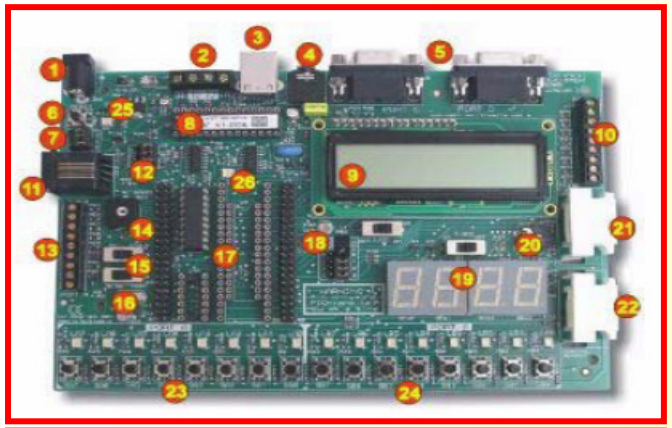
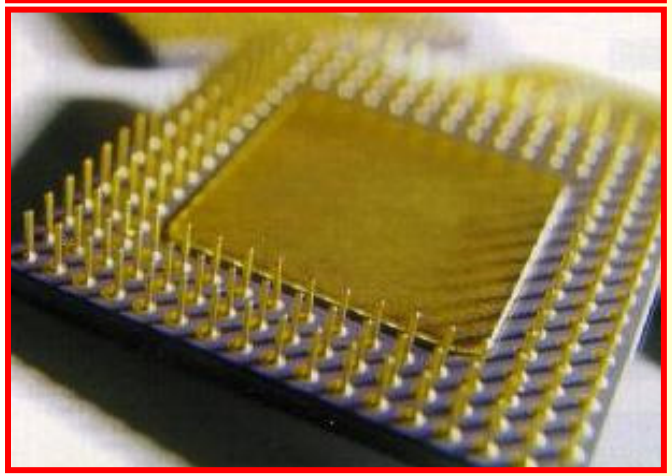
Università degli studi di Cassino

Corso di Laurea in
Ingegneria della Produzione Industriale





Corso di Informatica Applicata

Introduzione

Ing. Saverio De Vito
e-mail: saverio.devito@portici.enea.it
Tel.: +39 081 7723364

Contenuti della lezione

- MACROs;
- Chiamate a Sottoprocedura;
- Modalità di I/O;
- Introduzione agli Interrupt;

Struttura codice assembly

La struttura di un programma assembly è data tipicamente da tre sezioni:

a) Direttive all' assemblatore, definizioni

b) Codice per la logica di business (implementazione algoritmica)

c) Sezione di chiusura

Direttive al compilatore

Il nostro PIC può lavorare in diverse "modalità". In pratica l' output di un codice sorgente è influenzato dallo "stato" definito anche dalle "direttive all' assemblatore" , frasi di codice che addestrano il compilatore a "programmare" lo stato del PIC (es.: definire la locazione di memoria programma in cui caricare il codice)

Direttiva al compilatore ORG

- Un esempio di direttiva è la ORG
- ORG viene utilizzata per forzare il caricamento delle successive istruzioni in locazioni specifiche della memoria programma
- Utilizzo: Riservare locazioni di memoria per
  - Tabelle
  - Vettori di interrupt
  - Localizzazione di segmenti di codice

Definizioni di Macro

- La direttiva #DEFINE è disegnata per la definizione di macro ossia per definire segmenti di codice da associare all' utilizzo di una stringa particolare.
- Il compilatore sostituirà ogni occorrenza della stringa data con il segmento di codice associato.
- Attenzione si tratta di semplice sostituzione

Esempio:

```
#DEFINE PAGE0  BSF 3,5
```

In questo caso ad ogni occorrenza della stringa PAGE0 il code preprocessore (scanner) delll' assemblatore sostituirà l' istruzione BSF 3,5.

Tale istruzione causerà il setting della memoria sul banco 0. In tal modo il programma risulterà + leggibile.

Direttiva EQU

EQU è una direttiva che permette di associare ad una stringa un valore numerico.
E' utile quando si vogliono definire valori numerici di particolare importanza, che occorrono numerose volte nel codice sorgente o semplicemente per motivi di leggibilità.

Direttiva Include

L' utilizzo di tali direttive può ad esempio portare alla definizione di file di testo in cui sono collezionate numerose definizioni di particolare importanza per un particolare PIC target.
Tale file potrà essere incluso con l' utilizzo di una invocazione della direttiva INCLUDE in ogni file .ASM relativo a codive per quel particolare PIC evitando di ricodificare ogni volta definizioni ovvie (Indirizzo porte A e B, Indirizzo del registro di stato etc.).

Chiamate a sottoprocedura

Spesso, alcune operazioni di varia complessità, implementate con diverse istruzioni assembly, devono essere ripetute più volte ed operare con dati diversi al fine della soluzione del problema algoritmico assegnato.
I segmenti di codice relativi, inoltre, necessitano di attenzioni particolari in quanto influenzerebbero il codice sorgente relativo in più punti.
E' possibile definire delle sottoprocedure che implementano le suddette operazioni e possono essere richiamate quando è necessario.

Chiamate a sottoprocedura

In assembly PIC una sottoprocedura è delimitata da una locazione di memoria programma associata eventualmente ad una label e da una istruzione di ritorno da sottoprocedura (RETURN, RETLW).
L' istruzione per invocare una sottoprocedura è la CALL.

Es.: CALL ValveControl

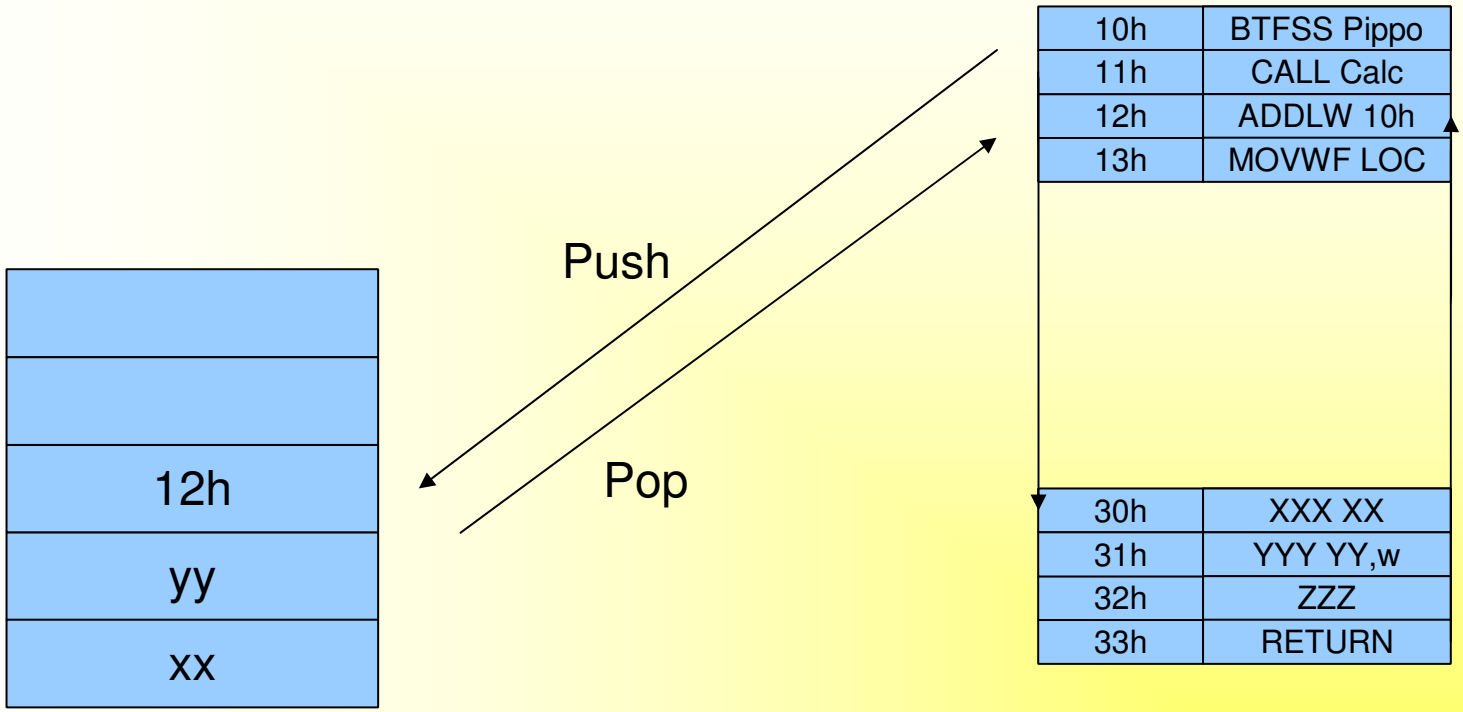
Per implementare la chiamata a sottoprocedura (eventualmente innestata) è necessaria una struttura dati particolare ad accesso ristretto: lo STACK

Lo Stack

Lo Stack (pila) è una struttura di memoria ad accesso contollato con una politica LIFO (Last In First Out).
Questa politica permette l' accesso in lettura (estrazione, pop), ma anche la scrittura (push), alla sola locazione di memoria alla testa dello Stack. Pertanto l' ultimo valore immesso è l' unico accedibile in lettura.
Al verificarsi di una chiamata a sottoprocedura occorre salvare sullo stack l' indirizzo di ritorno che al termine dell' esecuzione della sottoprocedura potrà esser richiamato e utilizzato per caricare l' IR.

Chiamata a Sottoprocedura

Meccanismo di salvataggio dell' IP:



Al termine della Subprocedure l' IP viene caricato con il TOS (Top of Stack).
Grazie a questo meccanismo e alle dimensioni dello stack è possibile rendere disponibili diversi livelli di chiamate a sottoprocedur innestate. Quando lo spazio nello stack si esaurisce si verifica un errore di stack overflow.

Ritorno da Sottoprocedura (RETLW)

L' istruzione RETLW permette di ritornare da una procedura caricando il registro w con un valore letterale, utile come valore di ritorno semplice e per il controllo dell' andamento delle computazioni che avvengono durante l' esecuzione della sottoprocedura.

Modalità di indirizzamento per I/O

Attualmente possiamo distinguere due modalità principali di indirizzamento I/O:

- Port Mapped
- Memory Mapped

Indirizzamento Memory Mapped

Nello schema di indirizzamento Memory Mapped, ogni dispositivo di I/O, o meglio il suo porto associato sono indirizzati attraverso la memoria. Ogni dispositivo ha infatti un indirizzo che rientra nello spazio di indirizzamento della memoria principale. E' questa la modalità utilizzata dal PICMicro 16f84.

Es.: La Porta B è mappata sull' indirizzo 06h

Indirizzamento Port Mapped

Nello schema di indirizzamento Port Mapped, lo spazio di indirizzamento dei porti I/O è separato da quello della memoria principale e il processore tipicamente utilizza un bus separato ad-hoc per indirizzare i dispositivi I/O. A differenza dell' I/O memory mapped, la scrittura e la lettura da un dispositivo I/O vengono espletate tramite un subset di istruzioni ad hoc.

Un esempio tipico è dato dalle architetture Intel che dispongono delle istruzioni IN e OUT.

Port vs Memory Mapping

I/O Memory Mapped è più semplice da realizzare quindi meno costi e più spazio a disposizione on chip

I/O Port Mapped è più semplice da programmare e influenza positivamente la leggibilità del codice, libera spazio in memoria principale, l' hardware comunque necessario per implementarlo è + costoso.

Interrupts : Concetti generali

Il concetto di interrupt si basa sulla necessità di richiamare l' attenzione del processore su eventi asincroni che si verificano al suo intorno.
Questi eventi necessitano di immediate azioni di regolazione e/o servizio da parte del processore stesso.
Nel nostro caso questi eventi particolari possono essere ad esempio legati ad azioni di controllo che un microcontrollore deve intraprendere in risposta al loro verificarsi (controllo di parametri significativi di un microimpianto, apertura airbags etc.)

Interrupts : Concetti generali

Un altro utilizzo fondamentale nell' ambito della programmazione general purpose è il servizio di interrupts legati all' I/O.

In origine sistemi di calcolo primitivi o molto semplici gestivano l' I/O in una modalità molto poco efficiente... il polling.

Un sistema di I/O veniva interrogato regolarmente dal processore per ottenere informazioni circa lo stato del dispositivo (presenza di un carattere in ingresso da un terminale, termine di una scrittura su disco etc.)

Questa modalità porta allo spreco di cicli macchina che sono impiegati di fatto per aspettare una sincronizzazione con il dispositivo.

Interrupts : Concetti generali

Gli interrupts, possono essere invece utilizzati per lanciare operazioni di I/O e lasciare il controllo delle stesse ad un dispositivo di I/O che richiederà l' attenzione del processore allorquando l' operazione sarà stata fisicamente portata a termine.

Es.: scrittura su disco di blocchi di memoria.

Nel frattempo il processore potrà eseguire altro codice senza doversi fermare in attesa attiva. Quando l' operazione sarà terminata il processore sarà avvisato da un opportuno interrupt.