Useful arithmetic and logical operations for the PIC24 processor

In these examples

- `Wx` is a PIC register (from `W0` to `W15`)
- `WREG` is PIC register `W0`
- `f` is a file register
- `#lit`*n* is an *n*-bit literal value
- `&Wx` is the file register whose address is stored in `Wx`

PIC instruction	C equivalent	Status bits set
`add Wa, Wb, Wd`	`Wd = Wa + Wb ;`	`C,N,V,Z`
`add f`	`f = f + WREG ;`	`C,N,V,Z`
`add f, WREG`	`WREG = f + WREG ;`	`C,N,V,Z`
`add #lit10, Wd`	`Wd = Wd + lit10 ;`	`C,N,V,Z`
`add Ws, #lit5, Wd`	`Wd = Ws + lit5 ;`	`C,N,V,Z`
`and Wa, Wb, Wd`	`Wd = Wa & Wb ;`	`N,Z`
`and f`	`f = f & WREG ;`	`N,Z`
`and f, WREG`	`WREG = f & WREG ;`	`N,Z`
`and #lit10, Wd`	`Wd = Wd & lit10 ;`	`N,Z`
`and Ws, #lit5, Wd`	`Wd = Ws & lit5 ;`	`N,Z`
`com Ws, Wd`	`Wd = ~Ws ;`	`N,Z`
`com f`	`f = ~f;`	`N,Z`
`com f, WREG`	`WREG = ~f ;`	`N,Z`
`inc f`	`f = f + 1 ;`	`C,N,V,Z`
`inc f, WREG`	`WREG = f + 1 ;`	`C,N,V,Z`
`inc Ws, Wd`	`Wd = Ws + 1 ;`	`C,N,V,Z`
`ior Wa, Wb, Wd`	`Wd = Wa | Wb ;`	`N,Z`
`ior f`	`f = f | WREG ;`	`N,Z`
`ior f, WREG`	`WREG = f | WREG ;`	`N,Z`
`ior #lit10, Wd`	`Wd = Wd | lit10 ;`	`N,Z`
`ior Ws, #lit5, Wd`	`Wd = Ws | lit5 ;`	`N,Z`
`mov Ws, Wd`	`Wd = Ws ;`	*none*
`mov Ws, [Wd]`	`&Wd = Ws ;`	*none*
`mov [Ws], Wd`	`Wd = &Ws ;`	*none*
`mov Ws, f`	`f = Wd ;`	*none*
`mov f, Wd`	`Wd = f ;`	*none*
`mov WREG, f`	`f = WREG ;`	*none*
`mov f, WREG`	`WREG = f ;`	`N,Z`
`mov #lit16, Wd`	`Wd = lit16 ;`	*none*
`sub Wa, Wb, Wd`	`Wd = Wa - Wb ;`	`C,N,V,Z`
`sub f`	`f = f - WREG ;`	`C,N,V,Z`
`sub f, WREG`	`WREG = f - WREG ;`	`C,N,V,Z`
`sub #lit10, Wd`	`Wd = Wd - lit10 ;`	`C,N,V,Z`
`sub Ws, #lit5, Wd`	`Wd = Ws - lit5 ;`	`C,N,V,Z`

Useful control flow operations for the PIC24 processor

The following instructions do not modify PIC registers, but they do set the STATUS bits based on the result of a computed result.

PIC instruction		C equivalent	Status bits set
cp	f	f - WREG	C,N,V,Z
cp	Wa, #lit5	Wa - lit5	C,N,V,Z
cp	Wa, Wb	Wa - Wb	C,N,V,Z
cp0	f	f - 0	C,N,V,Z
cp0	Wb	Wb - 0	C,N,V,Z

The following operations branch after (when appropriate) inspecting the status bits or by testing the result of the previous comparison operations of values A and B (or A and 0 for cp0)

PIC instruction		C equivalent
bra	C,loc	goto loc, if C bit is set
bra	GE,loc	goto loc, if A >= B when A and B are signed
bra	GEU,loc	goto loc, if A >= B when A and B are unsigned
bra	GT,loc	goto loc, if A > B when A and B are signed
bra	GTU,loc	goto loc, if A > B when A and B are unsigned
bra	LE,loc	goto loc, if A <= B when A and B are signed
bra	LEU,loc	goto loc, if A <= B when A and B are unsigned
bra	LT,loc	goto loc, if A < B when A and B are signed
bra	LTU,loc	goto loc, if A < B when A and B are unsigned
bra	N,loc	goto loc, if N bit is set
bra	NC,loc	goto loc, if C bit is *not* set
bra	NN,loc	goto loc, if N bit is *not* set
bra	NOV,loc	goto loc, if V bit is *not* set
bra	NZ,loc	goto loc, if Z bit is *not* set
bra	OV,loc	goto loc, if V bit is set
bra	Z,loc	goto loc, if Z bit is set
bra	loc	goto loc (unconditionally)
bra	Wd	goto addressed stored in Wd (unconditionally)

The following operations are used for stack access, W15 (sp) contains the address one past the top of the stack

PIC instruction		C equivalent
pop	f	sp = sp - 2 ; f = &sp ;
pop	Wd	sp = sp - 2 ; Wd = &sp ;
push	f	&sp = f ; sp = sp + 2 ;
push	Ws	&sp = Ws ; sp = sp + 2 ;

PIC24 processor instruction format

Here is the instruction format for the some ADD and MOV instructions

PIC instruction	Raw bits	Notes
add Wa, Wb, Wd	01000aaaa0000dddd000bbbb	a
add f	10110100001fffffffffffff	b
add f, WREG	10110100000fffffffffffff	b
add #lit10, Wd	1011000000nnnnnnnnnndddd	a,d
add Ws, #lit5, Wd	01000ssss0000dddd11nnnnn	a,d
mov Ws, Wd	0111100000000dddd000ssss	a
mov Ws, [Wd]	0111100000001dddd000ssss	a
mov [Ws], Wd	0111100000000dddd001ssss	a
mov Ws, f	10001ffffffffffffffssss	a,c
mov f, Wd	10000ffffffffffffffdddd	a,c
mov WREG, f	10110111101fffffffffffff	b
mov f, WREG	10111111100fffffffffffff	b
mov #lit16, Wd	0010nnnnnnnnnnnnnndddd	a,d

Notes

a) Registers are encoded as for bit values using the appropriate letters from the register name. For example, the bits dddd are the binary encoding of the d in Rd.
b) File registers are encoded as their lower 13 bits. Only file registers from 0 to 8191 (0x0000 to 0x1000) can be used.
c) File registers are encoded as 15 bits. Because file registers must be even, the lowest bit is omitted. Only even file registers from 0 to 65534 (0x0000 to 0xFFFE) can be used.
d) Literal values are encoded with the number of binary digits needed to encode their range.

Useful file registers - - names and addresses

name	address (in hex)
WREG0	0000
WREG15	001E
SPLIM	0020
PCL	002E
PCH	0030
SR	0042
TRISB	02C8
PORTB	02CA
LATB	02CC
ODCB	02CE