| Mnemonic | Operands | Description | Words | Cycles | Status bits affected |
|---|---|---|---|---|---|
| `movlw` | `k` | Move literal value to WREG | 1 | 1 | - |
| `movwf` | `f(, BANKED)` | Move WREG to f | 1 | 1 | - |
| `movff` | `f_S, f_D` | Move $f_S$ to $f_D$ (both with full 12-bit addresses) | 2 | 2 | - |
| `movf` | `f, F/W(, BANKED)` | Move f to F or WREG | 1 | 1 | Z, N |
| | | | | | |
| `lfsr` | `i, k` | Load FSRi with full 12-bit address, where i = 0 to 2 | 2 | 2 | - |
| `clrf` | `f(, BANKED)` | Clear f | 1 | 1 | Z |
| `setf` | `f(, BANKED)` | $0xff \rightarrow f$ | 1 | 1 | - |
| `movlb` | `k` | Move literal value to BSR<3 : 0>, where k = 0 to 15, to set the bank for direct addressing | 1 | 1 | - |
| `swapf` | `f, F/W(, BANKED)` | Swap nibbles of f, putting result in F or WREG | 1 | 1 | - |
| | | | | | |
| `bcf` | `f, b(, BANKED)` | Clear bit b of register f, where b = 0 to 7 | 1 | 1 | - |
| `bsf` | `f, b(, BANKED)` | Set bit b of register f, where b = 0 to 7 | 1 | 1 | - |
| `btg` | `f, b(, BANKED)` | Toggle bit b of register f, where b = 0 to 7 | 1 | 1 | - |
| | | | | | |
| `rlcf` | `f, F/W(, BANKED)` | Copy f into F or WREG; rotate F or WREG left through carry bit (9-bit rotate left) | 1 | 1 | C, N, Z |
| `rlncf` | `f, F/W(, BANKED)` | Copy f into F or WREG; rotate F or WREG left without carry bit (8-bit rotate left) | 1 | 1 | N, Z |
| `rrcf` | `f, F/W(, BANKED)` | Copy f into F or WREG; rotate F or WREG right through carry bit (9-bit rotate right) | 1 | 1 | C, N, Z |
| `rrncf` | `f, F/W(, BANKED)` | Copy f into F or WREG; rotate F or WREG right without carry bit (8-bit rotate right) | 1 | 1 | N, Z |
| | | | | | |
| `incf` | `f, F/W(, BANKED)` | Increment f, putting result in F or WREG | 1 | 1 | C, DC, Z, OV, N |
| `decf` | `f, F/W(, BANKED)` | Decrement f, putting result in F or WREG | 1 | 1 | C, DC, Z, OV, N |
| `comf` | `f, F/W(, BANKED)` | Complement f, putting result in F or WREG | 1 | 1 | Z, N |
| `negf` | `f(, BANKED)` | Change sign of a twos-complement-coded number | 1 | 1 | C, DC, Z, OV, N |
| | | | | | |
| `andlw` | `k` | AND literal value into WREG | 1 | 1 | Z, N |
| `andwf` | `f, F/W(, BANKED)` | AND WREG with f, putting result in F or WREG | 1 | 1 | Z, N |
| `iorlw` | `k` | Inclusive-OR literal value into WREG | 1 | 1 | Z, N |
| `iorwf` | `f, F/W(, BANKED)` | Inclusive-OR WREG with f, putting result in F or WREG | 1 | 1 | Z, N |
| `xorlw` | `k` | Exclusive-OR literal value into WREG | 1 | 1 | Z, N |
| `xorwf` | `f, F/W(, BANKED)` | Exclusive-OR WREG with f, putting result in F or WREG | 1 | 1 | Z, N |
| | | | | | |
| `addlw` | `k` | Add literal value into WREG | 1 | 1 | C, DC, Z, OV, N |
| `addwf` | `f, F/W(, BANKED)` | Add WREG and f, putting result in F or WREG | 1 | 1 | C, DC, Z, OV, N |
| `addwfc` | `f, F/W(, BANKED)` | Add WREG and f and carry bit, putting result in F or WREG | 1 | 1 | C, DC, Z, OV, N |
| `daw` | | Decimal adjust sum of two packed BCD digits to correct packed BCD result in WREG | 1 | 1 | C |
| `sublw` | `k` | Subtract WREG from literal value, putting result in WREG | 1 | 1 | C, DC, Z, OV, N |
| `subwf` | `f, F/W(, BANKED)` | Subtract WREG from f, putting result in f or WREG | 1 | 1 | C, DC, Z, OV, N |
| `subwfb` | `f, F/W(, BANKED)` | Subtract WREG and borrow bit from f, putting result in F or WREG | 1 | 1 | C, DC, Z, OV, N |
| `subfwb` | `f, F/W(, BANKED)` | Subtract f and borrow bit from WREG, putting result in F or WREG | 1 | 1 | C, DC, Z, OV, N |
| `mullw` | `k` | Multiply WREG with literal value, putting result in PRODH : PRODL (WREG remains unchanged) | 1 | 1 | - |
| `mulwf` | `f(, BANKED)` | Multiply WREG with f, putting result in PRODH : PRODL (WREG and f remain unchanged) | 1 | 1 | - |
| | | | | | |
| | | | | | |
| `btfsc` | `f, b(, BANKED)` | Test bit b of register f, where b = 0 to 7 ; skip if clear | 1 | 1/2 | - |
| `btfss` | `f, b(, BANKED)` | Test bit b of register f, where b = 0 to 7; skip if set | 1 | 1/2 | - |
| | | | | | |

| Mnemonic | Operands | Description | Words | Cycles | Status bits affected |
|---|---|---|---|---|---|
| bra | label | Branch to labeled instruction (within ±64 one-word instructions) | 1 | 2 | - |
| goto | label | Go to labeled instruction (anywhere) | 2 | 2 | - |
| bc | label | If carry (C=1), then branch to labeled instruction (within ±64 one-word instructions) | 1 | 1/2 | - |
| bnc | label | If no carry (C=0), then branch to labeled instruction (within ±64 one-word instructions) | 1 | 1/2 | - |
| bz | label | If zero (Z=1), then branch to labeled intruction (within ±64 one-word instructions) | 1 | 1/2 | - |
| bnz | label | If not zero (Z=0), then branch to labeled instruction (within ±64 one-word instructions) | 1 | 1/2 | - |
| bn | label | If negative (N=1), then branch to labeled instruction (within ±64 one-word instructions) | 1 | 1/2 | - |
| bnn | label | If not negative (N=0), then branch to labeled instruction (within ±64 one-word instructions) | 1 | 1/2 | - |
| bov | label | If overflow (OV=1), then branch to labeled instruction (within ±64 one-word instructions) | 1 | 1/2 | - |
| bnov | label | If no overflow (OV=0), then branch to labeled instruction (within ±64 one-word instructions) | 1 | 1/2 | - |
|  |  |  |  |  |  |
| cpfseq | f(, BANKED) | Skip if f is equal to WREG | 1 | 1/2 | - |
| cpfsgt | f(, BANKED) | Skip if f is greater than WREG (unsigned compare) | 1 | 1/2 | - |
| cpfslt | f(, BANKED) | Skip if f is less than WREG (unsigned compare) | 1 | 1/2 | - |
|  |  |  |  |  |  |
| tstfsz | f(, BANKED) | Test f; skip if zero | 1 | 1/2 | - |
| decfsz | f, F/W(, BANKED) | Decrement f, putting result in F or WREG; skip if zero | 1 | 1/2 | - |
| dcfsnz | f, F/W(, BANKED) | Decrement f, putting result in F or WREG; skip if not zero | 1 | 1/2 | - |
| incfsz | f, F/W(, BANKED) | Increment f, putting result in F or WREG; skip if zero | 1 | 1/2 | - |
| infsnz | f, F/W(, BANKED) | Increment f, putting result in F or WREG; skip if not zero | 1 | 1/2 | - |
|  |  |  |  |  |  |
| rcall | label | Call labeled subroutine (within ±512 one-word instructions) | 1 | 2 | - |
| call | label | Call labeled subroutine (anywhere) | 2 | 2 | - |
| call | label, FAST | Call labeled subroutine (anywhere); copy state to shadow registers: (WREG)→WS, (STATUS)→STATUSS, (BSR)→BSRS | 2 | 2 | - |
| return |  | Return from subroutine | 1 | 2 | - |
| return | FAST | Return from subroutine; restore state from shadow registers: (WS)→WREG, (STATUSS)→STATUS, (BSRS)→BSR | 1 | 2 | C, DC, Z, OV, N |
| retlw | k | Return from subroutine, putting literal value in WREG | 1 | 2 | - |
|  |  |  |  |  |  |
| retfie |  | Return from interrupt; reenable interrupts | 1 | 2 | - |
| retfie | FAST | Return from interrupt; restore state from shadow registers: (WS)→WREG, (STATUSS)→STATUS, (BSRS)→BSR; reenable interrupts | 1 | 2 | C, DC, Z, OV, N |
|  |  |  |  |  |  |
| push |  | Push address of next instruction onto stack | 1 | 1 | - |
| pop |  | Discard address on top of stack | 1 | 1 | - |
| clrwdt |  | Clear watchdog timer | 1 | 1 | - |
| sleep |  | Go into standby mode | 1 | 1 | - |
| reset |  | Software reset to same state as is achieved with the MCLR input | 1 | 1 | C, DC, Z, OV, N |
| nop |  | No operation | 1 | 1 | - |
|  |  |  |  |  |  |
| tblrd* |  | Read from program memory location pointed to by TBLPTR into TABLAT | 1 | 2 | - |
| tblrd*+ |  | Read from program memory location pointed to by TBLPTR into TABLAT, then increment TBLPTR | 1 | 2 | - |
| tblrd*− |  | Read from program memory location pointed to by TBLPTR into TABLAT, then decrement TBLPTR | 1 | 2 | - |
| tblrd+* |  | Increment TBLPTR, then read from program memory location pointed to by TBLPTR into TABLAT | 1 | 2 | - |
|  |  |  |  |  |  |