PIC18 e assembly: le prime istruzioni

TABLE 25-2: PIC18(L)F2X/4XK22 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and CARRY bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 1111	ffff ffff	ffff ffff	ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	1, 2
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	1, 2
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	1, 2
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	

Il PIC18 è un processore RISC che può eseguire 75 istruzioni diverse. In questa pagina vedremo come interpretare le informazioni contenute nei fogli tecnici che descrivono un'istruzione. In genere è poco utile imparare a memoria tutte le istruzioni: è sufficiente conosce quelle usate più frequentemente e, soprattutto, imparare a leggere i fogli tecnici che le descrivono.

L'istruzione

Come primo esempio utilizzeremo l'istruzione `movlw 0x55` della quale già abbiamo visto il funzionamento.
Le singole istruzioni del microcontrollore sono tutte descritte nella parte terminale dei fogli tecnici.

La struttura usata nella descrizione è simile per tutte le istruzioni:

MOVLW **Move literal to W**

Syntax:	MOVLW k
Operands:	$0 \leq k \leq 255$
Operation:	$k \rightarrow W$
Status Affected:	None
Encoding:	0000 1110 kkkk kkkk
Description:	The 8-bit literal 'k' is loaded into W.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `MOVLW 5Ah`

After Instruction

W = 5Ah

Esaminiamo quanto viene scritto:

- Il nome dell'istruzione cerca di richiamare il suo comportamento: **MOV**e **L**itterlal to **W** (muovi letteralmente nel registro W). In genere viene chiamato *codice mnemonico*

- **Syntax** mostra come l'istruzione dovrà apparire nel codice sorgente che andremo a scrivere, inclusi eventuali argomenti (se opzionali tra `{}`). Il nome delle istruzioni può essere sia maiuscolo che minuscolo (*case insensitive*)
- **Operands** indica che l'unico (in questo caso) operando k dovrà essere un numero compreso tra 0 e 255, cosa ovvia visto che W è un registro a 8 bit
- **Operation** descrive simbolicamente l'operazione eseguita. In particolare "→" indica che un numero viene copiato in un registro
- **Status Affected** indica l'elenco dei flags modificati (in questo caso nessuno)
- **Encoding** mostra la sequenza binaria corrispondente all'istruzione.
- **Description** ha un ovvio significato...
- **Words** lo spazio occupato in memoria dall'OpCode, misurato in parole da 16 bit
- **Cycles** indica il tempo necessario per l'esecuzione dell'istruzione. La "strana" unità di misura è 4 periodi di clock del processore (nell'esempio, con clock a 1 MHz: 4µs)
- **Q Cycles Activity** mostra le quattro fasi che, in questo caso, sono necessarie per completare l'istruzione.
- Infine, un esempio d'uso

Qualche osservazione secondaria:

- Per quasi tutte le istruzioni del PIC18, i campi *Word* e *Cycles* coincidono e sono pari a 1.
- Nella Program Memory abbiamo visto che l'istruzione `movlw 0x55` è stata codificata come `0x0E55`. Ricostruiamo a mano l'istruzione, facendo il lavoro dell'assemblatore:
  - Il primo byte è quello riportato nel foglio tecnico: `0000 1110` (=`0x0E`)
  - Il secondo byte (`0x55`) è il numero da caricare nel registro `W`, indicato nel foglio tecnico con la lettera k

Gli Status Bits

Gli *Status Bits* sono una serie di 5 flag che identificano particolari situazioni generate dalle operazioni aritmetiche, logiche o di altro tipo. Ciascun *flag* è una variabile booleana e vale 0 oppure 1 (o anche: Vero oppure Falso) ed è indicato da una lettera:

- Carry: potrebbe essere visto come il nono bit del risultato di una operazione aritmetica a otto bit. Può indicare, a seconda dell'operazione, il riporto delle operazioni di somma e sottrazione

oppure viene usato in alcune operazioni di rotazione o shift (scorrimento dei bit a destra o a sinistra).
Ad esempio sommando i numeri binari a otto bit 0000_0001 e 1111_1111, il risultato è 1_0000_0000, cioè 0000_0000 con il riporto di 1. Tale riporto verrà memorizzato nel flag C.

- (in genere poco rilevante) **D**igit **C**arry: è simile al carry, ma interviene come riporto tra il 4 bit che formano un byte. Viene utilizzato per le operazioni in aritmetica BCD.
- **Z**ero: indica che un risultato di una operazione è zero
- **OV**erflow: indica un riporto tra il sesto ed il settimo bit; è utilizzato nelle operazioni in complemento a due, cioè per la gestione dei numeri negativi
- **N**egative: indica il valore del settimo bit, cioè il segno di un numero in complemento a due

In genere i flags sono utilizzati nelle istruzioni di salto condizionato, cioè per eseguire alcune istruzioni piuttosto che altre a seconda del risultato di una operazione

Usando il simulatore. il valore dei **flags** è mostrato accanto al valore del PC e del registro W, usando una lettera maiuscola oppure minuscola. Ad esempio:

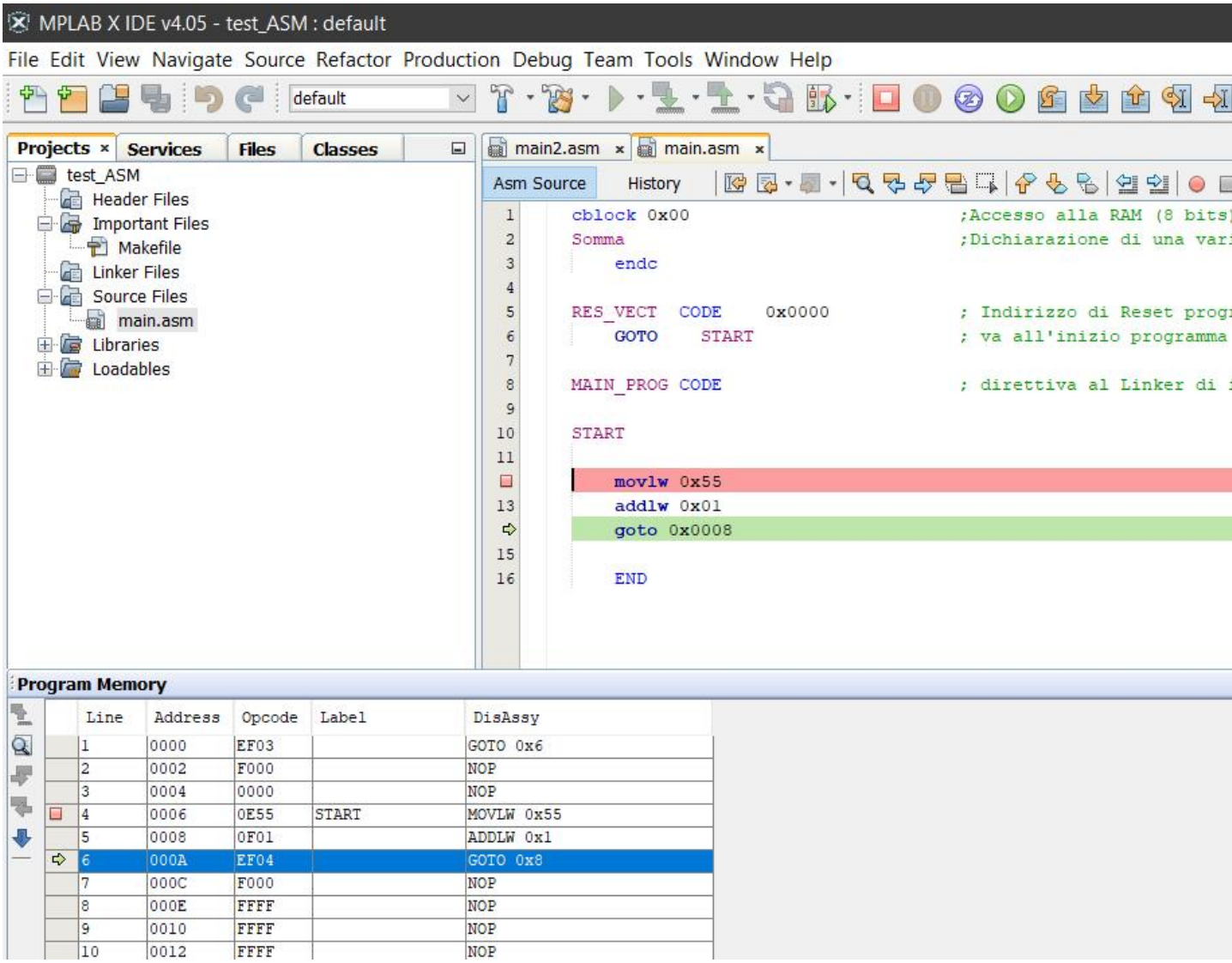
- **z** (come nell'esempio mostrato) significa che il risultato dell'ultima operazione eseguita non è zero
- **Z** significa che il risultato è zero
- **c** (come nell'esempio mostrato) significa che l'ultima operazione eseguita non ha prodotto riporto
- **C** significa che è presente un riporto
- ...

I salti

Il primo progamma visto è formato da istruzioni eseguite in ordine sequenziale, cioè ordinatamente una dopo l'altra. Il PIC possiede varie istruzioni che permettono di eseguire le istruzioni non in ordine sequenziale: qui ne vedremo tre, molto simili ed indicate con il nome generico di *salto*.

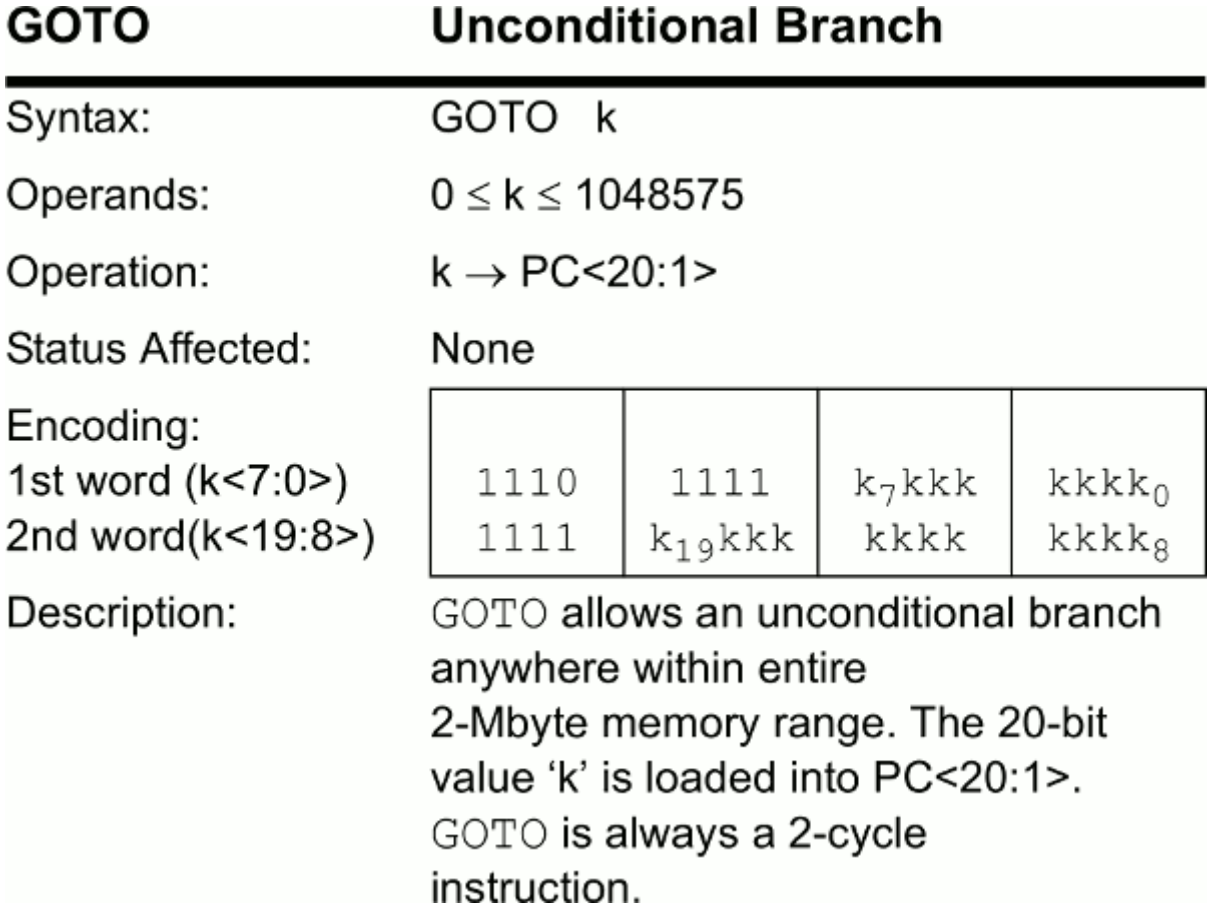
Salto incondizionato

Questa istruzione permette di eseguire una istruzione posta in una qualunque cella della memoria programma, *saltando* in avanti o indietro all'interno del programma.



Per esempio il programma appena mostrato, dopo l'esecuzione dell'istruzione `goto 0x00008` eseguirà l'istruzione posta all'indirizzo di memoria `0x00008` (cioè `addlw 0x01`), all'interno di un ciclo infinito di incremento del contenuto di `WREG`. Prima di proseguire, eseguire passo passo il programma.

Facciamo riferimento al foglio tecnico:



GOTO	Unconditional Branch			
Syntax:	GOTO k			
Operands:	$0 \le k \le 1048575$			
Operation:	$k \rightarrow PC<20:1>$			
Status Affected:	None			
Encoding: 1st word (k<7:0>) 2nd word(k<19:8>)	1110 1111	1111 $k_{19}kkk$	k_7kkk kkkk	$kkkk_0$ $kkkk_8$
Description:	GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a 2-cycle instruction.			

Come si legge nella descrizione il Program Counter assume un valore qualsiasi di 20 bit (5 cifre esadecimali), permettendo di eseguire una qualunque istruzione.

Il calcolo dell'indirizzo della cella in cui è contenuta l'istruzione che si vuole eseguire (`0x00008` nell'esempio) è piuttosto noioso e deve essere rifatto ogni volta che aggiunge o toglie un'istruzione. Per fortuna può essere calcolato automaticamente dall'assemblatore usando **labels**

Branch (diramazione) incondizionato

L'istruzione `bra`, a partire dal nome, è molto simile alla precedente. Essa somma (o sottrae) al valore del Program Counter un numero compreso tra -1024 e 1023, permettendo al programma di andare avanti (o indietro) saltando fino ad un massimo di circa 1000 istruzioni rispetto all'istruzione stessa di `bra`.

BRA	Unconditional Branch			
Syntax:	BRA n			
Operands:	$-1024 \leq n \leq 1023$			
Operation:	$(PC) + 2 + 2n \rightarrow PC$			
Status Affected:	None			
Encoding:	1101	0nnn	nnnn	nnnn
Description:	Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a 2-cycle instruction.			
Words:	1			
Cycles:	2			

A volte questo tipo di salto è indicato come *salto relativo*, mentre il `goto` è indicato come *salto assoluto*.

Branch condizionati

I salti condizionati permettono di eseguire un'istruzione piuttosto che un'altra a seconda del valore di uno dei flag. Sono la base dei costrutti che ad alto livello vengono indicati come `if-then-else` oppure cicli `while-do` e `for`.

TABLE 25-2: PIC18(L)F2X/4XK22 INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BIT-ORIENTED OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	