**AVR instruction set summary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mnemonics | Operands | Description | Operation | Flags |
| Arithmetic and Logic Instructions | | | | |
| ADD | Rd, Rr | Add without Carry | Rd ← Rd + Rr | Z,C,N,V,S,H |
| ADC | Rd, Rr | Add with Carry | Rd ← Rd + Rr + C | Z,C,N,V,S,H |
| AND | Rd, Rr | Logical AND | Rd ← Rd • Rr | Z,N,V,S |
| ANDI | Rd, K | Logical AND with Immediate | Rd ← Rd • K | Z,N,V,S |
| SUB | Rd, Rr | Subtract without Carry | Rd ← Rd - Rr | Z,C,N,V,S,H |
| SUBI | Rd, K | Subtract Immediate | Rd ← Rd - K | Z,C,N,V,S,H |
| OR | Rd, Rr | Logical OR | Rd ← Rd v Rr | Z,N,V,S |
| EOR | Rd, Rr | Exclusive OR | Rd ← Rd ⊕ Rr | Z,N,V,S |
| INC | Rd | Increment | Rd ← Rd + 1 | Z,N,V,S |
| DEC | Rd | Decrement | Rd ← Rd – 1 | Z,N,V,S |
| MUL(1) | Rd, Rr | Multiply Unsigned | R1:R0 ← Rd x Rr (UU) | Z,C |
| Branch Instructions | | | | |
| RJMP | k | Relative Jump | PC ← PC + k + 1 | None |
| JMP(1) | k | Jump | PC ← k | None |
| RCALL | k | Relative Call Subroutine | PC ← PC + k + 1 | None |
| CALL(1) | k | call Subroutine | PC ← k | None |
| RET |  | Subroutine Return | PC ← STACK | None |
| RETI |  | Interrupt Return | PC ← STACK | I |
| CP | Rd, Rr | Compare | Rd - Rr | Z,C,N,V,S,H |
| SBRC | Rr, b | Skip if Bit in Register Cleared | if (Rr(b) = 0) PC ← PC + 2 or 3 | None |
| SBRS | Rr, b | Skip if Bit in Register Set | if (Rr(b) = 1) PC ← PC + 2 or 3 | None |
| SBIC | A, b | Skip if Bit in I/O Register Cleared | if (I/O(A, b) = 0) PC ← PC + 2 or 3 | None |
| SBIS | A, b | Skip if Bit in I/O Register Set | If (I/O(A, b) =1) PC ← PC + 2 or 3 | None |
| BREQ | k | Branch if Equal | if (Z = 1) then PC ← PC + k + 1 | None |
| BRNE | k | Branch if Not Equal | if (Z = 0) then PC ← PC + k + 1 | None |
| BRSH | k | Branch if Same or Higher | if (C = 0) then PC ← PC + k + 1 | None |
| BRLO | k | Branch if Lower | if (C = 1) then PC ← PC + k + 1 | None |
| Data Transfer Instructions | | | | |
| MOV | Rd, Rr | Copy Register | Rd ← Rr | None |
| LDI | Rd, K | Load Immediate | Rd ← K | None |
| ST(2) | Y, Rr | Store Indirect | (Y) ← Rr | None |
| IN | Rd, A | In From I/O Location | Rd ← I/O(A) | None |
| OUT | A, Rr | Out To I/O Location | I/O(A) ← Rr | None |
| PUSH(1) | Rr | Push Register on Stack | STACK ← Rr | None |
| POP(1) | Rd | Pop Register from Stack | Rd ← STACK | None |
| Bit and Bit-test Instructions | | | | |
| SBI | A, b | Set Bit in I/O Register | I/O(A, b) ← 1 | None |
| CBI | A, b | Clear Bit in I/O Register | I/O(A, b) ← 0 | None |
| SEI |  | Global Interrupt | Enable I ← 1 | I |
| CLI |  | Global Interrupt | Disable I ← 0 | I |

1. This instruction is not available in all devices. Refer to the device specific instruction set summary.
2. Not all variants of this instruction are available in all devices. Refer to the device specific instruction set summary.

Syntax: Operands: Program Counter:

ADC Rd, Rr 0 ≤ d ≤ 31, 0 ≤ r ≤ 31 PC ← PC + 1

ADD Rd, Rr 0 ≤ d ≤ 31, 0 ≤ r ≤ 31 PC ← PC + 1

AND Rd, Rr 0 ≤ d ≤ 31, 0 ≤ r ≤ 31 PC ← PC + 1

ANDI Rd, K 16 ≤ d ≤ 31, 0 ≤ K ≤ 255 PC ← PC + 1

BREQ k -64 ≤ k ≤ +63 PC ← PC + k + 1

PC ← PC + 1, if condition is false

CBI A, b 0 ≤ A ≤ 31, 0 ≤ b ≤ 7 PC ← PC + 1

CP Rd, Rr 0 ≤ d ≤ 31, 0 ≤ r ≤ 31 PC ← PC + 1

DEC Rd 0 ≤ d ≤ 31 PC ← PC + 1

IN Rd, A 0 ≤ d ≤ 31, 0 ≤ A ≤ 63 PC ← PC + 1

INC Rd 0 ≤ d ≤ 31 PC ← PC + 1

JMP k 0 ≤ k < 4M PC ← k

LDI Rd, K 16 ≤ d ≤ 31, 0 ≤ K ≤ 255 PC ← PC + 1

MOV Rd, Rr 0 ≤ d ≤ 31, 0 ≤ r ≤ 31 PC ← PC + 1

MUL Rd, Rr 0 ≤ d ≤ 31, 0 ≤ r ≤ 31 PC ← PC + 1

OUT A, Rr 0 ≤ r ≤ 31, 0 ≤ A ≤ 63 PC ← PC + 1

RJMP k -2K ≤ k < 2K PC ← PC + k + 1

SBI A, b 0 ≤ A ≤ 31, 0 ≤ b ≤ 7 PC ← PC + 1

SBIC A, b 0 ≤ A ≤ 31, 0 ≤ b ≤ 7 PC ← PC + 1, Condition false - no skip

PC ← PC + 2, Skip a one word instruction

PC ← PC + 3, Skip a two word instruction

SBRC Rr, b 0 ≤ r ≤ 31, 0 ≤ b ≤ 7 PC ← PC + 1, Condition false - no skip

PC ← PC + 2, Skip a one word instruction

PC ← PC + 3, Skip a two word instruction

**8088 (8086) instruction set**

|  |  |  |
| --- | --- | --- |
| Instructions | Description | Notes |
| ADC | Add with Carry | destination := destination + source + carry\_flag |
| ADD | Add without Carry | (1) r/m += r/imm; (2) r += m/imm; |
| AND | Logical AND | (1) r/m &= r/imm; (2) r &= m/imm; |
| CALL | call Subroutine | **push** eip; |
| CLI | Global Interrupt Disable | I ← 0 |
| CMP | Compare |  |
| DEC | Decrement by 1 |  |
| DIV | Unsigned divide | DX:AX = DX:AX / r/m; resulting DX = remainder |
| IN | Input from port | (1) AL = port[imm]; (2) AL = port[DX];  (3) AX = port[DX]; |
| INC | Increment by 1 |  |
| INT | Call to interrupt |  |
| IRET | Return from interrupt |  |
| JE | Branch if Equal | if (Z = 1) then PC ← PC + k + 1 |
| JGE | Branch if Same or Higher | if (C = 0) then PC ← PC + k + 1 |
| JL | Branch if Lower | if (C = 1) then PC ← PC + k + 1 |
| JMP | Jump |  |
| JNE | Branch if Not Equal | if (Z = 0) then PC ← PC + k + 1 |
| MOV | copies data from one location to another | (1) r/m = r; (2) r = r/m; |
| MUL | Multiply Unsigned | (1) DX:AX = AX \* r/m; (2) AX = AL \* r/m; |
| NOT | Negate the operand, logical NOT | r/m ^= -1; |
| OR | Logical OR | (1) r/m |= r/imm; (2) r |= m/imm; |
| OUT | Output to port | (1) port[imm] = AL; (2) port[DX] = AL;  (3) port[DX] = AX; |
| POP | Pop Register from Stack | r/m = \*SP++; |
| PUSH | Push Register on Stack | \*--SP = r/m; |
| RET | Subroutine Return | It will be translated to a RETN or a RETF |
| RETF | Return from far procedure |  |
| RETN | Return from near procedure |  |
| ROL | Rotate left |  |
| ROR | Rotate right |  |
| STI | Global Interrupt Enable | I ← 1 |
| SUB | Subtract without Carry |  |
| XOR | Exclusive OR | (1) r/m ^= r/imm; (2) r ^= m/imm; |

(LDS: Load pointer using DS)