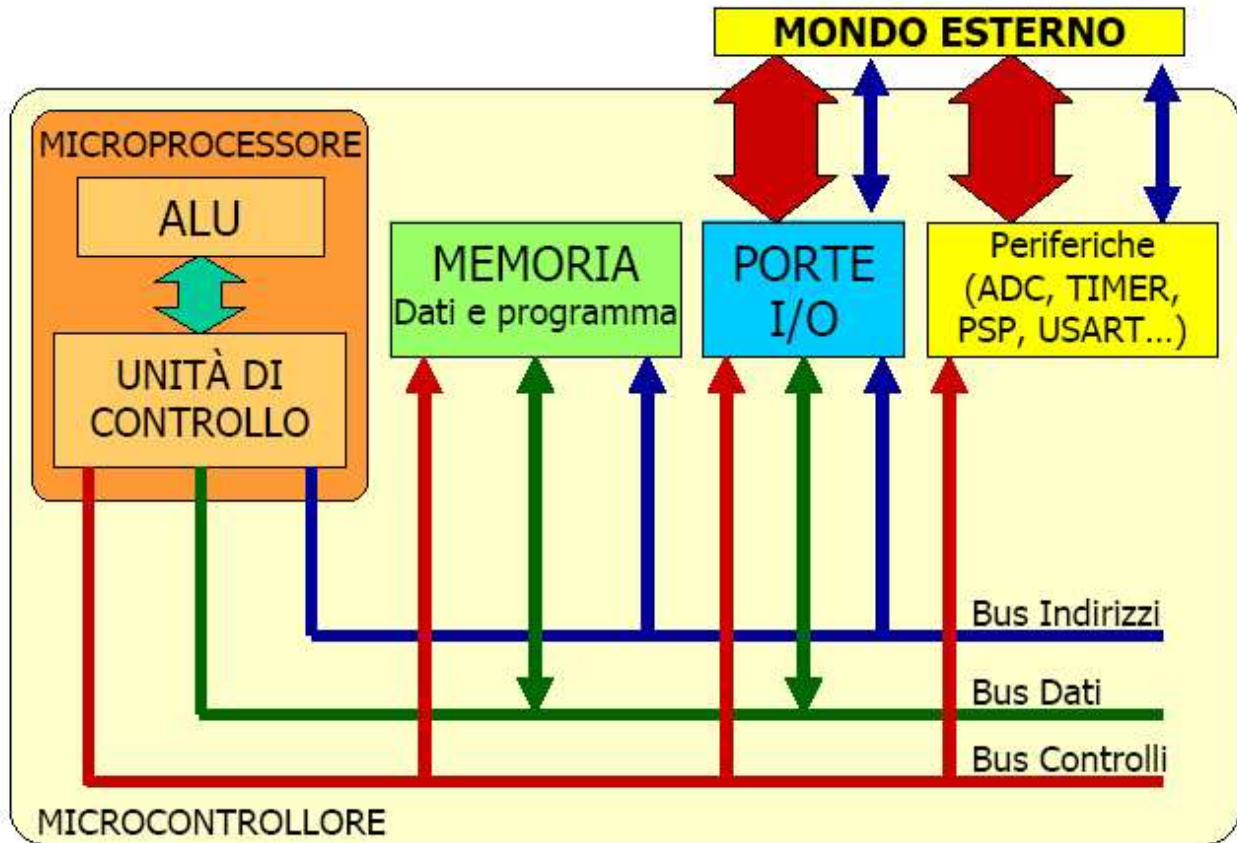


Caratteristiche del μC Pic 16

I micro controllori PIC 16 sono prodotti Microchip Technology Inc., sono componenti che integrano su un unico dispositivo tutti i circuiti necessari a realizzare un complesso sistema di elaborazione dati, internamente dispongono di tutti i dispositivi tipici dei sistemi a micro processori.



In figura è riportata la struttura base di un microcontrollore.

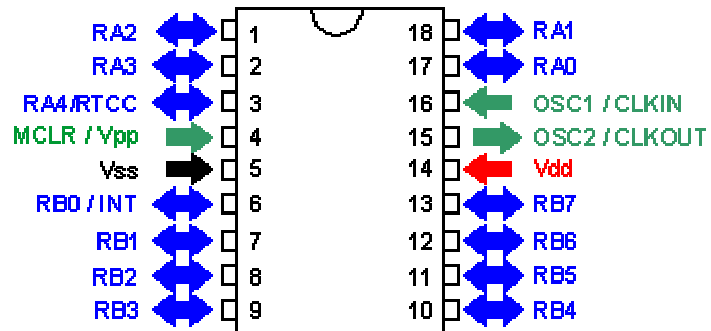
Il μC Pic consta:

- Una CPU unità centrale di elaborazione in grado di interpretare le istruzioni di programma ed eseguirle
- Memoria flash dove sono memorizzate in maniera permanente le istruzioni del programma da eseguire;
- Memoria ram utilizzata per memorizzare le variabili utilizzate dal programma;
- Linee di I/O (input/out) disposte per pilotare dispositivi esteri o ricevere dati da sensori,dispositivi vari
- Dispositivi ausiliari al funzionamento "generatore di klok, Bus, contatori, temporizzatori, ettc..;

la presenza di tutti questi dispositivi racchiusi in un unico cip, permettono di adottare i micro controllori per realizzare circuiti molto complessi facili da migliorare anche dopo la realizzazione pratica.

Uno dei μC della famiglia Pic 16 usato nelle nostre esperienze di laboratorio è il PIC16F84.

Il PIC16F84. presenta la seguente piedinatura.



Pin di alimentazione “Vss pin 5=gnd Vdd pin 14 (tensione compresa tra 2 e 6V)

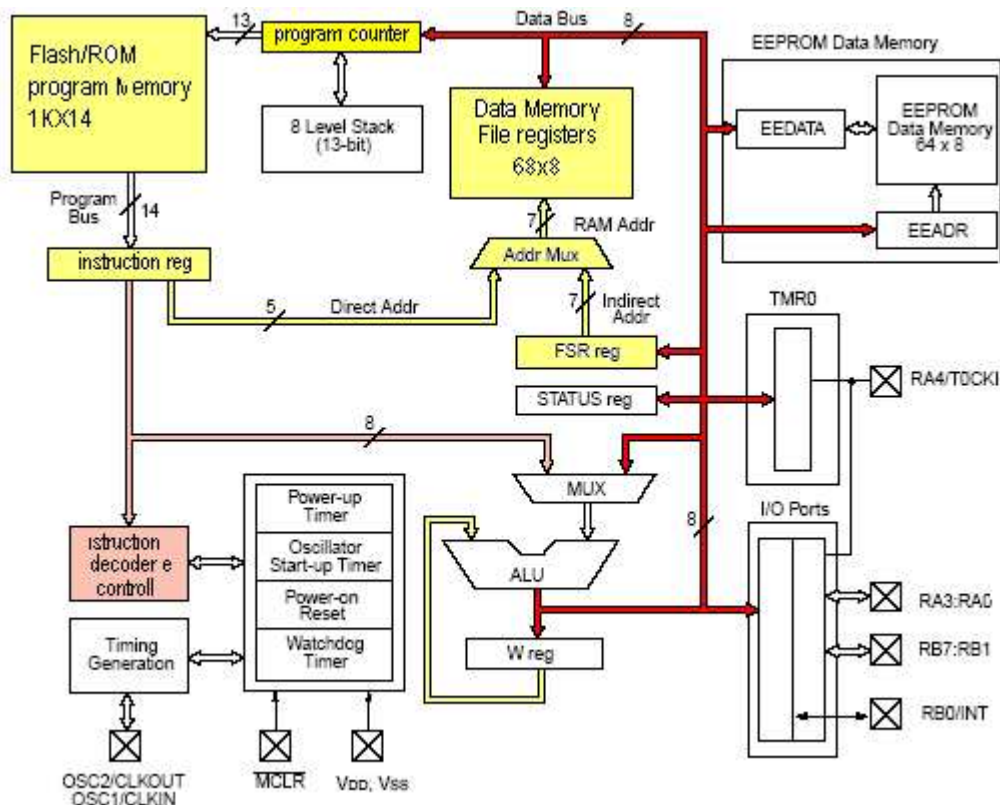
Pin linee di I/O”prendono due nomi, RA ed RB”. ci sono 4 linee RA che vanno da RA0 a RA3.

8 linee RB e vanno da RB0 a RB7. Le linee RA ed RB hanno una corrispondenza con i registri interno del PIC, registri PORTA e PORTB, in modo tale che una modifica al registro si ripercuota sulle linee e viceversa. Tutti i pin delle porte possono essere configurati singolarmente, via software, sia come entrate che come uscite.

Il pin 4 MCLR è il pin adibito al reset del μ C. Quando questo pin è a livello logico basso il PIC si trova in reset e non può eseguire nessun tipo di operazione.

I pin 16 e 17 OSC1/CLKIN e OSC2/CLKOUT sono i piedini predisposti per il cuore del micro cip. Il PIC16F84 ha bisogno di un clock che dia gli impulsi necessari al suo funzionamento. Per questo sono previsti diversi tipi di clock (rete RC, quarzo, oscillatori, ...) che vanno collegati in uno solo o tutti e due i pin OSC1/CLKIN e OSC2/CLKOUT.

Architettura

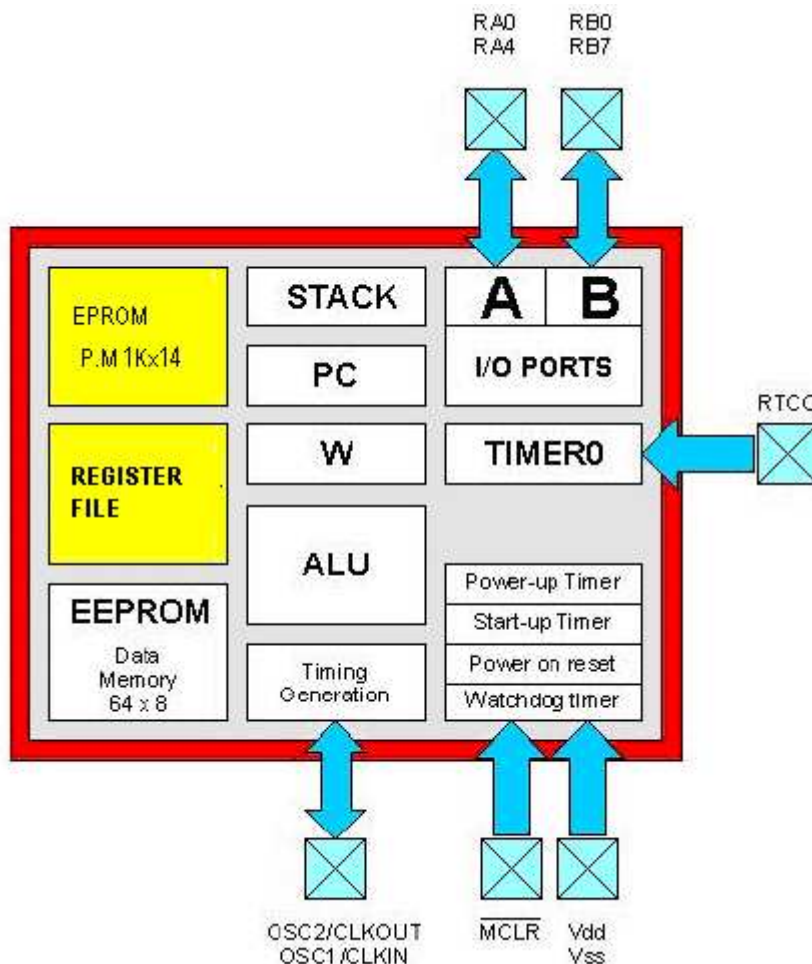


modalità.

Il blocco di decodifica delle istruzioni è collegato con diversi dispositivi, come il Power-Up Timer, il WDT e altri poiché questi ultimi comandano o sono comandati da certe funzioni del PIC che possono essere modificate via software e spigheremo più avanti.

Il PIC16F84 possiede una ALU a 8 bit che ha come ingressi i bus e il registro W. Il registro W è un registro molto utilizzato, sia per quanto riguarda le operazioni della ALU che per quanto riguarda operazioni con i dati e la memoria.

Per rendere più comprensibile la struttura del **PIC16F84A** si fa riferimento alla seguente figura .



Comportamento della program Memory.

Il PICmicro puo' solamente eseguire le istruzioni memorizzate Program Memory. Non puo' in alcun modo leggere, scrivere o cancellare quanto in esse contenuto. Questo vale in particolar modo per i PIC16F84A mentre su altri modelli quali i **PIC16F87x** e' possibile anche aggiornare la memoria programma a run time ovvero a programma in esecuzione. Per scrivere, leggere e cancellare queste locazioni e' necessario utilizzare un **programmatore per PIC**.

L'indirizzo di memoria 0x000, deve sempre contenere l'istruzione di reset. Istruzione che il PICmicro dovra' eseguire. Per questo viene nominata **Reset Vector**.

ORG 0x00 questa istruzione serve per segnare l'inizio del programma. Questa direttiva tiene conto del fatto che l'esecuzione del programma al reset parte dall'indirizzo 0x000 dell'area programma. L'istruzione che segue immediatamente la direttiva **ORG 0x00**, sara' la prima istruzione ad essere eseguita.

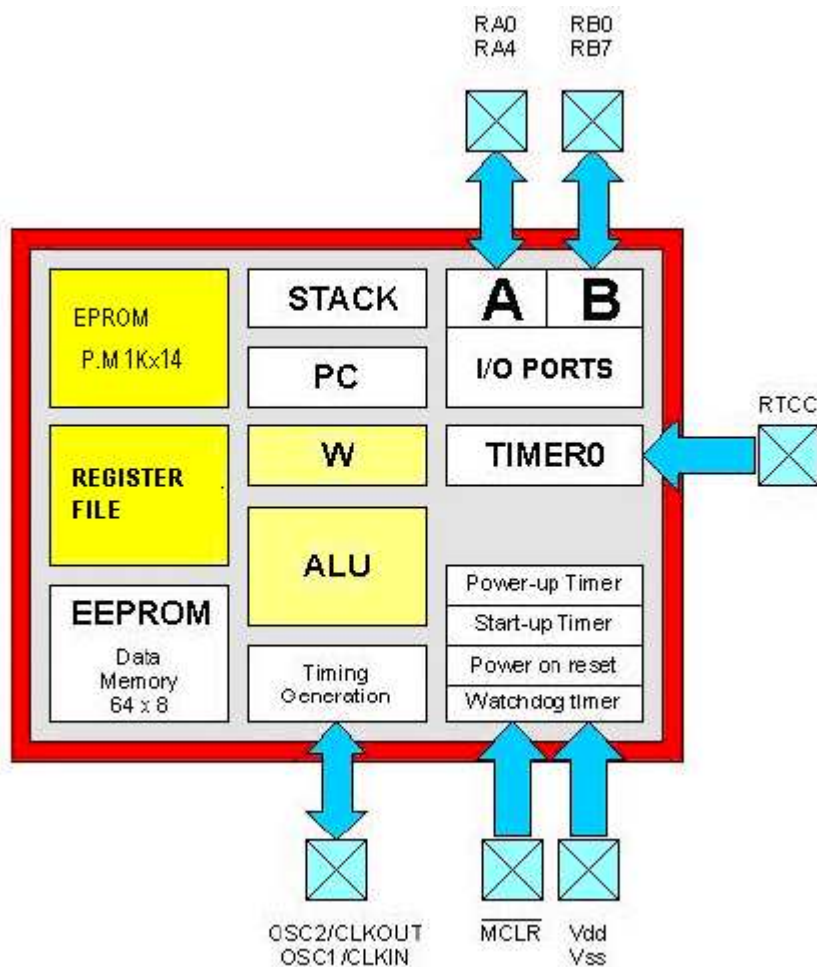
Il REGISTER FILE è un'insieme di locazioni di memoria **RAM** ovvero memorie con cui è possibile leggere e modificare il contenuto senza l'ausilio di programmatori esterni , direttamente dal programma in esecuzione sul PICmicro. Il REGISTER FILE e' la memoria normalmente utilizzata per memorizzare le variabili di programma.

Essendo una memoria RAM il REGISTER FILE perde il suo contenuto quando il PICmicro viene spento , alla successiva accensione e' necessario reinizializzare i valori di tutte le sue locazioni prima di poterla usare.

File Address			File Address
00h	Indirect addr. ⁽¹⁾	Indirect addr. ⁽¹⁾	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISE	86h
07h			87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2 ⁽¹⁾	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch			8Ch

Controllo dell'ardware: Alcune locazioni del REGISTER FILE ed in particolare quelle che si trovano agli indirizzi più bassi vengono usate per il controllo dell'hardware del PIC μ secondo quanto illustrato di seguito. Le locazioni di memoria presenti nel REGISTER FILE sono indirizzabili direttamente in uno spazio di memoria che va da **0x00** a **0x2F** per un totale di **48 byte**, denominato **pagina 0**. Un secondo spazio di indirizzamento denominato **pagina 1** va da **0x80** a **0xAF**. Per accedere a questo secondo spazio e' necessario ricorrere ai due bit ausiliari **RP0** e **RP1** che andremo a spiegare piu' avanti. Le prime **12 locazioni** della pagina 0 (da **0x00** a **0x0B**) e della pagina 1 (da **0x80** a **0x8B**) sono quelle riservate alle funzioni speciali per il funzionamento del PICmicro e non possono essere utilizzate per altri scopi.

Le **36 locazioni** in **pagina 0** indirizzate da **0x0C** a **0x2F** possono essere utilizzate liberamente dai nostri programmi per memorizzare variabili, contatori, ecc.



La ALU è una componente presente in tutti i microprocessori, essa è la componente più complessa del PIC micro in quanto contiene tutta la circuiteria in grado di compiere calcoli e manipolare dati durante l'esecuzione del programma.

La ALU del pic 16F84 è in grado di operare su valori di 8 bit.

L'accumulatore (registro W)

È direttamente connesso alla ALU .il registro W consta di una semplice locazione di memoria in grado di contenere 8 bit.

Il registro W

Rispetto alle altre locazioni di memoria, il registro W è in grado di dialogare con la ALU senza dover fornire nessun indirizzo di memoria, "puo' accedere direttamente".

Esempio pratico.

pensiamo di voler inserire nella locazione di memoria **0xC** del **REGISTER FILE** il valore **0x01**. Tra le istruzioni del PICmicro ci accorgiamo subito che non esiste un'unica istruzione in grado di effettuare questa operazione ma è necessario ricorrere all'accumulatore ed usare due istruzioni in sequenza,

perché un'istruzione non può essere più lunga di 14 bit, mentre a noi ne servono:

8 bit per specificare il valore che intendiamo inserire nella locazione di memoria,

7 bit per specificare in quale locazione di memoria vogliamo inserire il nostro valore,

6 bit per specificare quale istruzione intendiamo utilizzare.

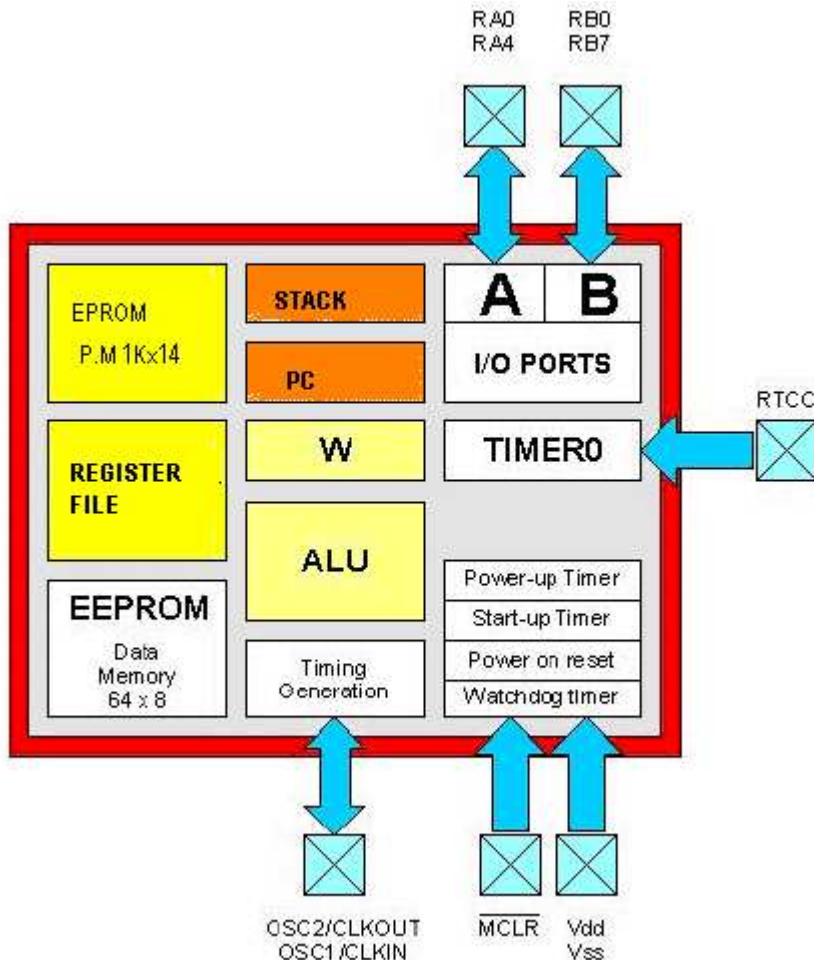
Per un totale di **8 + 7 + 6 = 21 bit**.

Pertanto dobbiamo ricorrere a due istruzioni

`movlw 0x01` inserisco nel **registro W** il valore **0x01H** con l'istruzione **MOVE Literal to W**

`movwf 0x0C` muovono" nella locazione 0x0C con l'istruzione **MOVE W to F**.

Il Program Counter (PC)



Il PIC16F84A inizia l'esecuzione del programma a partire dal vettore di reset (**Reset Vector**) dall'istruzione memorizzata nella prima locazione di memoria (indirizzo 0x000). Dopo aver eseguito questa prima istruzione passa all'istruzione successiva memorizzata nella locazione 0x001 e così di seguito. Se non esistesse nessuna istruzione in grado di influenzare in qualche modo l'esecuzione del programma, il PICmicro in sequenza arriverebbe ad eseguire tutte le istruzioni presenti nella sua memoria fino all'ultima locazione disponibile.

Sappiamo che qualsiasi microprocessore o linguaggio di programmazione dispone di istruzioni di salto, ovvero di istruzioni in grado di modificare il flusso di esecuzione del programma in base alle esigenze del programmatore.

Una di queste istruzioni è il GOTO che ci permette di cambiare la sequenza di esecuzione e di "saltare" direttamente ad un qualsiasi punto, all'interno della memoria programma, e di continuare quindi l'esecuzione a partire da quel punto.

ES.

```
ORG 0x00
```

```
Punto1
```

```
    movlw 10
```

```
    goto  Punto1
```

Al reset il PICmicro eseguirà l'istruzione MOVLW 10 memorizzata alla locazione 0x000, la quale inserirà nell'accumulatore il valore decimale 10, e passerà ad eseguire l'istruzione successiva GOTO Punto1.

L'istruzione goto determinerà un salto incondizionato alla locazione di memoria puntata dalla label Punto1 ovvero di nuovo alla locazione 0x000.

Durante questo ciclo (o loop), per definire quale sarà l'istruzione successiva da eseguire, il PIC utilizza un registro speciale denominato **PROGRAM COUNTER** la cui funzione è quella di mantenere traccia dell'indirizzo che contiene l'istruzione successiva da eseguire.

Questo registro viene incrementato automaticamente ad ogni istruzione eseguita per determinare il passaggio all'istruzione successiva. Al **RESET** del PIC il **PROGRAM COUNTER** viene azzerato, all'indirizzo 0x000.

L'istruzione di salto consente l'inserimento a programma di un nuovo valore nel **P.COUNTER** e di conseguente il salto ad una locazione qualsiasi dell'area programma del PIC.

Lo Stack Pointer

Un'altra istruzione, che influenza il valore del P. C. è la **CALL** con la quale è possibile effettuare delle CHIAMATE A SUBROUTINE.

Questa istruzione funziona in maniera molto simile alla **GOTO**. Come la **GOTO** infatti permette di scrivere nel P.C. un nuovo indirizzo di esecuzione del programma. La differenza sostanziale consiste nel fatto che prima di eseguire il salto, il PIC memorizza, in un altro registro speciale, denominato **STACK**, l'indirizzo di quella che sarebbe dovuta essere la successiva istruzione da eseguire se non si fosse incontrata la **CALL**.

ES:

```
        ORG    0x00
Punto1
        movlw 10
        call  Punto2
        goto  Punto1
punto2
        movlw 11
        return
```

In questo programma il PICmicro, dopo aver eseguito la **MOVLW 10** passa ad eseguire l'istruzione **CALL Punto2**.

Prima di saltare però, memorizza nello **STACK** l'indirizzo 0x002, l'indirizzo della locazione successiva alla **CALL**.

L'esecuzione passa all'istruzione **MOVLW 11** e quindi alla istruzione **RETURN**.

Questa istruzione, consente di riprendere l'esecuzione a partire dall'istruzione successiva alla **CALL** che aveva determinato l'abbandono del flusso principale del programma utilizzando il valore memorizzato nel registro di **STACK**.

La parola **STACK** in inglese significa "**catasta**" ed infatti su questa catasta è possibile depositare, uno sull'altro, più indirizzi per recuperarli quando servono. Questo tipo di memorizzazione viene anche denominata **LIFO** dall'inglese **Last In First Out**, in cui l'ultimo elemento inserito (last in) deve necessariamente essere il primo ad uscire (last out). Grazie a questa caratteristica è possibile effettuare più **CALL** annidate ovvero l'una nell'altra e mantenere sempre traccia del punto in cui riprendere il flusso al momento che si incontra una istruzione **RETURN**.

Vediamo un altro esempio:

```
        ORG    0x00
Point1
        movlw 10
        call  Point2
        goto  Point1
Point2
        movlw 11
        call  Point3
        return
Point3
```

movlw 12
return

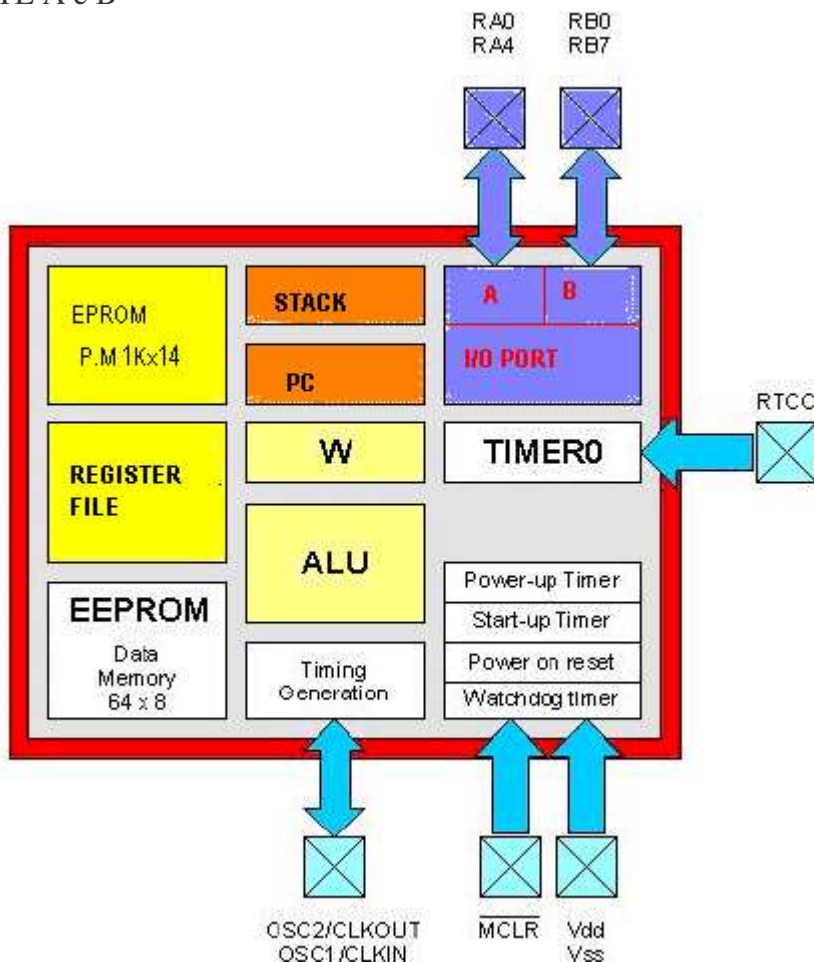
In questo caso nella subroutine Punto2 viene effettuata un'ulteriore CALL alla subroutine Punto3. Al ritorno da quest'ultima il programma rientra nella subroutine Punto2 ed esegue l'istruzione RETURN e ritorna nel flusso principale.

In questo esempio gli indirizzi da memorizzare nello stack sono due in quanto ci sono due CALL prima di incontrare l'istruzione RETURN corrispondente alla prima cal.

Il PIC16F84A dispone di uno stack a 8 livelli, ovvero uno stack che consente fino ad 8 chiamate annidate.

È importante assicurarsi, durante la stesura di un programma, che per ogni cal ci sia sempre una istruzione RETURN in questo modo si evitano disallineamenti dello stack “ si evitano errori che in esecuzione sono difficilmente rilevabili”.

PORTE A e B



Il PIC16F84A dispone in totale di 13 linee di I/O organizzate in due porte chiamate **PORTA A** e **PORTA B**.

La PORTA A consta di 5 linee con la possibilità di essere configurate in ingresso o in uscita (sigla di identificazione RA0, RA1, RA2, RA3 ed RA4).

La PORTA B consta di 8 linee anch'esse configurabili come la porta A (siglerei terminali di uscita RB0, RB1, RB2, RB3, RB4, RB5, RB6 ed RB7).

Il PIC16F84A prevede la gestione di dati di lunghezza massima pari a 8 bit.

Per la gestione delle linee di I/O, il PIC dispone di due registri interni per ogni porta chiamati TRISA e PORTA per la porta A e TRISB e PORTB per la porta B.

Il funzionamento in ingresso ed in uscita di ogni singola linea sono determinati dai registri TRIS A e B, i registri PORT A e B determinano lo stato delle linee in uscita o registrano lo stato delle linee in ingresso.

I bit contenuti nei registri sopra citati corrisponde univocamente ad una linea di I/O.

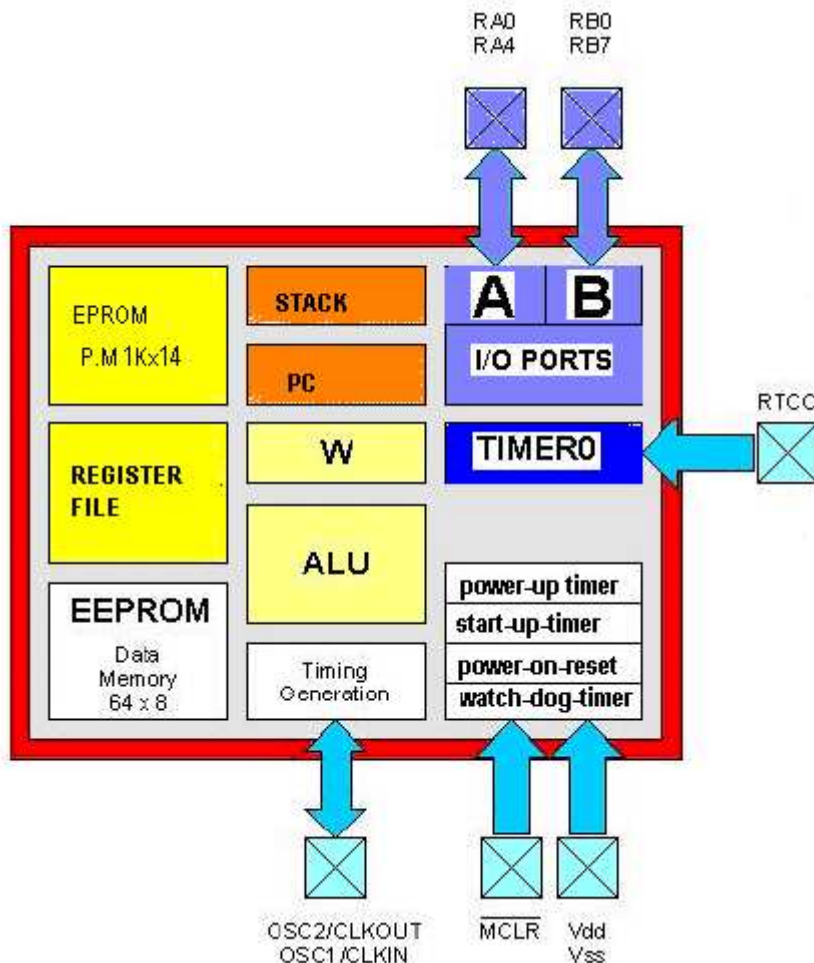
Es: il bit 0 del registro PORTA e del registro TRIS A corrispondono alla linea RA0 , il bit 1 alla linea RA1 e così via.

Per configurare come uscita la linea RA0, il bit 0 del registro TRISA viene messo a zero.

Il valore a cui verrà messo il bit 0 del registro PORTA determinerà lo stato logico di tale linea.

Per configurare come ingresso la linea RA0, il bit 0 del registro TRISA viene messo a uno.

Lo stato logico della linea RA0 verrà memorizzato sul bit 0 registro PORTA.



Il registro contatore TMR0

Il registro TMR0 è un contatore, un registro il cui contenuto viene incrementato con cadenza regolare e programmabile direttamente dall'hardware del PIC. A differenza di altri registri, il TMR0 non mantiene invariato il valore che gli viene memorizzato, ma il valore registrato viene incrementato continuamente,

Es registriamo inizialmente al suo interno il valore 05:

```
movlw 05
movwf TMR0
```

trascorso un tempo pari a quattro cicli macchina, il contenuto del registro viene incrementato a 6, 7, etc. con cadenza costante indipendente dall'esecuzione del programma.

Il valore massimo che il registro TMR0 può raggiungere è 255. Arrivato a questo valore il registro viene azzerato automaticamente riprendendo il conteggio non dal valore originariamente impostato ma da zero.

La frequenza di conteggio dipende dal clock applicata al chip, può essere modificata programmando opportunamente alcuni bit di configurazione.