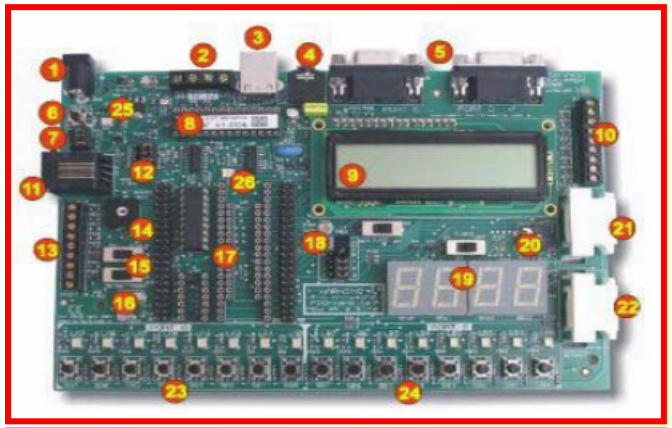
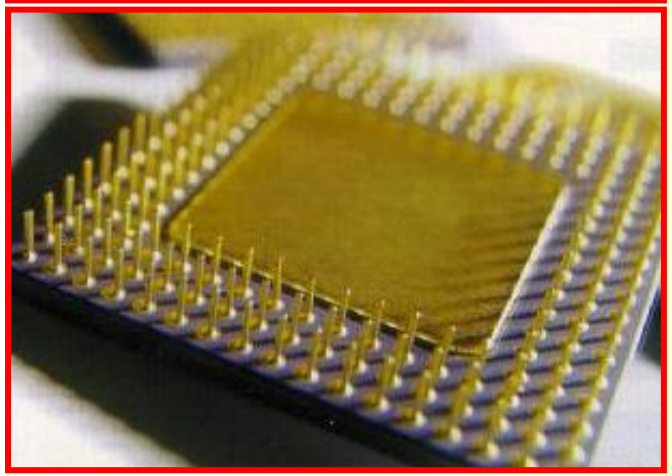
Università degli studi di Cassino

Corso di Laurea in
Ingegneria della Produzione Industriale





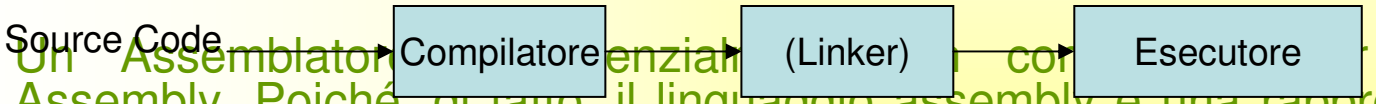
Corso di Informatica Applicata

Introduzione

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

Compilatori, Assemblatori, Interpreti

- Il processo di compilazione risolve il gap tra linguaggi “vicini” al linguaggio naturale e linguaggi vicini al livello della macchina standard (+ S. O.) attraverso un processo di traduzione dell’ intero codice sorgente. Il processo è più o meno complesso in dipendenza della differenza di complessità tra i linguaggi, e della complessità architetturale.



- Un Assemblatore è essenzialmente un compilatore per linguaggio Assembly. Poiché, di fatto, il linguaggio assembly è una rappresentazione simbolica del linguaggio numerico della macchina (corrispondenza 1-1), il compito di traduzione è molto più leggero di quello di un compilatore per linguaggi ad alto livello (C++, C, Pascal, Fortran, etc.)

- Il processo di interpretazione vince il gap traducendo le singole istruzioni in sequenze di istruzioni eseguibili da una macchina virtuale sottostante. Un interprete esegue il lavoro di una macchina virtuale (es.: Interprete Matlab)

Compilazione vs. Interpretazione

Vantaggi

- Esecuzione + Veloce
- Ottimizzazione (Efficienza) cross-instruction

Svantaggi

- Staticità

Vantaggi

- Portabilità (?)
- Ottimizzazione cross-instruction

Svantaggi

- Esecuzione + lenta
- Codice poco efficiente

Il processo di Assemblaggio

- A prima vista sembrerebbe che l' assemblatore non debba far altro che tradurre un istruzione assembly in una a livello macchina e ricominciare il processo con la prossima istruzione fino alla fine del codice sorgente… ed in un certo senso ciò è vero.

- In realtà la problematica è leggeremente + complessa (es.: riferimento anticipato – uso di un etichetta prima della definizione in un salto)

- Soluzioni per il riferimento anticipato:
  - Traduzione in due passi (sol. Vincente)
  - Traduzione a singolo passo con tabella dei non-risolti

Architetture PIC (PICmicro):

Le architetture PIC sono implementate su singolo chip con funzioni di microcontrollo, utilizzate in applicazioni embedded. A differenza di realizzazioni di architetture microprocessore tradizionali, i PIC sono orientati ad applicazioni:

•Single Task

I PIC sono

•Monoprogramma. Immagazzinato in memoria da dispositivi esterni.

L' organizzazione architetturale prevede:

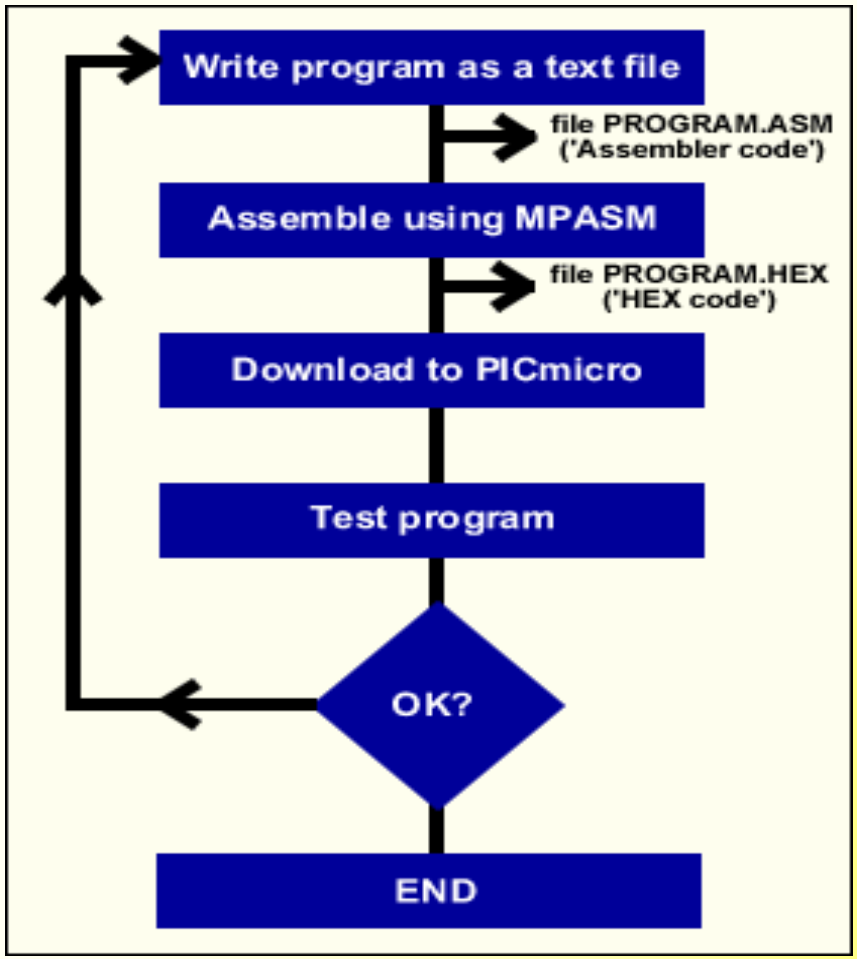
- Memoria e drivers di periferiche (I/O) interni
- Microarchitettura RISC
- Harvard Model (dual bus, differisce dalla von neumann per avere bus differenti per codice e dati)

Architetture PIC (PICmicro):

Il successo delle architetture PIC è dovuto sostanzialmente alla:

- Semplicità di programmazione (ISA ridotto)
- Facile integrabilità
- Flessibilita -> Famiglie
- Ricerca della massima efficienza a costo contenuto
- Al momento, queste architetture sperimentano una grande diffusione nella domotica ed in generale nel controllo digitale per applicazioni low-cost

Programming Cycle



L’ ambiente Target

PIC16F84 [18 pin]

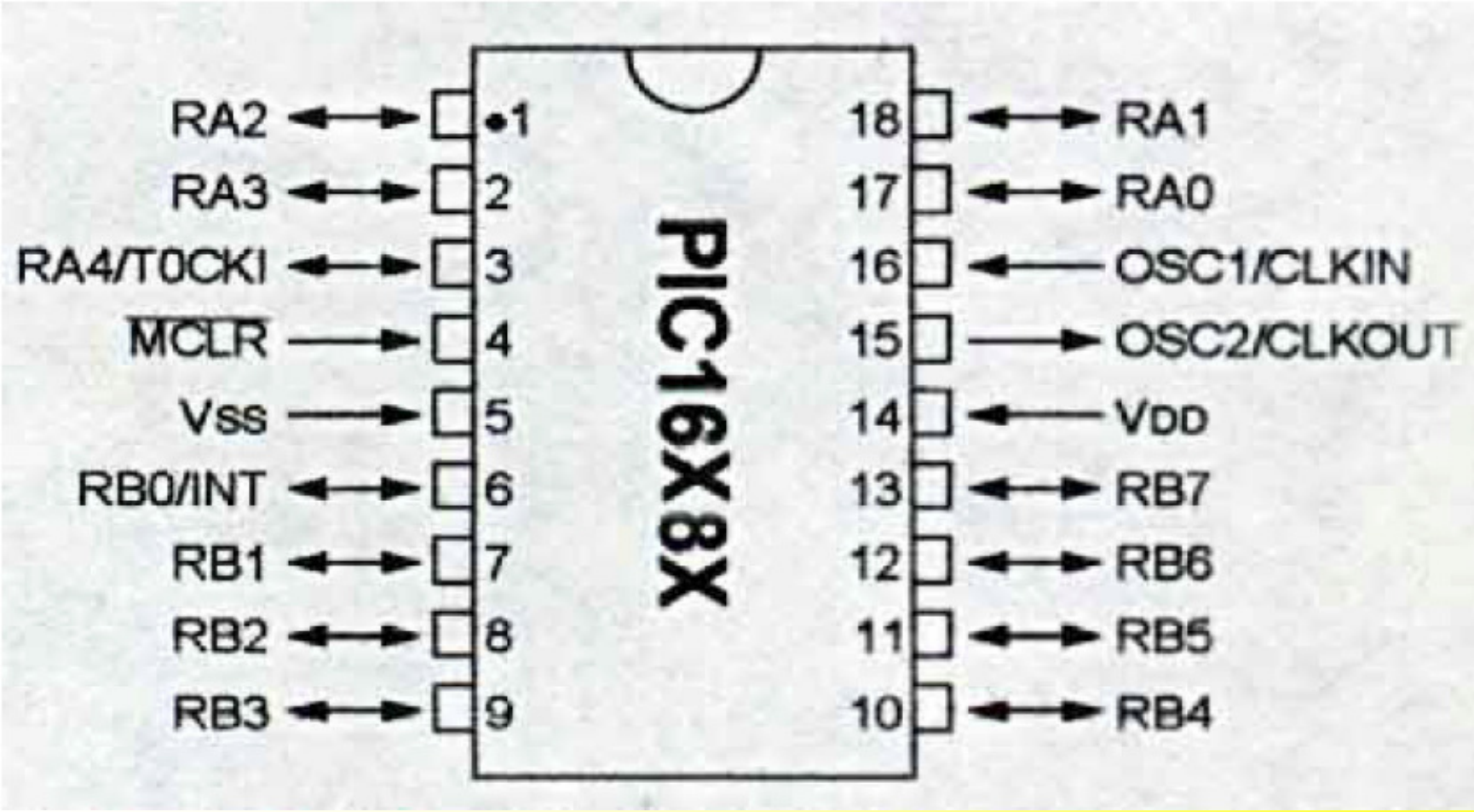
- E’ definibile come macchina RISC (35 istruzioni).
- 2-stage pipeline [Fetch/Decodifica e Esecuzione]
  - Tempo di esecuzione di una generica istruzione = 1 ciclo macchina (eccetto le istruzioni di salto, che ne richiedono due)
  - 1 ciclo macchina corrisponde a quattro cicli di clock.
- Architettura Harvard.
- Frequenza di clock pari a 10MHz
- Parola codice a 14 bit

- Memoria Interna separata per Codice (1k*14bit ->1024 istruzioni) e Dati (68 Byte SRAM) + (EEPROMs 64 bytes)
  - Deperibilità delle EEPROMs

L’ ambiente Target (2)

- Architettura ad accumulatore
- Stack separato a 8 livelli
- I/O TTL compatibile
- Power on Reset
- Sleep mode
- Cifratura dati
- Costo … pochi euro (<5)

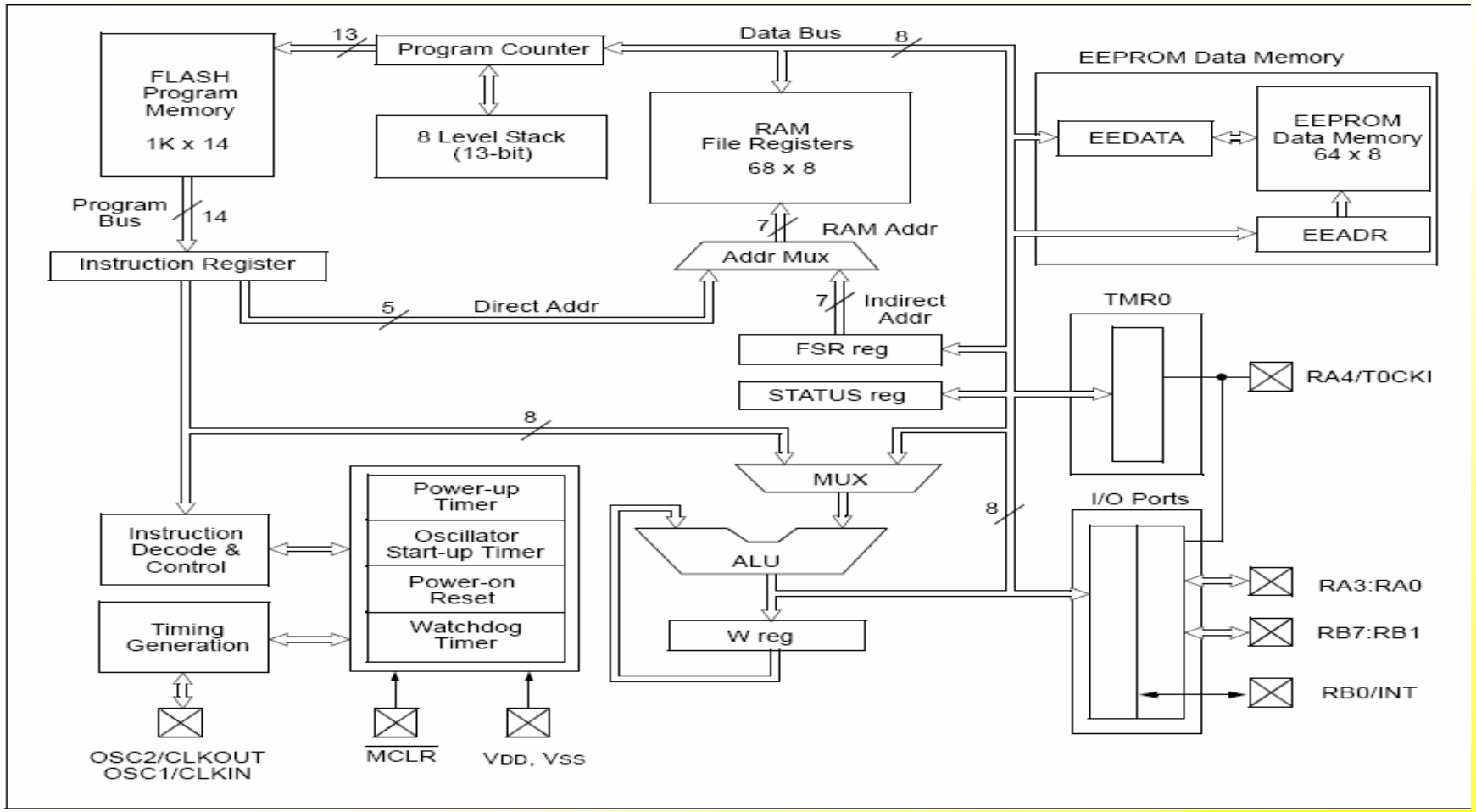
PIC16F84 [18 pin]



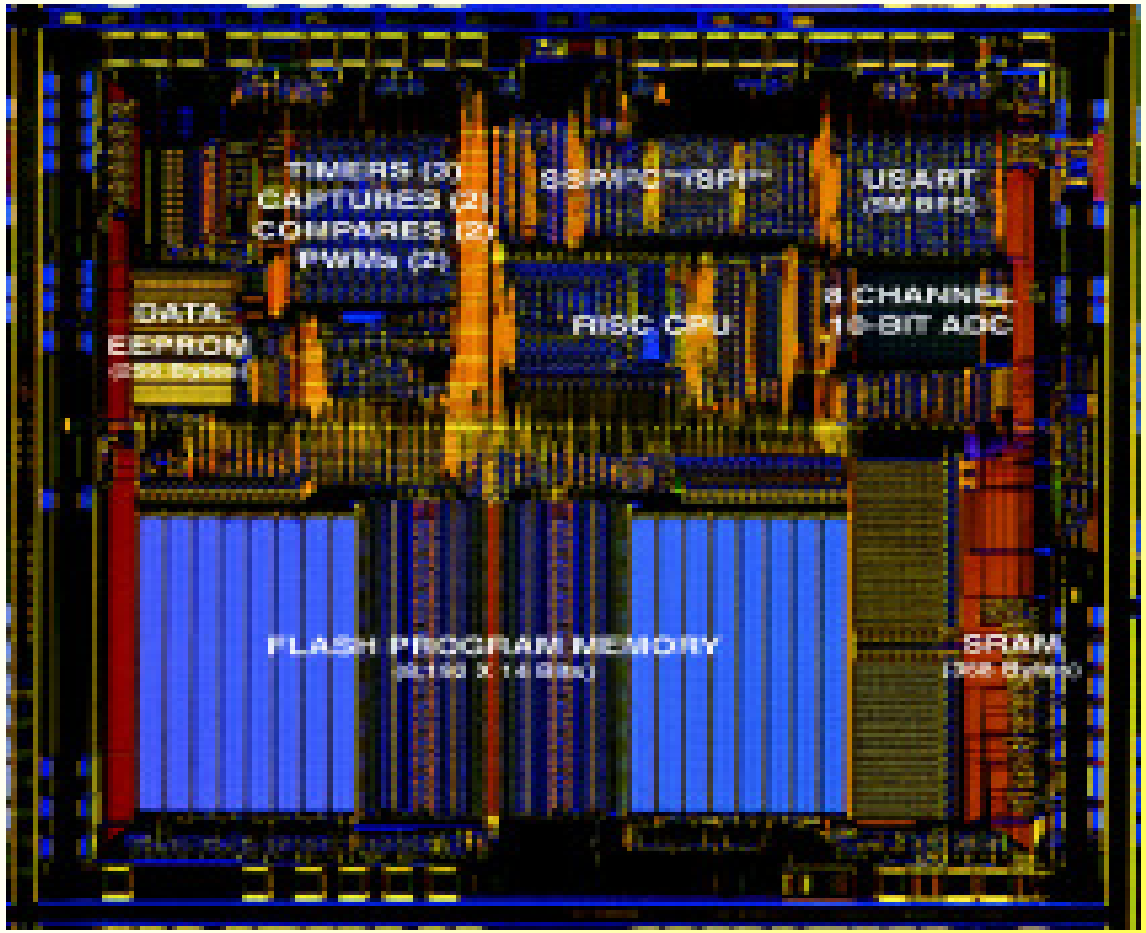
PIC 16F84 : Piedinatura

- *RA0 To RA4 :* RA è una porta bidirezionale (I/O).
- *RB0 To RB7 :*RB è una seconda porta bidirezionale. Si comporta esattamente come la RA, però ci sono 8 bit.
- *VSS And VDD: A*limentazione del microcontrollore. VDD è il positivo, and VSS è il negativo, o 0V. La massima tensione applicabile è di 6V, e il minimo è 2V
- *OSC1/CLK IN And OSC2/CLKOUT: P*in a cui si collega l'oscillatore esterno (external clock), in modo tale che il microcontrollore possa essere in grado di gestire il CLK.
- *MCLR* Pin utilizzato per cancellare la memoria del PIC (i.e. quando vogliamo riprogrammarlo).
- *INT ingrasso INT*.
- *T0CK1* è un altro ingresso di clock, su questo opera un timer interno. È totalmente indipendente rispetto al clock principale.

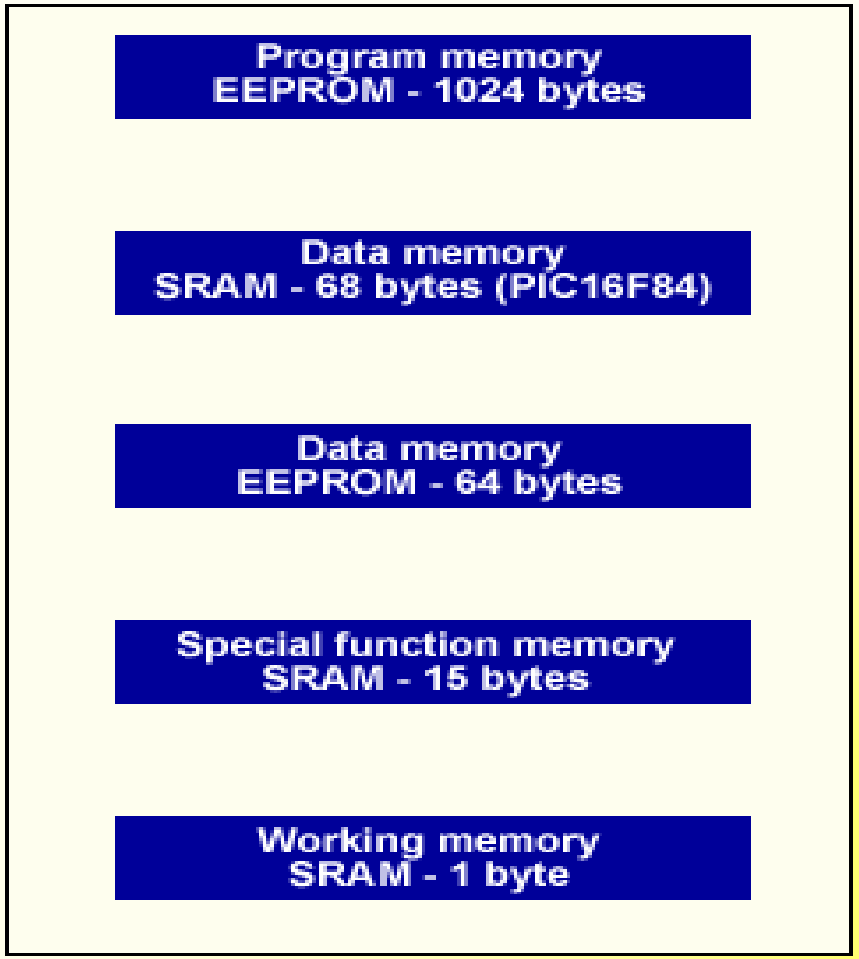
PIC 16F84 Data Path



Alziamo il coperchio ☺……



Organizzazione della Memoria



Architettura Register File

Il Register file è costituito da

- 48 locazioni (byte) di memoria RAM volatile veloce indirizzabili da 0x00 a 0x2F (pagina 0)
- 48 locazioni locazioni (byte) di memoria RAM volatile veloce indirizzabili da 0x80 a 0xAF (pagina 1) .

Le prime 12 locazioni delle due pagine sono utilizzate in modo speciale per specificare le modalità di operazione del PIC (ad esempio configurare le porte di I/O)

Per la natura delle applicazione dei PIC, i registri speciali vengono utilizzati di frequente

Le altre locazioni possono essere utilizzate liberamente dal programmatore.

Architettura ad Accumulatore : Registro W (Working Memory) -> (Ops A-L, Ops MD)

Program Memory

Caratteristiche

- 1024 locazioni ognuna in grado di contenere un' istruzione da 14 bit
- Indirizzi da 0x000 a 0x3FF
- Il PC è inizialmente posto a 0x000, per cui ad ogni reset il PIC inizierà ad eseguire le istruzioni a partire dalla locazione 0x000.
- Il PIC target non è in grado di modificare a run time il contenuto della memoria di programma, anyway esistono versioni (F87) in grado di fare ciò.

Porte di I/O

- Il PIC 16F84 è dotato di porte di input output (A e B)
- Porta A: 4 bits ; Porta B: 8 bits
- I singoli PIN delle porte possono essere configurati come Input o Output digitale mediante l' utilizzo di opportune istruzioni operanti su registri speciali

Istruzioni di movimentazione dati

Le istruzioni di movimentazione dati nel PIC target possono :

- Spostare dati dalla memoria dati al registro accumulatore
  - MOVLW (Move literal to W) : MOVLW K
- Spostare dati da locazioni della memoria a W/F (Diadica)
  - MOVF (Move File A to B) : MOVF A,B B = W o F (test)
- Spostare dati dall' accumulatore ad una locazione di memoria
  - MOVWF (Move W to F) : MOVWF K

Operazioni Aritmetiche Principali

- Addizionare
  - ADDLW (Add Literal to W) : ADDLW K
  - ADDWF (Add W to F) : ADDWF F,D
- Sottrarre
  - SUBLW (Subtract W from L) : SUBLW K
  - SUBWF (Subtract W form F : SUBWF F,D
- Incrementare
  - INCF (Increment F) : INCF F,D
- Decrementare
  - DECF (Decrement F) : DECF F,D

Operazioni Logiche Principali

Il PIC Target è dotato di aritmetica intera ad 8 bit

- AND
  - ANDLW (AND Literal and W): ANDLW K
  - ANDWF (AND W with F) ANDWF F,D
- OR
  - IORLW (Inclusive OR Literal with W) IORLW K
  - IORWF (Inclusive OR W with F) IORWF F,D
- XOR
  - XORLW (Exclusive OR Literal with W) XORLW K
  - XORWF (Exclusive OR W with F) XORWF F,D
- Shift
  - RLF (Rotate Left through Carry) RLF F,D
  - RRF (Rotate Right through Carry) RRF F,D

Le operazioni aritmetico logiche hanno effetti su alcuni bit del registro di Flag (STATUS)

Setting delle Porte

- L' ISA del PIC 16F84 mette a disposizione operazioni per il set di singoli bit che possono essere utilizzati per settare i valori in uscita

- BSF Bit Set File : BSF F,B
- BCF Bit Clear File : BCF F,B

- Set e Clear vanno intesi nella logica positiva (Set=porre ad 1; Clear=porre a 0)

DataSheet

. . . . . .

Continua sul Data Sheet

. . . . . .

Riferimenti utili:

MicroChip:

http://www.microchip.com

DataSheet PIC16F84

http://ww1.microchip.com/downloads/en/DeviceDoc/41202C.pdf

Breve introduzione a MPLAB:

http://www.dsi.unifi.it/users/banci/micro_pdf/PIC01.pdf

Tutorial su PIC16F84

http://www.tanzilli.com/?id=210

Riepilogo et al.:

- Compilatori, Assemblatori, Interpreti.
- Architetture PIC
- Il PIC 16F84(A)
- ISA e DataSheet del PIC 16F84(A)