

Contents

TechTools Instruction Set 2

Appendix A

TechTools Instruction Set

ADD	fr,#lit	CSNE	fr,#lit	MOVB	bit1,/bit2
ADD	fr1,fr2	CSNE	fr1,fr2	MOVSZ	W,++fr
ADD	fr,W	DEC	fr	MOVSZ	W,--fr
ADD	W,fr	DECSZ	fr	NOP	
ADDB*	fr,bit	DJNZ	fr,addr9	NOT	fr
AND	fr,#lit	IJNZ	fr,addr9	NOT	W
AND	fr1,fr2	INC	fr	OR	fr,#lit
AND	fr,W	INCSZ	fr	OR	fr1,fr2
AND	W,#lit	JB	bit,addr9	OR	fr,W
AND	W,fr	JC	addr9	OR	W,#lit
CALL	addr8	JMP	addr9	OR	W,fr
CJA	fr,#lit,addr9	JMP	PC+W	RET	
CJA	fr1,fr2,addr9	JMP	W	RETW	lit,lit,...
CJAE	fr,#lit,addr9	JNB	bit,addr9	RL	fr
CJAE	fr1,fr2,addr9	JNC	addr9	RR	fr
CJB	fr,#lit,addr9	JNZ	addr9	SB	bit
CJB	fr1,fr2,addr9	JZ	addr9	SC	
CJBE	fr,#lit,addr9	LCALL	addr11	SETB	bit
CJBE	fr1,fr2,addr9	LJMP	addr11	SKIP	
CJE	fr,#lit,addr9	LSET	addr11	SLEEP	
CJE	fr1,fr2,addr9	MOV	fr,#lit	SNB	bit
CJNE	fr,#lit,addr9	MOV	fr1,fr2	SNC	
CJNE	fr1,fr2,addr9	MOV	fr,W	SNZ	
CLC		MOV	OPTION,#lit	STC	
CLR	fr	MOV	OPTION,fr	STZ	
CLR	W	MOV	OPTION,W	SUB	fr,#lit
CLR	WDT	MOV	!port_fr,#lit	SUB	fr1,fr2
CLR ^B	bit	MOV	!port_fr,fr	SUB	fr,W
CLZ		MOV	!port_fr,W	SUBB*	fr,bit
CSA	fr,#lit	MOV	W,#lit	SWAP	fr
CSA	fr1,fr2	MOV	W,fr	SZ	
CSAE	fr,#lit	MOV	W,/fr	TEST	fr
CSAE	fr1,fr2	MOV	W,fr-W	XOR	fr,#lit
CSB	fr,#lit	MOV	W,++fr	XOR	fr1,fr2
CSB	fr1,fr2	MOV	W,--fr	XOR	fr,W
CSBE	fr,#lit	MOV	W,<<fr	XOR	W,#lit
CSBE	fr1,fr2	MOV	W,>>fr	XOR	W,fr
CSE	fr,#lit	MOV	W,<>fr		
CSE	fr1,fr2	MOVB	bit1,bit2		

* These instructions are only available on 16C5x parts.

ADD fr,#literal Add literal into fr

Words: 2 Cycles: 2 Affects: W, C, DC, Z

Operation: Literal is added into fr via W. C will be set if an overflow occurs; otherwise, C will be cleared. DC will be set or cleared depending on whether or not an overflow occurs in the lower nibble. Z will be set if the result is 0; otherwise, Z will be cleared. W is left holding the literal value.

Coding: MOV W,#lit (MOVLW lit)
 ADD fr,W (ADDWF fr,1)

A

ADD fr1,fr2 Add fr2 into fr1

Words: 2 Cycles: 2 Affects: W, C, DC, Z

Operation: Fr2 is added into fr1 via W. C will be set if an overflow occurs; otherwise, C will be cleared. DC will be set or cleared depending on whether or not an overflow occurs in the lower nibble. Z will be set if the result is 0; otherwise, Z will be cleared. W is left holding the contents of frb.

Coding: MOV W,fr2 (MOVF fr2,0)
 ADD fr1,W (ADDWF fr1,1)

ADD fr,W Add W into fr

Words: 1 Cycles: 1 Affects: C, DC, Z

Operation: W is added into fr. C will be set if an overflow occurs, otherwise C will be cleared. DC will be set or cleared depending on whether or not an overflow occurs in the lower nibble. Z will be set if the result is 0, otherwise Z will be cleared.

Coding: ADD fr,W (ADDWF fr,1)

ADD W,fr Add fr into W

Words: 1 Cycles: 1 Affects: C, DC, Z

Operation: Fr is added into W. C will be set if an overflow occurs, otherwise C will be cleared. DC will be set or cleared depending on whether or not an overflow occurs in the lower nibble. Z will be set if the result is 0, otherwise Z will be cleared.

Coding: ADD W,fr (ADDWF fr,0)

ADDB fr,bit Add bit into fr

Words: 2 Cycles: 2 Affects: Z

Operation: If bit is set, fr is incremented. If fr is incremented, Z will be set if the result is 0; otherwise, Z will be cleared. This instruction is useful for adding the carry into the upper byte of a double-byte sum after the lower byte has been computed.

Coding: SNC bit (BTFSC bit)
 INC fr (INCF fr,1)

This instruction is only available for 16C5x parts.

AND fr,#literal AND literal into fr

Words: 2 Cycles: 2 Affects: W, Z

Operation: Literal is AND'd into fr via W. Z will be set if the result is 0; otherwise, Z will be cleared.

Coding: MOV W,#lit (MOVLW lit)
 AND fr,W (ANDWF fr,1)

AND fr1,fr2 AND fr2 into fr1

Words: 2 Cycles: 2 Affects: W, Z

Operation: Fr2 is AND'd into fr1 via W. Z will be set if the result is 0; otherwise, Z will be cleared.

Coding: MOV W,fr2 (MOVF fr2,0)
 AND fr1,W (ANDWF fr1,1)

A

AND fr,W AND W into fr

Words: 1 Cycles: 1 Affects: Z

Operation: W is AND'd into fr. Z will be set if the result is 0; otherwise, Z will be cleared.

Coding: AND fr,W (ANDWF fr,1)

AND W,#literal AND literal into W

Words: 1 Cycles: 1 Affects: Z

Operation: Literal is AND'd into W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: ANDLW literal

AND W,fr AND fr into W

Words: 1 Cycles: 1 Affects: Z

Operation: Fr is AND'd into W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: ANDWF fr,0

CALL addr8 Call subroutine

Words: 1 Cycles: 2 Affects: none

Operation: The next instruction address is pushed onto the stack and addr8 is moved to the program counter. The ninth bit of the program counter will be cleared to 0. Therefore, calls are only allowed to the first half of any 512-word page, although the CALL instruction can be anywhere.

Coding: CALL addr8

CJA fr,#literal,addr9 Compare fr to literal and jump if above

Words: 4 Cycles: 4 or 5 (jump) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is greater than literal, a jump to addr9 is executed.

Coding: MOVLW literal^0FFh
 ADDWF fr,0
 BTFSC 3,0
 GOTO addr9

CJA fr1,fr2,addr9 Compare fr1 to fr2 and jump if above

Words: 4 Cycles: 4 or 5 (jump) Affects: W, C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is greater than fr2, a jump to addr9 is executed.

Coding: MOVWF fr1,0
 SUBWF fr2,0
 BTFSS 3,0
 GOTO addr9

A

CJAE fr,#literal,addr9 Compare fr to literal and jump if above or equal

Words: 4 Cycles: 4 or 5 (jump) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is greater than or equal to literal, a jump to addr9 is executed.

Coding: MOVLW literal
 SUBWF fr,0
 BTFSC 3,0
 GOTO addr9

CJAE fr1,fr2,addr9 Compare fr1 to fr2 and jump if above or equal

Words: 4 Cycles: 4 or 5 (jump) Affects: C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is greater than or equal to fr2, a jump to addr9 is executed.

Coding: MOVWF fr2,0
 SUBWF fr1,0
 BTFSC 3,0
 GOTO addr9

CJB fr,#literal,addr9 Compare fr to literal and jump if below

Words: 4 Cycles: 4 or 5 (jump) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is less than literal, a jump to addr9 is executed.

Coding: MOVLW literal
 SUBWF fr,0
 BTFSS 3,0
 GOTO addr9

CJB fr1,fr2,addr9 Compare fr1 to fr2 and jump if below

Words: 4 Cycles: 4 or 5 (jump) Affects: C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is less than fr2, a jump to addr9 is executed.

Coding: MOVF fr2,0
 SUBWF fr1,0
 BTFSS 3,0
 GOTO addr9

CJBE fr,#literal,addr9 Compare fr to literal and jump if below or equal

Words: 4 Cycles: 4 or 5 (jump) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is less than or equal to literal, a jump to addr9 is executed.

Coding: MOVLW literal
 ADDWF fr,0
 BTFSS 3,0
 GOTO addr9

CJBE fr1,fr2,addr9 Compare fr1 to fr2 and jump if below or equal

Words: 4 Cycles: 4 or 5 (jump) Affects: C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is less than or equal to fr2, a jump to addr9 is executed.

Coding: MOVWF fr1,0
 SUBWF fr2,0
 BTFSS 3,0
 GOTO addr9

A

CJE fr,#literal,addr9 Compare fr to literal and jump if equal

Words: 4 Cycles: 4 or 5 (jump) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is equal to literal, a jump to addr9 is executed.

Coding: MOVLW literal
 SUBWF fr,0
 BTFSC 3,2
 GOTO addr9

CJE fr1,fr2,addr9 Compare fr1 to fr2 and jump if equal

Words: 4 Cycles: 4 or 5 (jump) Affects: C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is equal to fr2, a jump to addr9 is executed.

Coding: MOVWF fr2,0
 SUBWF fr1,0
 BTFSC 3,2
 GOTO addr9

CJNE fr,#literal,addr9 Compare fr to literal and jump if not equal

Words: 4 Cycles: 4 or 5 (jump) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is not equal to literal, a jump to addr9 is executed.

Coding: MOVLW literal
 SUBWF fr,0
 BTFSS 3,2
 GOTO addr9

CJNE fr1,fr2,addr9 Compare fr1 to fr2 and jump if not equal

Words: 4 Cycles: 4 or 5 (jump) Affects: C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is not equal to fr2, a jump to addr9 is executed.

Coding: MOVF fr2,0
 SUBWF fr1,0
 BTFSS 3,2
 GOTO addr9

CLC Clear carry

Words: 1 Cycles: 1 Affects: C

Operation: C is cleared to 0.

Coding: BCF 3,0

CLR fr Clear fr

Words: 1 Cycles: 1 Affects: Z

Operation: Fr is cleared to 0. Z is set to 1.

Coding: CLRf fr

A

CLR W Clear W

Words: 1 Cycles: 1 Affects: Z

Operation: W is cleared to 0. Z is set to 1.

Coding: CLRW

CLR WDT Clear the watchdog timer

Words: 1 Cycles: 1 Affects: TO, PD

Operation: The watchdog timer is cleared, along with the prescaler, if assigned. TO and PD are set to 1.

Coding: CLRWDT

TechTools instruction set

CLRB bit Clear bit

Words: 1 Cycles: 1 Affects: none

Operation: Bit is cleared to 0.

Coding: BCF bit

Note: The TechTools assemblers define a bit as port.bitposition, as in the following examples:

RA.3 = bit 3 of port A
PortB.0 = bit 0 of port B

CLZ Clear zero

Words: 1 Cycles: 1 Affects: Z

Operation: Z is cleared to 0.

Coding: BCF 3,2

CSA fr,#literal Compare fr to literal and skip if above

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is greater than literal, the following instruction word is skipped.

Coding: MOVLW literal
ADDWF fr,0
BTFSS 3,0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSA is a single-word instruction.

CSA fr1,fr2 Compare fr1 to fr2 and skip if above

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is greater than fr2, the following instruction word is skipped.

Coding: MOVFB fr1,0
 SUBWFB fr2,0
 BTFSFB 3,0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSA is a single-word instruction.

A

CSAE fr,#literal Compare fr to literal and skip if above or equal

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is greater than or equal to literal, the following instruction word is skipped.

Coding: MOVLW literal
 SUBWFB fr,0
 BTFSFB 3,0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSAE is a single-word instruction.

CSAE fr1,fr2 Compare fr1 to fr2 and skip if above or equal

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is \geq fr2, the following instruction word is skipped.

Coding: MOVFB fr2,0
 SUBWFB fr1,0
 BTFSFB 3,0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSAE is a single-word instruction.

CSB fr,#literal Compare fr to literal and skip if below

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is less than literal, the following instruction word is skipped.

Coding: MOVLW literal
 SUBWF fr,0
 BTFSC 3,0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSB is a single-word instruction.

CSB fr1,fr2 Compare fr1 to fr2 and skip if below

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is less than fr2, the following instruction word is skipped.

Coding: MOVF fr2,0
 SUBWF fr1,0
 BTFSC 3,0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSB is a single-word instruction.

CSBE fr,#literal Compare fr to literal and skip if below or equal

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is less than or equal to literal, the following instruction word is skipped.

Coding: MOVLW literal
 ADDWF fr,0
 BTFSC 3,0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSBE is a single-word instruction.

CSBE fr1,fr2 Compare fr1 to fr2 and skip if below or equal

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is less than or equal to fr2, the following instruction word is skipped.

Coding: MOVF fr1,0
 SUBWF fr2,0
 BTFSS 3,0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSBE is a single-word instruction.

A

CSE fr,#literal Compare fr to literal and skip if equal

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is equal to literal, the following instruction word is skipped.

Coding: MOVLW literal
 SUBWF fr,0
 BTFSS 3,2

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSE is a single-word instruction.

CSE fr1,fr2 Compare fr1 to fr2 and skip if equal

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is equal to fr2, the following instruction word is skipped.

Coding: MOVF fr2,0
 SUBWF fr1,0
 BTFSS 3,2

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSE is a single-word instruction.

CSNE fr,#literal Compare fr to literal and skip if not equal

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr is compared to literal via W. If fr is not equal to literal, the following instruction word is skipped.

Coding: MOV LW literal
 SUBWF fr, 0
 BTFS C 3, 2

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSNE is a single-word instruction.

CSNE fr1,fr2 Compare fr1 to fr2 and skip if not equal

Words: 3 Cycles: 3 or 4 (skip) Affects: C, DC, Z

Operation: Fr1 is compared to fr2 via W. If fr1 is not equal to fr2, the following instruction word is skipped.

Coding: MOV F fr2, 0
 SUBWF fr1, 0
 BTFS C 3, 2

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following CSNE is a single-word instruction.

DEC fr Decrement fr

Words: 1 Cycles: 1 Affects: Z

Operation: Fr is decremented. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: DEC F fr, 1

DECSZ fr Decrement fr and skip if zero

Words: 1 Cycles: 1 or 2 (skip) Affects: none

Operation: Fr is decremented. The next instruction word will be skipped if the result was 0.

Coding: DECFSZ fr, 1

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following DECSZ is a single-word instruction.

A

DJNZ fr,addr9 Decrement fr and jump if not zero

Words: 2 Cycles: 2 or 3 (jump) Affects: none

Operation: Fr is decremented. If the result is not 0, a jump to addr9 is executed.

Coding: DECFSZ fr, 1
 GOTO addr9

IJNZ fr,addr9 Increment fr and jump if not zero

Words: 2 Cycles: 2 or 3 (jump) Affects: none

Operation: Fr is incremented. If the result is not 0, a jump to addr9 is executed.

Coding: INCFSZ fr, 1
 GOTO addr9

TechTools instruction set

INC fr Increment fr

Words: 1 Cycles: 1 Affects: Z

Operation: Fr is incremented. Z will be set if the result was 0, otherwise Z will be cleared.

Coding: INCF fr,1

INCSZ fr Increment fr and skip if zero

Words: 1 Cycles: 1 or 2 (skip) Affects: none

Operation: Fr is incremented. The next instruction word will be skipped if the result was 0.

Coding: INCFSZ fr,1

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following INCSZ is a single-word instruction.

JB bit,addr9 Jump if bit

Words: 2 Cycles: 2 or 3 (jump) Affects: none

Operation: If bit is set, a jump to addr9 is executed.

Coding: BTFSC bit
GOTO addr9

JC addr9 Jump if carry

Words: 2 Cycles: 2 or 3 (jump) Affects: none

Operation: If the carry bit is set, a jump to addr9 is executed.

Coding: BTFSC 3, 0
 GOTO addr9

A

JMP addr9 Jump to address

Words: 1 Cycles: 2 Affects: none

Operation: The lower 9-bits of the literal *addr9* is moved into the program counter.

Coding: GOTO addr9

JMP PC+W Jump to PC+W

Words: 1 Cycles: 2 Affects: C, DC, Z

Operation: W+1 is added into the program counter. The 9th bit of the program counter is always cleared to 0, so the jump destination will be in the first 256 words of any 512-word page. This instruction is useful for jumping into lookup tables comprised of RETW data, or jumping to particular routines. The flags are set as they would be by an ADD instruction.

Coding: ADDWF 2, 1

TechTools instruction set

JMP W Jump to W

Words: 1 Cycles: 2 Affects: none

Operation: W is moved into the program counter. The 9th bit of the program counter is always cleared to 0, so the jump destination will be in the first 256 words of any 512-word page. This instruction is useful for jumping into lookup tables comprised of RETW data, or jumping to particular routines.

Coding: **MOVWF** 2

JNB bit,addr9 Jump if not bit

Words: 2 Cycles: 2 or 3 (jump) Affects: none

Operation: If bit reads 0, a jump to addr9 is executed.

Coding: **BTFSS** bit
 GOTO addr9

Note: The TechTools assemblers define a bit as port.bitposition, as in the following examples:

RA.3 = bit 3 of port A
PortB.0 = bit 0 of port B

JNC addr9 Jump if not carry

Words: 2 Cycles: 2 or 3 (jump) Affects: none

Operation: If C is 0, a jump to addr9 is executed.

Coding: **BTFSS** 3,0
 GOTO addr9

JNZ addr9 Jump if not zero

Words: 2 Cycles: 2 or 3 (jump) Affects: none

Operation: If Z is 0, a jump to addr9 is executed.

Coding: BTFSS 3 , 2
 GOTO addr9

A

JZ addr9 Jump if zero

Words: 2 Cycles: 2 or 3 (jump) Affects: none

Operation: If Z is 1, a jump to addr9 is executed.

Coding: BTFSC 3 , 2
 GOTO addr9

LCALL addr11 Long call

Words: 1-3 Cycles: 2-4 Affects: none

Operation: Depending on the device size, from zero to two BCF/BSF instructions will be assembled to point the page pre-select bits to addr11's page. The bit set/clear instructions are followed by a CALL to addr11. This instruction is only useful for PICs with more than 512 words.

Coding: (BCF/BSF 3 , x)
 (BCF/BSF 3 , x)
 CALL addr11

Note: Please note that LCALL does not set the page select bits upon returning to the calling routine. Therefore, your program must set these bits. This can be done using LSET \$, which sets the page select bits to the current page.

LJMP addr11 Long jump

Words: 1-3 Cycles: 2-4 Affects: none

Operation: Depending on the device size, from zero to two BCF/BSF instructions will be assembled to point the page pre-select bits to addr11's page. The bit set/clear instructions are followed by a jump to addr11. This instruction is only useful for PICs with more than 512 words.

Coding: (BCF/BSF 3 , x)
 (BCF/BSF 3 , x)
 GOTO addr11

LSET addr11 Long set

Words: 0-2 Cycles: 0-2 Affects: none

Operation: Depending on the device size, from zero to two BCF/BSF instructions will be assembled to point the page pre-select bits to addr11's page. This instruction is only useful for PICs with more than 512 words.

Coding: (BCF/BSF 3 , x)
 (BCF/BSF 3 , x)

MOV fr,#literal Move literal into fr

Words: 2 Cycles: 2 Affects: none

Operation: Literal is moved into fr via W.

Coding: MOVLW literal
 MOVWF fr

MOV fr1,fr2 Move fr2 into fr1

Words: 2 Cycles: 2 Affects: Z

Operation: Fr2 is moved into fr1 via W. Z will be set to 1 if the value moved was 0, otherwise Z will be cleared to 0.

Coding: MOVF fr2,0
 MOVWF fr1

A

MOV fr,W Move W into fr

Words: 1 Cycles: 1 Affects: none

Operation: W is moved into fr.

Coding: MOVWF fr

MOV [!]OPTION,#literal Move literal into OPTION

Words: 2 Cycles: 2 Affects: none

Operation: Literal is moved into OPTION via W.

Coding: MOVLW literal
 OPTION

Note: On 16C5x parts, you must use the explanation mark (!) before the word OPTION; this causes the assembler to assemble an OPTION instruction. On 16Cxx parts, the mark (!) is optional. Microchip added an OPTION register to the newer parts, so a regular MOV is possible, as long as your program is in the proper bank (usually bank 1). Microchip may remove the OPTION instruction in newer PICs, so they recommend against using the '!'

MOV [!]OPTION,fr Move fr into OPTION

Words: 2 Cycles: 2 Affects: Z

Operation: Fr is moved into OPTION via W. Z will be set to 1 if the value moved was 0, otherwise Z will be cleared to 0.

Coding: MOVF fr,0
 OPTION

Note: On 16C5x parts, you must use the explanation mark (!) before the word OPTION; this causes the assembler to assemble an OPTION instruction. On 16Cxx parts, the mark (!) is optional. Microchip added an OPTION register to the newer parts, so a regular MOV is possible, as long as your program is in the proper bank (usually bank 1). Microchip may remove the OPTION instruction in newer PICs, so they recommend against using the '!'

MOV [!]OPTION,W Move W into OPTION

Words: 1 Cycles: 1 Affects: none

Operation: W is moved into OPTION.

Coding: OPTION

Note: On 16C5x parts, you must use the explanation mark (!) before the word OPTION; this causes the assembler to assemble an OPTION instruction. On 16Cxx parts, the mark (!) is optional. Microchip added an OPTION register to the newer parts, so a regular MOV is possible, as long as your program is in the proper bank (usually bank 1). Microchip may remove the OPTION instruction in newer PICs, so they recommend against using the '!'

MOV !port_fr,#literal Move literal into port_fr's I/O control register

Words: 2 Cycles: 2 Affects: none

Operation: Literal is moved into the I/O control register of port_fr via W. A "1" bit in W disables the corresponding port pin's output buffer, allowing input use, while a "0" bit enables the output buffer for high or low output. Port_fr must be 5, 6, or 7.

Coding: MOVLW literal
 TRIS port_fr

MOV !port_fr,fr Move fr into port_fr's I/O control register

Words: 2 Cycles: 2 Affects: Z

Operation: Fr is moved into the I/O control register of port_fr via W. A "1" bit in W disables the corresponding port pin's output buffer, allowing input use, while a "0" bit enables the output buffer for high or low output. Z will be set to 1 if the value moved was 0, otherwise Z will be cleared to 0. Port_fr must be 5, 6, or 7.

Coding: MOVF fr,0
 TRIS port_fr

A

MOV !port_fr,W Move W into port_fr's I/O control register

Words: 1 Cycles: 1 Affects: none

Operation: W is moved into the I/O control register of port_fr. A "1" bit in W disables the corresponding port pin's output buffer, allowing input use, while a "0" bit enables the output buffer for high or low output. Port_fr must be 5, 6, or 7.

Coding: TRIS port_fr

MOV W,#literal Move literal into W

Words: 1 Cycles: 1 Affects: none

Operation: Literal is moved into W.

Coding: MOVLW literal

MOV W,fr Move fr into W

Words: 1 Cycles: 1 Affects: Z

Operation: Fr is moved into W. Z will be set to 1 if the value moved was 0, otherwise Z will be cleared to 0.

Coding: MOVF fr,0

MOV W,/fr Move not fr into W

Words: 1 Cycles: 1 Affects: Z

Operation: The one's complement of fr is moved into W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: COMF fr,0

MOV W,fr-W Move fr-W into W

Words: 1 Cycles: 1 Affects: C, DC, Z

Operation: W is subtracted from fr and the result is stored in W. C will be cleared to 0 if an underflow occurred, otherwise C will be set to 1. DC will be cleared or set depending on whether or not an underflow occurred in the least-significant nibble. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: SUBWF fr,0

MOV W,++fr Move the incremented value of fr into W

Words: 1 Cycles: 1 Affects: Z

Operation: The incremented value of fr is moved into W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: INCF fr,0

A

MOV W,--fr Move the decremented value of fr into W

Words: 1 Cycles: 1 Affects: Z

Operation: The decremented value of fr is moved into W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: DECF fr,0

MOV W,<<fr Move the left-rotated value of fr into W

Words: 1 Cycles: 1 Affects: C

Operation: The left-rotated value of fr is moved into W. On entry, C must hold the value to be shifted into the least-significant bit of the fr value. On exit, C will hold the previous most-significant bit of the fr value.

Coding: RLF fr,0

MOV W,>>fr Move the right-rotated value of fr into W

Words: 1 Cycles: 1 Affects: C

Operation: The right-rotated value of fr is moved into W. On entry, C must hold the value to be shifted into the most-significant bit of the fr value. On exit, C will hold the previous least-significant bit of the fr value.

Coding: RRF fr,0

MOV W,<>fr Move the nibble-swapped value of fr into W

Words: 1 Cycles: 1 Affects: none

Operation: The nibble-swapped value of fr is moved into W.

Coding: SWAPF fr,0

MOVB bit1,bit2 Move bit2 to bit1

Words: 4 Cycles: 4 Affects: none

Operation: Bit2 is moved to bit1.

Coding: BTFSS bit2
BCF bit1
BTFSC bit2
BSF bit1

Note: The TechTools assemblers define a bit as port.bitposition, as in the following examples:

RA.3 = bit 3 of port A PortB.0 = bit 0 of port B

MOVB bit1,/bit2 Move not bit2 to bit1

Words: 4 Cycles: 4 Affects: none

Operation: The complement of bit2 is moved to bit1.

Coding: BTFSC bit2
 BCF bit1
 BTFSS bit2
 BSF bit1

Note: The TechTools assemblers define a bit as port.bitposition, as in the following examples:

RA.3 = bit 3 of port A PortB.0 = bit 0 of port B

A

MOVSZ W,++fr Move the incremented value of fr into W and skip if zero

Words: 1 Cycles: 1 or 2 (skip) Affects: none

Operation: The incremented value of fr is moved into W. The next instruction word will be skipped if the result was 0.

Coding: INCFSZ fr,0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following MOVSZ is a single-word instruction.

MOVSZ W,--fr Move the decremented value of fr into W and skip if zero

Words: 1 Cycles: 1 or 2 (skip) Affects: none

Operation: The decremented value of fr is moved into W. The next instruction word will be skipped if the result was 0.

Coding: DECFSZ fr,0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following MOVSZ is a single-word instruction.

TechTools instruction set

NOP **No operation**

Words: 1 Cycles: 1 Affects: none

Operation: none

Coding: NOP

NOT **fr** **Not fr**

Words: 1 Cycles: 1 Affects: Z

Operation: Fr is converted into its one's complement value. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: COMF fr,1

NOT **W** **Not W**

Words: 1 Cycles: 1 Affects: Z

Operation: W is converted into its one's complement value. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: XORLW 0FFh

OR fr,#literal OR literal into fr

Words: 2 Cycles: 2 Affects: Z

Operation: Literal is OR'd into fr via W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: MOVLW literal
 IORWF fr,1

A

OR fr1,fr2 OR fr2 into fr1

Words: 2 Cycles: 2 Affects: Z

Operation: Fr2 is OR'd into fr1 via W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: MOVF fr2,0
 IORWF fr1,1

OR fr,W OR W into fr

Words: 1 Cycles: 1 Affects: Z

Operation: W is OR'd into fr. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: IORWF fr,1

OR W,#literal OR literal into W

Words: 1 Cycles: 1 Affects: Z

Operation: Literal is OR'd into W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: IORLW literal

OR W,fr OR fr into W

Words: 1 Cycles: 1 Affects: Z

Operation: Fr is OR'd into W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: IORWF fr,0

RET Return from subroutine

Words: 1 Cycles: 2 Affects: none

Operation: The next stack value is moved into the program counter. W is cleared to 0.

Coding: RETLW 0

TechTools instruction set

SB **bit** **Skip if bit**

Words: 1 Cycles: 1 or 2 (skip) Affects: none

Operation: If bit reads 1, the following instruction word is skipped.

Coding: BTFSS bit

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following SB is a single-word instruction.

Note: The TechTools assemblers define a bit as port.bitposition, as in the following examples:

RA.3 = bit 3 of port A PortB.0 = bit 0 of port B

SC **Skip if carry**

Words: 1 Cycles: 1 or 2 (skip) Affects: none

Operation: If C is 1, the following instruction word is skipped.

Coding: BTFSS 3, 0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following SC is a single-word instruction.

SETB **bit** **Set bit**

Words: 1 Cycles: 1 Affects: none

Operation: Bit is set to 1.

Coding: BSF bit

Note: The TechTools assemblers define a bit as port.bitposition, as in the following examples:

RA.3 = bit 3 of port A PortB.0 = bit 0 of port B

SKIP **Skip the following instruction word**

Words: 1 Cycles: 2 Affects: none

Operation: The following instruction word is skipped.

Coding: BTFSS 4, 7

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following SKIP is a single-word instruction.

A

SLEEP **Enter sleep mode**

Words: 1 Cycles: 1 Affects: TO, PD

Operation: The watchdog timer is cleared and the oscillator is stopped. TO is set to 1. PD is cleared to 0.

Coding: SLEEP

SNB bit **Skip if not bit**

Words: 1 Cycles: 1 or 2 (skip) Affects: none

Operation: If bit reads 0, the following instruction word is skipped.

Coding: BTFSC bit

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following SNB is a single-word instruction.

Note: The TechTools assemblers define a bit as port.bitposition, as in the following examples:

RA.3 = bit 3 of port A PortB.0 = bit 0 of port B

TechTools instruction set

SNC **Skip if not carry**

Words: 1 Cycles: 1 or 2 (skip) Affects: none

Operation: If C is 0, the following instruction word is skipped.

Coding: BTFSC 3, 0

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following SNC is a single-word instruction.

SNZ **Skip if not zero**

Words: 1 Cycles: 1 or 2 (skip) Affects: none

Operation: If Z is 0, the following instruction word is skipped.

Coding: BTFSC 3, 2

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following SNZ is a single-word instruction.

STC **Set carry**

Words: 1 Cycles: 1 Affects: C

Operation: C is set to 1.

Coding: BSF 3, 0

STZ **Set zero flag**

Words: 1 Cycles: 1 Affects: Z

Operation: Z is set to 1.

Coding: BSF 3,2

A

SUB fr,#literal **Subtract literal from fr**

Words: 2 Cycles: 2 Affects: C, DC, Z

Operation: Literal is subtracted from fr via W. C will be cleared to 0 if an underflow occurred, otherwise C will be set to 1. DC will be cleared or set, depending on whether or not an underflow occurred in the least-significant nibble. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: MOVLW literal
SUBWF fr,1

SUB fr1,fr2 **Subtract fr2 from fr1**

Words: 2 Cycles: 2 Affects: C, DC, Z

Operation: Fr2 is subtracted from fr1 via W. C will be cleared to 0 if an underflow occurred, otherwise C will be set to 1. DC will be cleared or set, depending on whether or not an underflow occurred in the least-significant nibble. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: MOVF fr2,0
SUBWF fr1,1

SUB fr,W Subtract W from fr

Words: 1 Cycles: 1 Affects: C, DC, Z

Operation: W is subtracted from fr. C will be cleared to 0 if an underflow occurred, otherwise C will be set to 1. DC will be cleared or set, depending on whether or not an underflow occurred in the least-significant nibble. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: SUBWF fr,1

SUBB fr,bit Subtract bit from fr

Words: 2 Cycles: 2 Affects: Z

Operation: If bit reads 0, fr is decremented. If fr was decremented, Z will be set to 1 if the result was 0, else Z will be cleared to 0. This instruction is useful for subtracting the carry from the upper byte of a double-byte value after the lower byte has been subtracted.

Coding: BTFSS 3,0
DECF fr,1

Note: The TechTools assemblers define a bit as port.bitposition, as in the following examples:

RA.3 = bit 3 of port A PortB.0 = bit 0 of port B

Note: This instruction only works on 16C5x parts.

SWAP fr Swap nibbles in fr

Words: 1 Cycles: 1 Affects: none

Operation: The high- and low-order nibbles in fr are swapped.

Coding: SWAPF fr,1

SZ **Skip if zero**

Words: 1 Cycles: 1 or 2 (skip) Affects: none

Operation: If Z is 1, the following instruction word is skipped.

Coding: BTFSS 3, 2

Note: Only one word is skipped by this instruction. To avoid strange results, make sure that any instruction following SZ is a single-word instruction.

A

TEST fr **Test fr for zero**

Words: 1 Cycles: 1 Affects: Z

Operation: Fr is read and copied back to itself. Z will be set to 1 if the value moved was 0, otherwise Z will be cleared to 0.

Coding: MOVF fr, 1

TEST W **Test W for zero**

Words: 1 Cycles: 1 Affects: Z

Operation: Z will be set to 1 if W is 0, otherwise Z will be cleared to 0.

Coding: IORLW 0

XOR fr,#literal XOR literal into fr

Words: 2 Cycles: 2 Affects: Z

Operation: Literal is XOR'd into fr via W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: `MOVLW literal`
`XORWF fr,1`

XOR fr1,fr2 XOR fr2 into fr1

Words: 2 Cycles: 2 Affects: Z

Operation: Fr2 is XOR'd into fr1 via W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: `MOVF fr2,0`
`XORWF fr1,1`

XOR fr,W XOR W into fr

Words: 1 Cycles: 1 Affects: Z

Operation: W is XOR'd into fr. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: `XORWF fr,1`

XOR W,#literal XOR literal into W

Words: 1 Cycles: 1 Affects: Z

Operation: Literal is XOR'd into W. Z will be set if the result was 0, otherwise Z will be cleared.

Coding: XORLW literal

A

XOR W,fr XOR fr into W

Words: 1 Cycles: 1 Affects: Z

Operation: Fr is XOR'd into W. Z will be set to 1 if the result was 0, otherwise Z will be cleared to 0.

Coding: XORWF fr,0

Index

A

ADD	fr,#lit	2	
ADD	fr,#literal		Add literal into fr 3
ADD	fr1,fr2		Add fr2 into fr1 3
ADD	fr,W		Add W into fr 3
ADD	W,fr		Add fr into W 4
ADDB	fr,bit		Add bit into fr 4
AND	fr,#literal		AND literal into fr 4
AND	fr1,fr2		AND fr2 into fr1 5
AND	fr,W		AND W into fr 5
AND	W,#literal		AND literal into W 5
AND	W,fr		AND fr into W 6

C

CALL	addr8		Call subroutine 6
CJA	fr,#literal,addr9		Compare fr to literal and 6
CJA	fr1,fr2,addr9		Compare fr1 to fr2 and jump if 7
CJAE	fr,#literal,addr9		Compare fr to literal a 7
CJAE	fr1,fr2,addr9		Compare fr1 to fr2 and jump i 7
CJB	fr,#literal,addr9		Compare fr to literal and 8
CJB	fr1,fr2,addr9		Compare fr1 to fr2 and jump 8
CJBE	fr,#literal,addr9		Compare fr to li 8
CJBE	fr1,fr2,addr9		Compare fr1 to fr2 and jump i 9
CJE	fr,#literal,addr9		Compare fr to literal and 9
CJE	fr1,fr2,addr9		Compare fr1 to fr2 and jump if 9
CJNE	fr,#literal,addr9		Compare fr to literal and 10
CJNE	fr1,fr2,addr9		Compare fr1 to fr2 and jump i 10
CLC			Clear carry 10
CLR	fr		Clear fr 11
CLR	W		Clear W 11
CLR	WDT		Clear the watchdog timer 11
CLRB	bit		Clear bit 12
CLZ			Clear zero 12

CSA	fr,#literal	Compare fr to literal and skip i	12
CSA	fr1,fr2	Compare fr1 to fr2 and skip if above	13
CSAE	fr,#literal	Compare fr to literal and skip	13
CSAE	fr1,fr2	Compare fr1 to fr2 and skip if above	13
CSB	fr,#literal	Compare fr to literal and skip i	14
CSB	fr1,fr2	Compare fr1 to fr2 and skip if below	14
CSBE	fr,#literal	Compare fr to literal and skip	14
CSBE	fr1,fr2	Compare fr1 to fr2 and skip if below	15
CSE	fr,#literal	Compare fr to literal and skip i	15
CSE	fr1,fr2	Compare fr1 to fr2 and skip if equal	15
CSNE	fr,#literal	Compare fr to literal and skip	16
CSNE	fr1,fr2	Compare fr1 to fr2 and skip if not	16

D

DEC	fr	Decrement fr	16
DECSZ	fr	Decrement fr and skip if zero	17
DJNZ	fr,addr9	Decrement fr and jump if not zero	17

I

IJNZ	fr,addr9	Increment fr and jump if not zero	17
INC	fr	Increment fr	18
INCSZ	fr	Increment fr and skip if zero	18

J

JB	bit,addr9	Jump if bit	18
JC	addr9	Jump if carry	19
JMP	addr9	Jump to address	19
JMP	PC+W	Jump to PC+W	19
JMP	W	Jump to W	20
JNB	bit,addr9	Jump if not bit	20
JNC	addr9	Jump if not carry	20
JNZ	addr9	Jump if not zero	21
JZ	addr9	Jump if zero	21

L

LCALL	addr11	Long call	21
LJMP	addr11	Long jump	22
LSET	addr11	Long set	22

Index

M

Microchip 23, 24
MOV !port_fr,#literal Move literal into port_f 24
MOV !port_fr,fr Move fr into port_fr's I/O contr 25
MOV !port_fr,W Move W into port_fr's I/O control 25
MOV [!]OPTION,#literal Move literal into OPTION 23
MOV [!]OPTION,fr Move fr into OPTION 24
MOV [!]OPTION,W Move W into OPTION 24
MOV fr,#literal Move literal into fr 22
MOV fr1,fr2 Move fr2 into fr1 23
MOV fr,W Move W into fr 23
MOV W,#literal Move literal into W 25
MOV W,++fr Move the incremented value of fr into 27
MOV W,--fr Move the decremented value of fr in 27
MOV W,/fr Move not fr into W 26
MOV W,<<fr Move the left-rotated value of fr int 27
MOV W,<>fr Move the nibble-swapped value of fr i 28
MOV W,>>fr Move the right-rotated value of fr in 28
MOV W,fr Move fr into W 26
MOV W,fr-W Move fr-W into W 26
MOVB bit1,/bit2 Move not bit2 to bit1 29
MOVB bit1,bit2 Move bit2 to bit1 28
MOVSZ W,++fr Move the incremented value of fr 29
MOVSZ W,--fr Move the decremented value of 29

N

NOP No operation 30
NOT fr Not fr 30
NOT W Not W 30

O

OR fr,#literal OR literal into fr 31
OR fr1,fr2 OR fr2 into fr1 31
OR fr,W OR W into fr 31
OR W,#literal OR literal into W 32
OR W,fr OR fr into W 32
Oscillator 35

P

Program counter 6, 19, 20, 32

R

RET Return from subroutine 32
 RETW literal1,literal2,... Assemble RET's wh 33
 RL fr Rotate fr left 33
 RR fr Rotate fr right 33

S

SB bit Skip if bit 34
 SC Skip if carry 34
 SETB bit Set bit 34
 SKIP Skip the following instruction word 35
 SLEEP Enter sleep mode 35
 SNB bit Skip if not bit 35
 SNC Skip if not carry 36
 SNZ Skip if not zero 36
 STC Set carry 36
 STZ Set zero flag 37
 SUB fr,#literal Subtract literal from fr 37
 SUB fr1,fr2 Subtract fr2 from fr1 37
 SUB fr,W Subtract W from fr 38
 SUBB fr,bit Subtract bit from fr 38
 SWAP fr Swap nibbles in fr 38
 SZ Skip if zero 39

T

TEST fr Test fr for zero 39
 TEST W Test W for zero 39

X

XOR fr,#literal XOR literal into fr 40
 XOR fr1,fr2 XOR fr2 into fr1 40
 XOR fr,W XOR W into fr 40
 XOR W,#literal XOR literal into W 41
 XOR W,fr XOR fr into W 41