E' opportuno fare una cartella per ogni esempio in quanto, alla fine dell' assemblaggio, verranno prodotti diversi file che può essere interessante visionare con un editor; metter più sorgenti in una sola cartella e poi assemblarli creerebbe un certo guazzabuglio in una singola cartella.

1. Apriamo un editor di nostra scelta (In mancanza di meglio, va bene anche Notepad (orrore!!!)) ed editiamo un file di testo ASCII, quindi non .DOC, .PDF,.EPS,.RTF o altro. Un normale testo .TXT, che chiameremo prova.asm e salveremo dove volete, meglio se in una cartella dedicata, ad esempio /prove_MPASM/.
   Il testo sarà il seguente (di cui potete fare semplicemente un taglia-e-incolla):

```
            list p=16f876             ; define processor
               #include <p16f876.inc>  ; variable definitions
counter     equ     0x35
            ORG     0x00
             goto        init
            ORG         0x20
Init         clrf       counter
Loop         incfsz     counter, f
             goto       Loop
            END
```

2. Attiviamo MPASMWIN e selezioniamo con BROWSE il file prova.asm. Abilitiamo i file in uscita ***List*** ,***Error*** e ***Cross Reference***. Il resto può rimanere in default. Premendo Start si otterrà quasi immediatamente la compilazione

3. Nella cartella ***/prove_PIC/*** ora ci saranno i file ***Prova.asm***, ***Prova.cod***, ***Prova.hex***, ***Prova.lst*** e***Prova.err***., mentre non saranno stati rilevati errori. Possiamo editare questi file (sono tutti testi ASCII e si possono anche leggere con Notepad !) e vederne il contenuto.
   Prova.asm è il nostro sorgente
   Prova.cod è la lista di cross reference
   Prova.hex è l' eseguibile in formato adatto al programmatore
   Prova.lst è la lista assemblata
   Prova.err è la lista degli errori.
   Osservate ad esempio la differenza tra il .asm sorgente e la la lista assemblata .lst, mentre il file di errore sarà vuoto.

4. Proviamo ora a modificare il sorgente così :

```
            list p=16f876           ; define processor
               #include <p16f876.inc>  ; variable definitions
counter     equ      0x35
            ORG      0x00
             goto        init
            ORG         0x20
Init         clrf       counter
Loop         incfsz     counter
             goto       Loop
```

   la differenza pare minima, per noi , ma non per l'Assembler. Salviamo il file come Prova1.asm e ripetiamo l' assemblaggio puntando in BROWSE questo file. Proviamo ad osservare i file .err e .lst : apparirà un warning relativo ad un "Using default destination of 1 (file)". Questo è

dovuto alla mancata indicazione di dove l' istruzione incfsz deve depositare il risultato (nella lista corretta questo era definito dal -,f- che manca in Prova1.asm), Da notare che il warning non è un errore, ma l' avviso di un possibile errore che l' Assembler corregge utilizzanzo un default interno. Quindi sarà prodotto un file Prova1.hex (identico al precedente, mentre il file Prova1.err conterrà l'indicazione della linea incriminata.
Inoltre si determinerà un errore per la mancanza dello statement END che deve chiudere sempre la lista sorgente.

Potete provare a fare qualche modifica a caso al testo ed osservare cosa succede ai messaggi di errore e ai files di uscita.

Fatto tutto questo, magari non sappiamo ancora nulla dell' Assembly PIC, ma almeno abbiamo pronto all' uso l' elemento indispensabile, ovvero l' Assembler MPASM