded)	-----	3	1001 0101 0000 1001	... ; put value in Z (ZH:ZL) icall ; call routine pointed to by Z
IJMP	PC←Z	Indirect Jump to [Z] (High bits of Z discarded)	-----	2	1001 0100 0000 1001	... ; put value in Z (ZH:ZL) ijmp ; jump to code at address Z
IN Rd, P	Rd←IO[P]	Load an I/O Location to Register	-----	1	1011 0PPd dddd PPPP	in r25, \$16 ; read port B
INC Rd	Rd←Rd + 1	Increment register	---SVNZ-	1	1001 010d dddd 0011	inc r22
LD Rd, W	Rd←M[W]	Load Indirect (Y or Z case)	-----	2	1000 000d dddd W000	clr YH ; clear high byte of Y ldi YL, \$60 ; set low byte of Y = \$60
LD Rd, X	Rd←M[X]	Load Indirect (X case)	-----	2	1001 000d dddd 1100	ld r1, Y+ ; load r1 with value at ; \$60
LDD Rd, W+q	Rd←M[W+q]	Load Indirect with Displacement (Y or Z only)	-----	2	10q0 qq0d dddd Wqqq	ld r0, Y ; load r0 with value at ; \$61
LD Rd, W+	Rd←M[W]; W←W+1	Load Indirect with Post- increment (Y or Z)	-----	2	1001 000d dddd W001	ldd r2, Y+2 ; load r2 with value at ; \$63
LD Rd, X+	Rd←M[X]; X←X+1	Load Indirect with Post- increment (X)	-----	2	1001 000d dddd 1101	; (Y still has value \$61)
LD Rd, -W	W←W-1; Rd←M[W]	Load Indirect with Pre-decrement (Y or Z)	-----	2	1001 000d dddd W010	ld r3, -Y ; load r3 with value at ; \$60
LD Rd, -X	X←X-1; Rd←M[X]	Load Indirect with Pre-decrement (X)	-----	2	1001 000d dddd 1110	
LDI Rd, K	Rd←K	Load Immediate, 16 ≤ d ≤ 31	-----	1	1110 kkkk dddd kkkk	ldi r30, \$F0 ; set S low byte to \$F0
LDS Rd, k	Rd←M[k]	Load Direct from SRAM, 0≤k≤65535	-----	2	1001 000d dddd 0000 kkkk kkkk kkkk kkkk	lds r2, \$FF00 ; load r2 with contents ; of mem. location \$FF00

Mnemonic and Operands	Operation	Description	Flags Affected	No. Ckls	OpCode	Example
LPM		$R0 \leftarrow PM[Z]$ Load program memory, Z contains a <i>byte</i> address. Least significant bit of Z selects low byte of the program word (if 0) or high byte (if 1)	-----	3	1001 0101 1100 1000	ldi ZH, high(table << 1); init. Z ldi ZL, low(table*2) lpm r16, Z ; load const from prog mem. ... table: .dw 0x5876 ; \$76 at prog. memory byte ; address table*2
LSL	Rd	$C \leftarrow Rd(7);$ $Rd(7...1) \leftarrow Rd(6...0);$ $Rd(0) \leftarrow 0$ 	--HSVNZC	1	0000 11dd dddd DDDD (DDDDD=dddd)	lsl r0 ; multiply r0 by 2
LSR	Rd	$C \leftarrow Rd(0);$ $Rd(6...0) \leftarrow Rd(7...1);$ $Rd(7) \leftarrow 0$ 	---SVNZC, N←0	1	1001 010d dddd 0110	lsr r0 ; divide r0 by 2, ; remainder in C
MOV	Rd, Rr	$Rd \leftarrow Rr$	-----	1	0010 11rd dddd rrrr	mov r16, r0 ; copy r0 to r16
NEG	Rd	$Rd \leftarrow \$00 - Rd$	--HSVNZC	1	1001 010d dddd 0001	neg r11 ; negate value in r11
NOP		No operation	-----	1	0000 0000 0000 0000	nop ; do nothing for 1 clock cycle
OR	Rd, Rr	$Rd \leftarrow Rd \text{ or } Rr$	---SVNZ-, V cleared	1	0010 10rd dddd rrrr	ldi r16, 3 or r2, r16 ; set bits 0 and 1 of r2
ORI	Rd, K	$Rd \leftarrow Rd \text{ or } K$ $16 \leq d \leq 31$	---SVNZ-, V cleared	1	0110 KKKK dddd KKKK	ori r17, \$0F; set bits 0,1,2,3 of r17
OUT	P, Rr	$IO[P] \leftarrow Rr$	-----	1	1011 1PPr rrrr PPPP	clr r16 ; clear r16 out \$18, r16 ; write zeroes to port B
POP	Rd	$SP \leftarrow SP+1;$ $Rd \leftarrow STACK$	-----	2	1001 000d dddd 1111	rcall routine ... routine: push r14 ; save r14 on stack push r13 ; save r13 on stack ... ; do stuff using r13, r14 pop r13 ; restore r13 pop r14 ; restore r14 ret ; return from ; subroutine
PUSH	Rr	$STACK \leftarrow Rr;$ $SP \leftarrow SP-1$	-----	2	1001 001r rrrr 1111	
RCALL	k	$PC \leftarrow PC + k + 1;$ $Stack \leftarrow PC + 1;$ $SP \leftarrow SP-2$ $-2048 \leq k \leq 2047$	-----	3	1101 kkkk kkkk kkkk	
RET		$SP \leftarrow SP+2;$ $PC \leftarrow Stack$	-----	4	1001 0101 0000 1000	int_handler: push r0 ... pop r0 reti ; return and enable interrupts
RETI		$SP \leftarrow SP+2;$ $PC \leftarrow Stack$	I-----, I is set	4	1001 0101 0001 1000	
RJMP	k	$PC \leftarrow PC + k + 1;$ $-2048 \leq k \leq 2047$	-----	2	1100 kkkk kkkk kkkk	rjmp RESET ... RESET: ...
ROL	Rd	$C \leftarrow Rd(7);$ $Rd(7...1) \leftarrow Rd(6...0);$ $Rd(0) \leftarrow C$ 	--HSVNZC	1	0001 11dd dddd DDDD (DDDDD=dddd)	; multiply r19:r18 by 2 lsl r18 rol r19
ROR	Rd	$C \leftarrow Rd(0);$ $Rd(6...0) \leftarrow Rd(7...1);$ $Rd(7) \leftarrow C$ 	---SVNZC	1	1001 010d dddd 0111	; divide r17:r16 (signed) by 2 asr r17 ror r16
SBC	Rd, Rr	$Rd \leftarrow Rd - Rr - C$	--HSVNZC	1	0000 10rd dddd rrrr	; subtract r1:r0 from r3:r2 sub r2, r0 ; subtract low byte sbc r3,r1 ; subtract high byte
SBCI	Rd, K	$Rd \leftarrow Rd - K - C$ $16 \leq d \leq 31$	--HSVNZC	1	0100 KKKK dddd KKKK	; subtract \$4F23 from r17:r16 subi r16, \$23 ; subtract low byte sbci r17, \$4F ; sub. w/ carry hi byte
SBI	P, b	$IO[P](b) \leftarrow 1$	-----	2	1001 1010 PPPP Pbbb	sbi \$12, 7 ; Set bit 7 in port D
SBIC	P, b	if $IO[P](b) = 0$ $PC \leftarrow PC + 2$ (or 3)	-----	1,2, 3 [†]	1001 1001 PPPP Pbbb	; wait until bit 3 of port D is 0 wait: sbic \$10, 3 rjmp wait ; if bit set, wait ... ; continue
SBIS	P, b	if $IO[P](b) = 1$ $PC \leftarrow PC + 2$ (or 3)	-----	1,2, 3 [†]	1001 1011 PPPP Pbbb	; wait until bit 4 of port B is 1 wait: sbis \$16, 4 rjmp wait ; if bit cleared, wait ... ; continue
SBIW	Rh:Rl, K	$Rh:Rl \leftarrow Rh:Rl - K$ $0 \leq K \leq 63$	---SVNZC	2	1001 0111 KKdd KKKK	sbiw r25:r24,1; sub. 1 from r25:r24 sbiw YH:YL,63 ; subtract 63 from Y

AVR AT90S8515 Instruction Set Summary

Mnemonic and Operands		Operation	Description	Flags Affected	No. Ckls	OpCode	Example
SBR	Rd, K	Rd←Rd or K	Set bit(s) in register, $16 \leq d \leq 31$, same as ori	---SVNZ-, V cleared	1	0110 KKKK dddd KKKK	sbr r16, 3 ; set bits 0 and 1 in r16
SBRC	Rr, b	if(Rr(b) = 0) PC←PC + 2 (or 3)	Skip if bit in register is cleared	-----	1,2, 3 [†]	1111 110r rrrr 0bbb	; r0 ← absolute value (r0) sbrc r0, 7 ; skip if bit 7 of r0 is 0 neg r0 ; negate r0 (if r0(7) = 1) ... ; continue
SBRS	Rr, b	if(Rr(b) = 1) PC←PC + 2 (or 3)	Skip if bit in register is set	-----	1,2, 3 [†]	1111 111r rrrr 0bbb	sbrs r0, 6 ; skip if bit 6 of r0 is 1 sub r2, r3 ; only if r0(6) = 0 ... ; continue
SEC		C←1	Set carry flag, Same as bset 0	-----C	1	1001 0100 0000 1000	sec
SEH		H←1	Set half-carry flag, Same as bset 5	--H-----	1	1001 0100 0101 1000	seh
SEI		I←1	Set global interrupt flag (enable interrupts). Instruction following sei will always be executed before any pending interrupts are handled. Same as bset 7	I-----	1	1001 0100 0111 1000	sei ; enable interrupts
SEN		N←1	Set negative flag, Same as bset 2	-----N--	1	1001 0100 0010 1000	sen
SER	Rd	Rd←\$FF	Set register, $16 \leq d \leq 31$, Same as LDI Rd, \$FF	-----	1	1110 1111 dddd 1111	ser r16 ; set r16 (all ones) out \$17, r16 ; make port B an output
SES		S←1	Set signed flag, Same as bset 4	---S----	1	1001 0100 0100 1000	ses
SET		T←1	Set T flag, Same as bset 6	-T-----	1	1001 0100 0110 1000	set
SEV		V←1	Set overflow flag, Same as bset 3	----V----	1	1001 0100 0011 1000	sev
SEZ		Z←1	Set zero flag, Same as bset 1	-----Z-	1	1001 0100 0001 1000	sez
SLEEP			Sleep. Sets CPU in sleep mode defined by the MCU control register	-----	1	1001 0101 1000 1000	sleep
ST	W, Rr	M[W] ←Rr	Store Indirect (Y or Z cases)	-----	2	1000 001r rrrr W000	clr r31 ; clear Z high byte
ST	X, Rr	M[X] ←Rr	Store Indirect (X case)	-----	2	1001 001r rrrr 1100	ldi r30, \$60 ; set Z low byte to \$60
ST	W+, Rr	M[W] ←Rr; W←W+1	Store Indirect with Post-increment (Y or Z)	-----	2	1001 001r rrrr W001	st Z+, r0 ; store r0 to \$60 st Z, r1 ; store r1 to \$61 std Z+2, r2 ; store r2 to \$63 ; (Z still is \$61)
ST	X+, Rr	M[X] ←Rr; X←X+1	Store Indirect with Post-increment (X)	-----	2	1001 001r rrrr 1101	st -Z, r3 ; store r3 to \$60
ST	-W, Rr	W←W-1; M[W] ←Rr	Store Indirect with Pre-decrement (Y or Z)	-----	2	1001 001r rrrr W010	
ST	-X, Rr	X←X-1; M[X] ←Rr	Store Indirect with Pre-decrement (X)	-----	2	1001 001r rrrr 1110	
STD	W+q,Rr	M[W+q] ←Rr	Store Indirect with Displacement (Y or Z only)	-----	2	10q0 qq1r rrrr Wqqq	
STS	k, Rr	M[k] ←Rr	Store Direct To SRAM	-----	2	1001 001r rrrr 0000 kkkk kkkk kkkk kkkk	lds r2, \$FF00 ; load r2 with value at ; \$FF00 add r2, r1 ; add r1 to r2 sts \$FF00, r2 ; Write back
SUB	Rd, Rr	Rd←Rd - Rr	Subtract two registers	--HSVNZC	1	0001 10rd dddd rrrr	sub r13, r12 ; subtract r12 from r13
SUBI	Rd, K	Rd←Rd - K	Subtract immediate, $16 \leq d \leq 31$	--HSVNZC	1	0101 KKKK dddd KKKK	subi r22, \$11 ; subtract \$11 from r22
SWAP	Rd	Rd(7...4) ←Rd(3...0); Rd(3...0) ←Rd(7...4)	Swap nibbles (i.e. high 4 bits is exchanged with low 4 bits)	-----	1	1001 010d dddd 0010	swap r1 ; swap high and low nibbles ; of r1
TST	Rd	Rd←Rd • Rd	Test for zero or minus, same as And Rd, Rd	---SVNZ-, V cleared	1	0010 00Dd dddd DDDD (DDDDD=dddd)	tst r0 ; test r0 breq label ; branch if r0 = 0
WDR			Watchdog reset	-----	1	1001 0101 1010 1000	wdr ; reset watchdog timer

* 1 cycle if branch is not taken (condition is false) or 2 cycles if branch is taken (condition is true)

† 1 cycle if no skip (condition is false), 2 cycles if condition is true and skip 16-bit instruction, 3 cycles if condition is true and skip 32-bit instruction