

## INTRODUZIONE ALLA PROGRAMMAZIONE DEI PIC16XXX

Queste pagine forniscono alcune nozioni base riguardo alla programmazione in linguaggio macchina dei PIC, i microcontrollori prodotti dalla [Microchip](#).

### **microprocessore e microcontrollore.**

Il primo è un componente che ha bisogno di numerosi integrati esterni aggiuntivi per poter funzionare (memoria, oscillatore di clock, periferiche di ingresso/uscita ecc) e può diventare l'elemento di controllo di un computer molto sofisticato.

**Un microcontrollore** invece racchiude tutti questi elementi all'interno di un unico piccolo contenitore, e ha bisogno di pochissimi componenti esterni per funzionare.

La memoria per il programma, la memoria RAM per i dati di lavoro), l'oscillatore, il circuito di reset e le diverse periferiche, sono tutte racchiuse in un unico chip.

Rispetto ai **microprocessori** le capacità di calcolo sono estremamente ridotte, la memoria RAM per esempio è formata da un numero ridotto di celle. ( solitamente non è espandibile in alcun modo). I microprocessori possono essere usati per effettuare elaborazioni molto complesse su grandi quantità di dati, i microcontrollori sono invece adatti per compiti di controllo hardware a basso livello, che non richiedono grandi quantità di memoria, ma prezzo, consumo e dimensioni ridotti, uniti ad una discreta velocità di esecuzione e ad una nutrita serie di periferiche già pronte come timers, convertitori analogico/digitali (ADC), generatori PWM, porte seriali ecc.

Tipiche applicazioni di un microcontroller possono essere automatismi, antifurti, strumenti di misura, regolazione luminosità, caricabatterie, trasmettitori/ricevitori codificati ecc.

Cos'è il linguaggio macchina?

Ogni microprocessore è progettato e costruito per eseguire determinate operazioni binarie che vengono lette una dopo l'altra da un'apposita area di memoria, detta memoria programma.

Queste sequenze binarie sono le istruzioni in linguaggio macchina (L.M.), istruzioni direttamente comprensibile dai circuiti del micro.

Una tipica istruzione in linguaggio macchina è la sequenza:

**01010100000011**

Ad ogni istruzione L.M. è stato dato un nome simbolico detto codice mnemonico. L'insieme dei codici mnemonici prende il nome di linguaggio assembler (o assembly). Il programma in assembler si scrive con un qualsiasi editor di testo.

Tramite il programma assembler si convertono poi le istruzioni assembler in codice macchina direttamente eseguibile.

Per ogni istruzione assembler esiste naturalmente una e una sola sequenza binaria in codice macchina.

### **Il motivo perché studiare il linguaggio macchina?**

Un programma scritto in L.M. è molto compatto:

Usa poca memoria, è veloce nell'esecuzione, può accedere completamente alle risorse fornite dal micro e ottenere il controllo assoluto delle sue temporizzazioni. Lavora in stretto contatto con l'hardware...

svantaggi : i programmi sono più difficili da scrivere, interpretare e correggere, ci si impiega molto più tempo per scriverli, richiedono molte istruzioni anche per effettuare operazioni semplici, ed è molto difficoltoso effettuare calcoli.

Ogni micro dispone di un set ben definito di istruzioni elementari, che sono naturalmente dedicate a quel singolo micro, pertanto un programma assembler scritto per un micro non è compatibile con un programma assembler per PIC o per altro processore.

Essendo l'assembler strettamente legato all'hardware, le sue istruzioni risentono dei limiti (o delle potenzialità) offerte dall'hardware stesso. Una minima conoscenza dell'hardware è quindi indispensabile. I PIC di cui si parla qui sono micro con architettura a 8 bit, questo significa che possono lavorare direttamente solo con numeri rappresentabili in 8 bit (valori compresi tra 0 e 255). Hanno una memoria per le istruzioni del programma separata dalla memoria dati (RAM). La prima è di tipo flash, riprogrammabile elettricamente almeno 1000 volte con un apposito programmatore.

La RAM invece contiene tutte le informazioni di lavoro necessarie durante l'esecuzione del programma, ma a differenza della prima si cancella ogni volta che viene tolta l'alimentazione. In questi micro ogni cella (o locazione) della RAM può essere pensata (ed effettivamente usata) come un registro a 8 bit in cui salvare o da cui leggere i nostri dati.

**Le locazioni di memoria, sia programma che dati, hanno un indirizzo crescente che parte da 0. In particolare le prime locazioni della RAM prendono il nome di e SFR (special function registers), e servono per controllare il funzionamento dell'hardware.**

In quest'area troviamo per esempio i registri che permettono l'accesso ai piedini (pin) del microcontrollore, che consentono cioè al programma di generare una tensione (livello logico) verso l'esterno per comandare ad esempio :

display, relè, transistor ecc..., oppure di leggerla, per esempio per determinare la posizione di un interruttore o di un pulsante.

Attraverso i registri della sezione SFR si possono anche attivare e usare le diverse periferiche interne, come i timers, il convertitore analogico/digitale (ADC), la memoria EEPROM ecc...

Le aree di memoria su cui si può agire da programma sono i registri della memoria dati e il registro accumulatore W, che non fa parte dell'area dati ma è un ulteriore registro hardware specializzato, usato nelle operazioni aritmetico logiche.

La RAM è inoltre suddivisa in due o più banchi, un po' come se vi fossero più RAM attivabili solo uno alla volta, questa selezione si ottiene impostando alcuni bit specifici (nel registro STATUS).

Durante l'esecuzione di un programma è importante sapere sempre quale banco si sta utilizzando.

Il registro STATUS è un registro molto importante dei PIC, perché permette di selezionare i banchi RAM e contiene anche i flags, particolari bit indispensabili per far funzionare programmi complessi.

A differenza di altri micro, i PIC sono microcontrollori di tipo RISC, dispongono cioè di un set ridotto di solamente 35 istruzioni elementari eseguite molto velocemente.

Ogni istruzione occupa una sola locazione della memoria programma, e quasi tutte vengono eseguite in 4 cicli di clock.

Se un PIC viene cloccato a 4Mhz è in grado di eseguire 1 milione di istruzioni al secondo (1 mips) e ogni istruzione dura 1µS (1 microsecondo). Le istruzioni di branch (salto, ramificazione) possono richiedere 8 cicli di clock anziché 4.

Nella terminologia Microchip un gruppo di 4 cicli di clock è detto "ciclo macchina", per cui le istruzioni vengono eseguite in uno o due cicli macchina.

Ci sono molti modelli di pic all'interno di una famiglia (architettura), ciascuno con le proprie peculiarità, qualcuno ha più memoria programma, qualche altro ha più periferiche, qualcuno ha dimensioni ridotte (solo 8 pin), altri invece mettono a disposizione più di 30 pin di ingresso/uscita (I/O).

Le istruzioni però sono le stesse e funzionano nello stesso modo.

Per gli esempi che seguiranno verrà usato soprattutto il PIC16F628 e 16 f84 che sono molto diffusi ed economici, non ha bisogno di nessun componente esterno per funzionare (a parte un'alimentazione stabilizzata a 5V).

Per la descrizione dell'architettura interna di uno o più PIC, o delle periferiche in esso contenute si rimanda ai relativi datasheets reperibili sul sito della casa costruttrice [www.microchip.com](http://www.microchip.com).

In questi appunti verranno date le spiegazioni strettamente indispensabili per la comprensione degli esempi pratici.

**IL LINGUAGGIO MACCHINA** Istruzioni, programmi, dati

Il linguaggio macchina è quanto di più vicino ci sia all'hardware.

Il comportamento di ogni istruzione elementare è realizzato in modo fisico dall'insieme dei circuiti logici del micro, ma si può anche dire che l'insieme delle funzioni logiche realizzabili dal circuito determina quelle che sono le istruzioni elementari utilizzabili. Dal punto di vista fisico ogni istruzione è una sequenza binaria di livelli elettrici in grado di attivare in modi differenti i circuiti interni. Le istruzioni assembler sono solo una forma mnemonica comoda per descrivere le sequenze binarie che danno luogo alle funzioni logiche che vogliamo far eseguire al micro, determinate operazioni.

L'assembler è in rapporto 1:1 con il linguaggio macchina e con le funzionalità dell'hardware.

Un programma non è altro che un'insieme di istruzioni, che vengono eseguite una dopo l'altra producendo il risultato voluto, sia esso un gioco o un processo di controllo.

Cosa possono fare le istruzioni dei PIC e in generale di ogni altro microprocessore?

Fondamentalmente solo poche cose:

**Assegnare un valore ad un registro**

**Spostare un valore tra registri**

**Effettuare somme e sottrazioni, incrementi e decrementi su un registro**

**Effettuare operazioni logiche AND OR XOR NOT su un registro**

**Effettuare scambi e rotazioni dei bit di un registro**

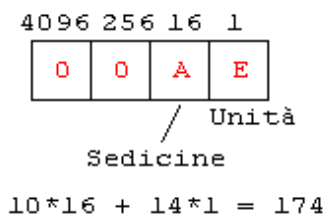
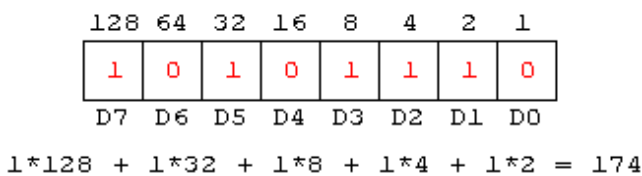
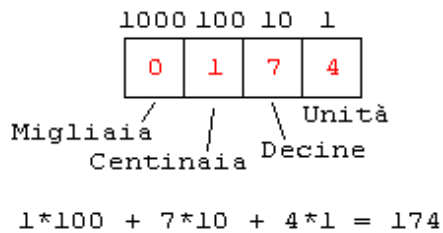
**Settare o resettare singoli bit di un registro**

**Effettuare dei salti condizionati da un punto all'altro del programma**

La maggior parte dell'attività riguarda la manipolazione elementare dei valori o dei singoli bit dei registri.

La sequenza di queste istruzioni ci permettono di costituire la base per il funzionamento di un robot o di uno strumento di misura.

Dal punto di vista fisico il contenuto dei registri non sono altro che sequenze binarie di bit. Questi bit possono rappresentare qualsiasi cosa o qualsiasi tipo di informazione (un numero, un carattere, una parte di un numero più grande ecc...) ed è il programmatore che decide questo al momento in cui scrive il programma. Le istruzioni sono le operazioni elementari che può usare per manipolare i bit delle sue informazioni in modo da arrivare al risultato voluto. In particolare però le istruzioni aritmetiche considerano effettivamente i registri come bytes (8 bit) che contengono un valore numerico da 0 a 255 codificato in forma binaria. La figura seguente mostra le tre rappresentazioni numeriche comunemente usate in campo elettronico e "microinformatico".



Qui a fianco si vede la codifica del valore 174 nel nostro consueto sistema decimale posizionale che usiamo abitualmente. La cifra di destra è la meno significativa (leggera) ed è chiamata unità, mentre quella di sinistra è la più significativa (pesante) ed è chiamata migliaia. Il valore è dato dalla somma delle diverse cifre moltiplicate per il loro peso, quindi abbiamo una volta 100, 7 volte 10 e 4 volte 1, per un totale di 174. Il peso delle cifre è dato dalle potenze di 10, e quindi il sistema si dice decimale, o in base 10.

In binario è la stessa cosa, solo che invece di avere potenze di 10 (1, 10, 100, 1000 ecc) abbiamo potenze di 2 (1,2,4,8 ecc) e le cifre invece di poter assumere valori da 0 a 9 possono essere solo 0 e 1 (da qui il nome bit: binary digit). La cifra meno significativa è chiamata D0 o bit 0 (LSB) e

ha peso 1, quella più significativa è chiamata D7 o bit 7 (MSB) ed ha peso 128.

Si può calcolare che se tutti i bit fossero a 1 la somma dei loro pesi darebbe esattamente 255, cioè il massimo valore codificabile con 8 bit.

La rappresentazione esadecimale è il terzo caso, ed è molto usata perché una cifra (digit) esadecimale rappresenta esattamente 4 bit binari (un nibble). Con due cifre esa da 00 a FF si rappresenta l'intero range di valori codificabili con 8 bit (1 byte). Anche per l'esadecimale abbiamo la cifra meno significativa a destra e quella più significativa a sinistra. Il peso delle diverse cifre questa volta è dato dalle potenze di 16 (sistema in base 16). le cifre possono assumere tutti i valori compresi tra 0 e 15, e i valori tra 10 e 15 vengono indicati con le lettere dalla A alla F. L'esadecimale è molto usato per rappresentare in modo compatto i valori binari contenuti in memoria e il valore degli indirizzi di memoria, e anche perché permette di passare rapidamente alla notazione binaria.

Nell'esempio della figura infatti le due cifre A ed E corrispondono alle due sequenze binarie del 10 e del 14, cioè ai due nibbles 1010 e 1110. La cifra nella posizione delle sedecine rappresenta valori 16 volte più grandi di quelli della cifra delle unità, pertanto  $10 * 16 + 14 = 174$ .