Mnemonic	Op-code	Operands
NOP	0000 0000 0000 0000	
MOVW	0000 0001 dddd rrrr	d = destination regiser pair (0,2,..30) r = source register pair (0,2,..30)
MULS	0000 0010 dddd rrrr	d = multiplicand 1 (16..31) r = multiplicand 2 (16..31)
MULSU	0000 0011 0ddd 0rrr	d = signed multiplicand (16..32) r = unsigned multiplicand (16..32)
FMUL	0000 0011 0ddd 1rrr	d = multiplicand register (16..23) r = multiplicand register (16..23)
FMULS	0000 0011 1ddd 0rrr	d = multiplicand register (16..23) r = multiplicand register (16..23)
FMULSU	0000 0011 1ddd 1rrr	d = signed multiplicand register (16..23) r = unsigned multiplicand register (16..23)
CPC	0000 01rd dddd rrrr	d = register A to be compared thru carry (0..31) r = register B to be compared thru carry (0..31)
SBC	0000 10rd dddd rrrr	d = destination register (0..31) r = source register (0..31)
ADD	0000 11rd dddd rrrr	d = destination register (0..31) r = source register (0..31)
CPSE	0001 00rd dddd rrrr	d = register A to be compared (0..31) r = register B to be compared (0..31)
CP	0001 01rd dddd rrrr	d = register A to be compared (0..31) r = register B to be compared (0..31)
SUB	0001 10rd dddd rrrr	d = destination register (0..31) r = source register (0..31)
ADC	0001 11rd dddd rrrr	d = destination register (0..31) r = source register (0..31)
AND	0010 00rd dddd rrrr	d = destination register (0..31) r = source register (0..31)
EOR	0010 01rd dddd rrrr	d = destination register (0..31) r = source register (0..31)
OR	0010 10rd dddd rrrr	d = destination register (0..31) r = source register (0..31)
MOV	0010 11rd dddd rrrr	d = destination register (0..31) r = source register (0..31)
CPI	0011 KKKK dddd KKKK	d = register to be compared (16..31) K = constant to be compared (0..255)
SBCI	0100 KKKK dddd KKKK	d = destination register (16..31) K = immediate constant (0..255)
SUBI	0101 KKKK dddd KKKK	d = destination register (16..31) K = immediate constant (0..255)
ORI	0110 KKKK dddd KKKK	d = destination register (16..31) K = immediate constant (0..255)
ANDI	0111 KKKK dddd KKKK	d = destination register (16..31) K = constant (0..255)
LDD Rd, p+q	10q0 qq0d dddd pqqq	d = destination register (0..31) q = offset (0..63), p = pointer (Z or Y)
STD Rd, p+q	10q0 qq1r rrrr pqqq	r = source register (0..31) q = offset (0..31), p = pointer (Z or Y)
LDS	1001 000d dddd 0000 kkkk kkkk kkkk kkkk	d = destination register (0..31), k = address to fetch (0..65535)
LPM Rd, Z	1001 000d dddd 0100	d = destination register (0..31)
LPM Rd, Z+	1001 000d dddd 0101	d = destination register (0..31)
ELPM Rd, Z	1001 000d dddd 0110	d = destination register (0..31)
ELPM Rd, Z+	1001 000d dddd 0111	d = destination register (0..31)
LD Rd, X	1001 000d dddd 1100	d = destination register (0..31)
LD Rd, p+	1001 000d dddd pp01	d = destination register (0..31) p = pointer (X, Y, Z)
LD Rd, -p	1001 000d dddd pp10	d = destination register (0..31)
POP	1001 000d dddd 1111	d = destination register (0..31)
STS	1001 001d dddd 0000 kkkk kkkk kkkk kkkk	d = source register (0..31) k = address to store (0..65535)
ST X, Rr	1001 001r rrrr 1100	r = source register (0..31)
ST p+, Rr	1001 001r rrrr pp01	r = source register (0..31) p = pointer (X, Y, Z)
ST -p, Rr	1001 001r rrrr pp10	r = source register (0..31) p = pointer (X, Y, Z)
PUSH	1001 001d dddd 1111	d = destination register (0..31)
COM	1001 010d dddd 0000	d = register to be 1's complemented (0..31)
SWAP	1001 010d dddd 0010	d = register to swap nibbles in (0..31)
NEG	1001 010d dddd 0001	d = register to be 2's complemented (0..31)
INC	1001 010d dddd 0011	d = destination register (0..31)
ASR	1001 010d dddd 0101	d = destination register (0..31)
LSR	1001 010d dddd 0110	d = destination register (0..31)
ROR	1001 010d dddd 0111	d = destination register (0..31)
BSET	1001 0100 0sss 1000	s = bit in SREG to set (0..7)
IJMP	1001 0100 0000 1001	
EIJMP	1001 0100 0001 1001	
DEC	1001 010d dddd 1010	d = destination register (0..31)
BCLR	1001 0100 1sss 1000	s = bit in SREG to clear (0..7)
JMP	1001 010k kkkk 110k kkkk kkkk kkkk kkkk	k = absolute address to jump to (0..4,194,304)
CALL	1001 010k kkkk 111k kkkk kkkk kkkk kkkk	k = absolute address to jump to (0..4,194,304)
RET	1001 0101 0000 1000	
ICALL	1001 0101 0000 1001	
RETI	1001 0101 0001 1000	
EICALL	1001 0101 0001 1001	
SLEEP	1001 0101 1000 1000	
BREAK	1001 0101 1001 1000	
WDR	1001 0101 1010 1000	
LPM	1001 0101 1100 1000	
ELPM	1001 0101 1101 1000	
SPM	1001 0101 1110 1000	
ADIW	1001 0110 KKdd KKKK	d = destination register pair (24:25, 26:27, 28:29, 30:31) K = constant (0..63)
SBIW	1001 0111 KKdd KKKK	d = destination register pair (24:25, 26:27, 28:29, 30:31) K = constant (0..63)
CBI	1001 1000 AAAA Abbb	A = destination I/O register (0..31) b = bit in I/O register to clear (0..7)
SBIC	1001 1001 AAAA Abbb	A = I/O register to inspect (0..31) b = bit in I/O register to test to determine branch
SBI	1001 1010 AAAA Abbb	A = destination I/O register (0..31) b = bit in I/O register to set (0..7)
SBIS	1001 1011 AAAA Abbb	A = I/O register to inspect (0..31) b = bit in I/O register to test to determine branch
MUL	1001 11rd dddd rrrr	d = multiplicand 1 (0..31) r = multiplicand 2 (0..31)
IN	1011 0AAd dddd AAAA	d = destination register (0..31) A = source I/O port (0..63)
OUT	1011 1AAr rrrr AAAA	r = source register (0..31) A = destination I/O port (0..63)
RJMP	1100 kkkk kkkk kkkk	k = offset to jump to (-2048 to +2047)
RCALL	1101 kkkk kkkk kkkk	k = offset to jump to (-2048 to +2047)
LDI	1110 KKKK dddd KKKK	d = destination register (0..31) K = immediate constant (0..255)
BRBS	1111 00kk kkkk ksss	s = bit in SREG to test (0..7) k = offset to jump by if bit is set
BRBC	1111 01kk kkkk ksss	s = bit in SREG to test (0..7) k = offset to jump by if bit is clear (-63..64)
BST	1111 101d dddd 0bbb	d = source register (0..31) b = bit in source register to copy into the T bit (0..7)
BLD	1111 100d dddd 0bbb	d = destination register (0..31) b = bit in destination register to transfer T bit into (0..7)
SBRC	1111 110r rrrr 0bbb	r = register to inspect (0..31) b = bit in register to test to determine branch (0..7)
SBRS	1111 111r rrrr 0bbb	r = register to inspect (0..31) b = bit in register to test to determine branch (0..7)

Unique op-codes by byte pattern:
81

Special Cases:
LSL: Assembler automatically inserts ADD with Rd == Rr
ROL: Assembler automatically inserts ADC with Rd == Rr
TST: Assembler automatically inserts AND with Rd == Rr
CLR: Assembler automatically inserts EOR with Rd == Rr
SBR: Assembler automatically replaces SBR with ORI
CBR: Assembler automatically replaces with ANDI with K complemented
LDD Rd, Z+q: See LDD op-code, with (p = 0)
LDD Rd, Y+q: See LDD op-code, with (p = 1)
LD Rd, Z: See LDD op-code, with (qqqqqq = 000000, p = 0)
LD Rd, Y: See LDD op-code, with (qqqqqq = 000000, p = 1)
STD Rr, Z+q: See STD op-code, with (p = 0)

STD Rr, Y+q: See STD op-code, with (p = 1)
ST Rr, Z: See STD op-code, with (qqqqqq = 000000, p = 0)
ST Rr, Y: See STD op-code, with (qqqqqq = 000000, p = 1)
LD Rd, Z+: See LD Rd, p+ op-code, with (pp = 00)
LD Rd, Y+: See LD Rd, p+ op-code, with (pp = 10)
LD Rd, X+: See LD Rd, p+ op-code, with (pp = 11)
LD Rd, -Z: See LD Rd, -p op-code, with (pp = 00)
LD Rd, -Y: See LD Rd, -p op-code, with (pp = 10)
LD Rd, -X: See LD Rd, -p op-code, with (pp = 11)
ST Z+, Rr: See ST p+, Rr op-code, with (pp = 00)
ST Y+, Rr: See ST p+, Rr op-code, with (pp = 10)
ST X+, Rr: See ST p+, Rr op-code, with (pp = 11)
ST -Z, Rr: See ST -p, Rr op-code, with (pp = 00)
ST -Y, Rr: See ST -p, Rr op-code, with (pp = 10)
ST -X, Rr: See ST -p, Rr op-code, with (pp = 11)
SEC: Assembler inserts BSET, with (sss = 000)
SEZ: Assembler inserts BSET, with (sss = 001)
SEN: Assembler inserts BSET, with (sss = 010)
SEV: Assembler inserts BSET, with (sss = 011)
SES: Assembler inserts BSET, with (sss = 100)
SEH: Assembler inserts BSET, with (sss = 101)
SET: Assembler inserts BSET, with (sss = 110)
SEI: Assembler inserts BSET, with (sss = 111)
CLC: Assembler inserts BCLR, with (sss = 000)
CLZ: Assembler inserts BCLR, with (sss = 001)
CLN: Assembler inserts BCLR, with (sss = 010)
CLV: Assembler inserts BCLR, with (sss = 011)
CLS: Assembler inserts BCLR, with (sss = 100)
CLH: Assembler inserts BCLR, with (sss = 101)
CLT: Assembler inserts BCLR, with (sss = 110)
CLI: Assembler inserts BCLR, with (sss = 111)
SER: Assember automatically inserts LDI with KKKKKKKK = 11111111
BRCS, BRLO: Assembler automatically inserts BRBS, with (sss = 000)
BREQ: Assembler automatically inserts BRBS, with (sss = 001)
BRMI: Assembler automatically inserts BRBS, with (sss = 010)
BRVS: Assembler automatically inserts BRBS, with (sss = 011)
BRLT: Assembler automatically inserts BRBS, with (sss = 100)
BRHS: Assembler automatically inserts BRBS, with (sss = 101)
BRTS: Assembler automatically inserts BRBS, with (sss = 110)
BRIE: Assembler automatically inserts BRBS, with (sss = 111)
BRCC, BRSH: Assembler automatically inserts BRBC, with (sss = 000)
BRNE: Assembler automatically inserts BRBC, with (sss = 001)
BRPL: Assembler automatically inserts BRBC, with (sss = 010)
BRVC: Assembler automatically inserts BRBC, with (sss = 011)
BRGE: Assembler automatically inserts BRBC, with (sss = 100)
BRHC: Assembler automatically inserts BRBC, with (sss = 101)
BRTC: Assembler automatically inserts BRBC, with (sss = 110)
BRID: Assembler automatically inserts BRBC, with (sss = 111)