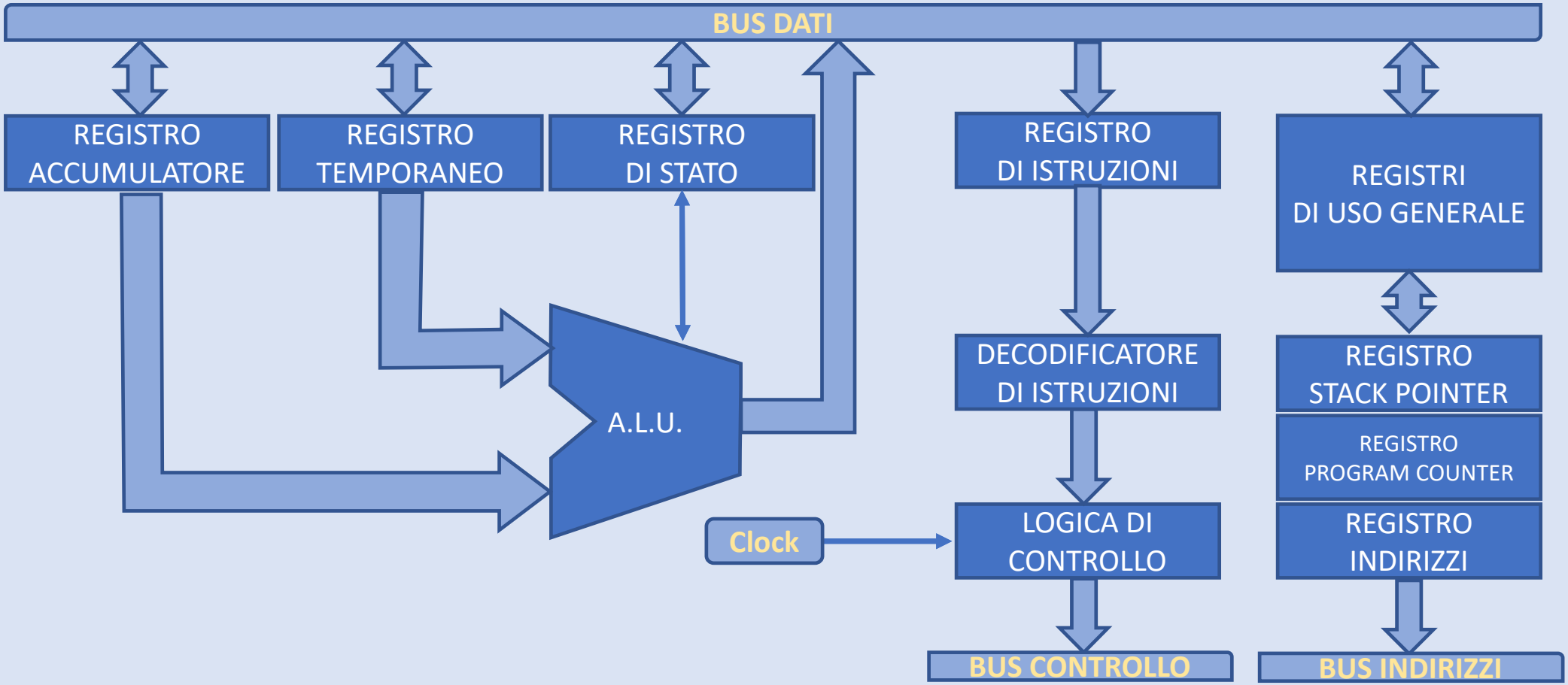
Il Microprocessore

- *Associamo tutti il nome **MICROPROCESSORE** all'unità di calcolo di un Personal Computer.*
- *Il suo nome ha origine da **PROCESSORE** e cioè quel dispositivo **HARDWARE** in grado di eseguire autonomamente delle istruzioni.*
- ***Il MICROPROCESSORE** in effetti esegue questa funzione all'interno di un **PC**, e viene definito in questo modo per le sue caratteristiche dimensionali, in quanto il circuito **HARDWARE** in grado di eseguire istruzioni, è implementato in un unico **CHIP INTEGRATO.***
- *Al suo interno troviamo una serie di **REGISTRI** cioè delle memorie **RAM** (memorie volatili, in quanto perdono il loro contenuto allo spegnimento) che hanno delle funzioni ben definite.*
- *Troviamo anche una **ALU** cioè un**'UNITA' ARITMETICO LOGICA** che rappresenta quella parte circuitale che esegue le **OPERAZIONI ARITMETICO/LOGICHE** previste nel **SET di ISTRUZIONI.***
- *Quest'ultimo rappresenta l'insieme di **OPERAZIONI** che il microprocessore può eseguire.*
- *Il microprocessore viene definito spesso con l'acronimo **CPU** (Central Processing Unit).*

Il Microprocessore

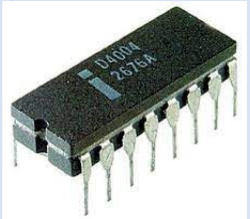


Il Microprocessore

*Il padre del Microprocessore è **Federico Faggin**, un fisico Italiano che nel **1970** ha realizzato il primo microprocessore alla **INTEL** il **4004**. All'interno di un microprocessore c'è un vero e proprio circuito elettronico composto da migliaia fino a miliardi di transistor.*

Federico Faggin

INTEL 4004
2300 Transistor

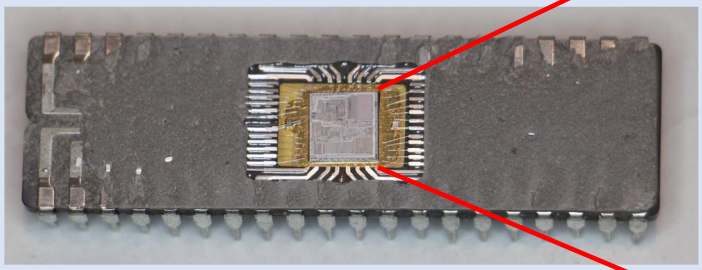


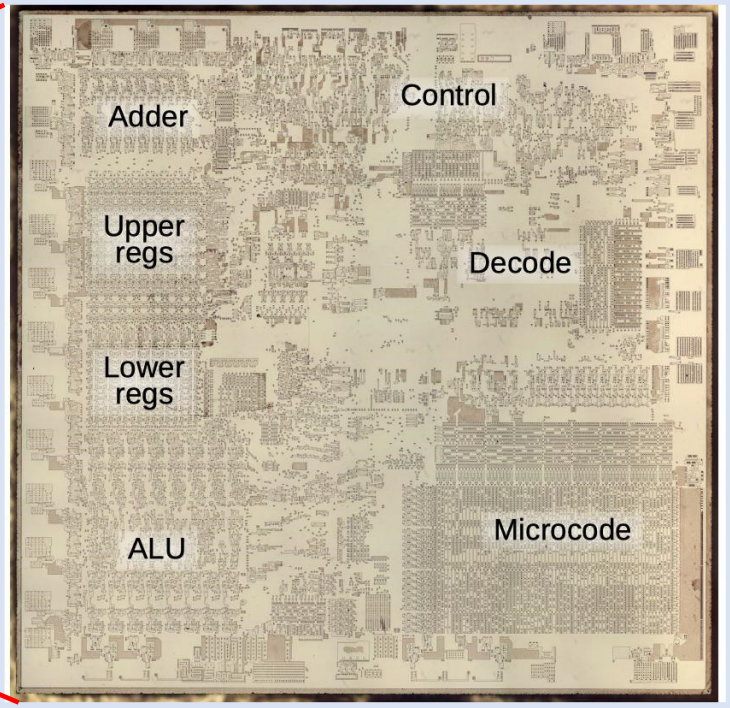
INTEL 8086
29000 Transistor

INTEL 8086
2,6 miliardi di
Transistor

Il Microprocessore

Ma come fatto internamente un Microprocessore?
Nella figura che segue possiamo vedere l'interno del vecchio microprocessore 80286 della INTEL.
In pratica non è altro che un complesso circuito miniaturizzato e composto da migliaia di componenti, costruito su un piccolissimo substrato di Silicio.





Ovviamente tralasciamo la complessa fase di progettazione e realizzazione.

Il Microprocessore

*Il microprocessore è però una parte di un elaboratore e da **solo non è in grado di eseguire nulla**.*

Per realizzare un sistema in grado di eseguire un programma, servono infatti altri dispositivi:

- *una memoria **RAM** esterna per salvare dei dati durante l'esecuzione del programma,*
- *una memoria **non volatile (PROM, EPROM, EEPROM** ecc…) per memorizzare il programma da eseguire,*
- *un circuito per generare il **CLOCK**, cioè quel segnale che scandisce la velocità di esecuzione delle istruzioni,*
- *delle **PERIFERICHE** di ingresso e di uscita, per interfacciarsi con il mondo esterno e con l'operatore.*

Il microprocessore è connesso con questi dispositivi tramite le linee che compongono:

- *Il BUS Dati,*
- *Il BUS Indirizzi,*
- *Il BUS di controllo.*

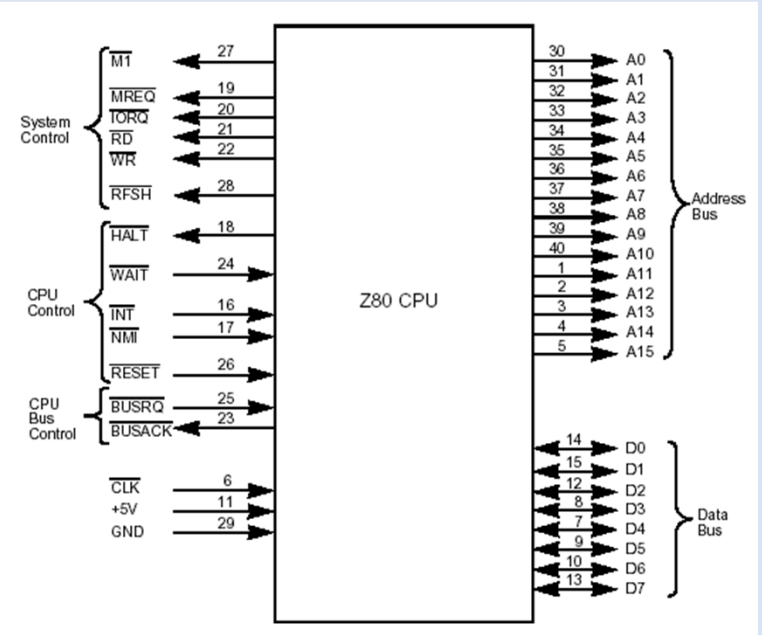
*Con il termine BUS si identifica un insieme di collegamenti in grado di trasferire delle informazioni (**BIT**) sottoforma di segnali elettrici.*

Il Microprocessore

Un altro famoso microprocessore progettato e realizzato da Federico Faggin, è lo Z80.

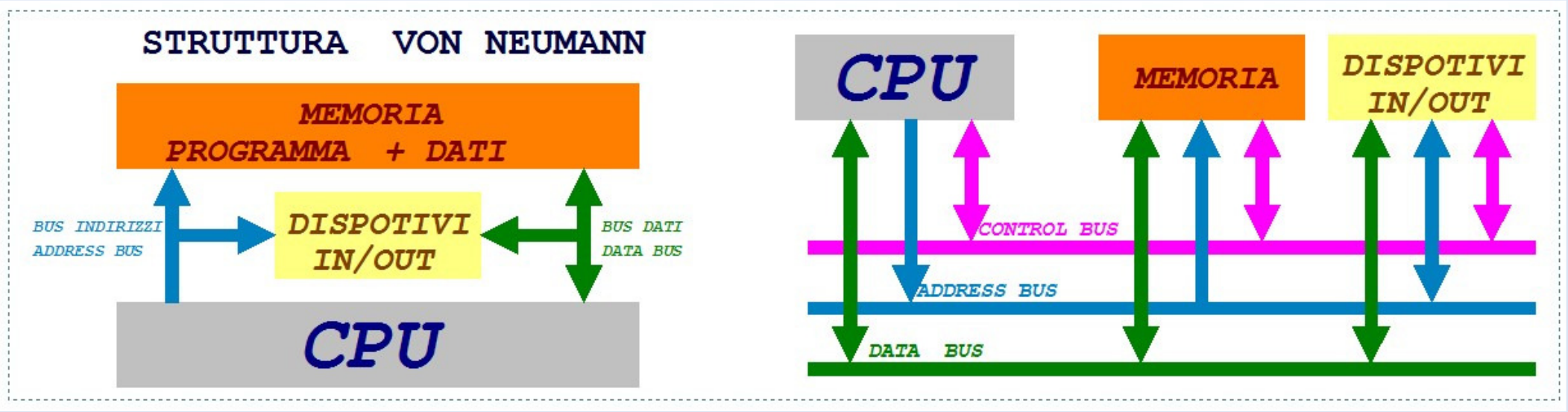
Utilizzato nelle prime consolle di videogiochi ed in ambiti Industriali, lo Z80 è sicuramente uno dei microprocessori più utilizzati al mondo.

In figura possiamo vedere le linee dei BUS con cui il microprocessore si collega con le varie memorie e periferiche di input/output.



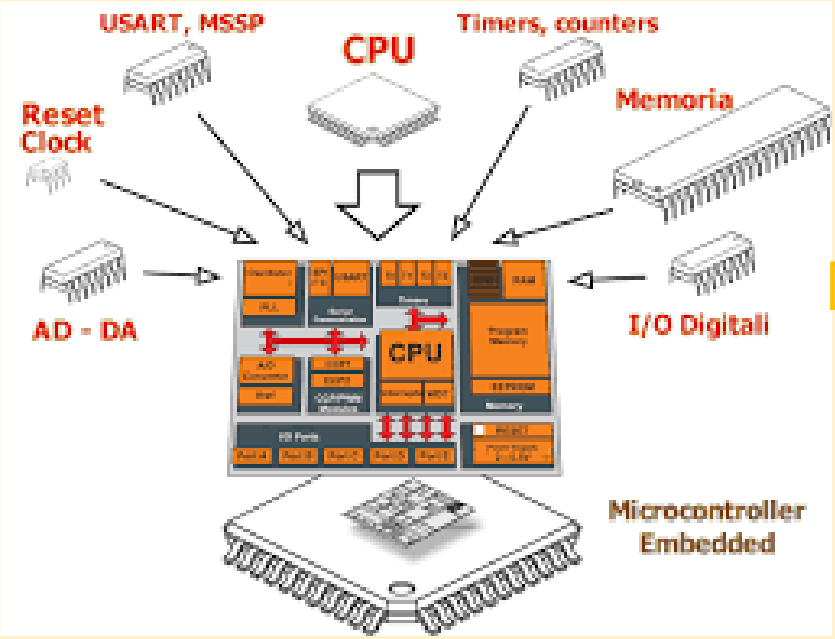
Il Microprocessore

*La classica struttura di un elaboratore è rappresentata dal modello di **Von Newmann.***
Dove la memoria contiene sia i dati che il programma, e condivide con le periferiche di input e output le linee dei BUS.



Il Microcontrollore

Consideriamo ora una schema a Microprocessore, composta come detto prima da CPU, memoria volatile, memoria non volatile e periferiche di ingresso ed uscita e realizziamola tutta dentro ad un circuito integrato. In questo modo otterremo un ***MICROCONTROLLORE****.*

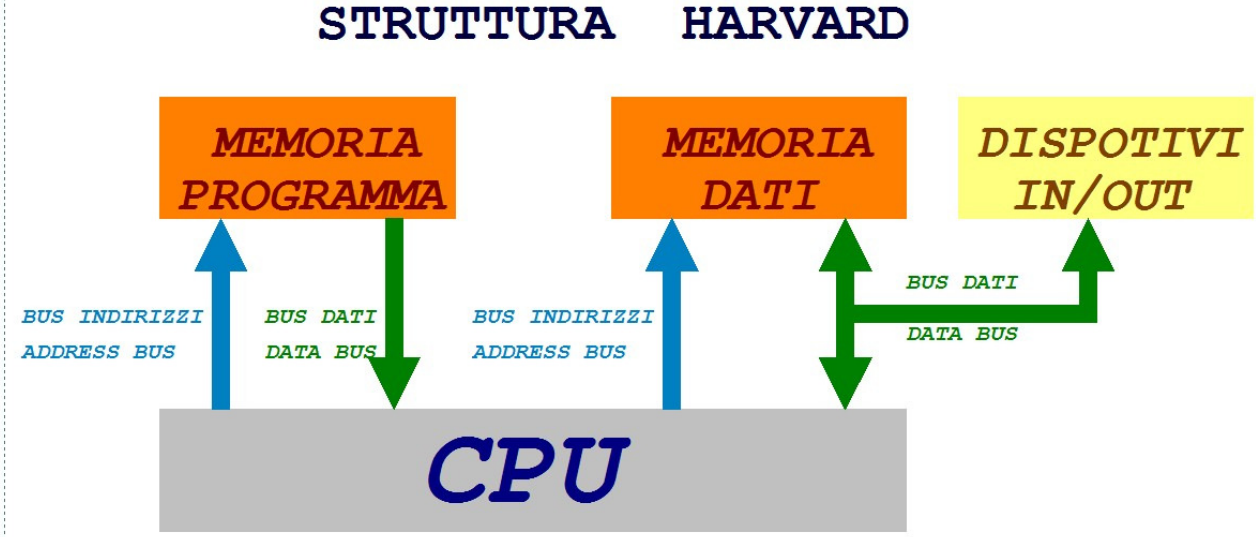


Anche se l'aspetto può essere simile, la differenza è ***sostanziale****.*

Il Microcontrollore

Le risorse di un microcontrollore, memoria, dispositivi di input/output, circuiti interni, sono ben definite e scelte dal costruttore. Anche la quantità di queste risorse non può essere paragonata a quella di un sistema a microprocessore, se possiamo avere un PC con una RAM da 16GB non avremo mai lo stesso quantitativo di RAM interna ad un microcontrollore.

*Inoltre l'architettura interna non è di tipo Von Newmann, ma di tipo **HARVARD**.*
*Dove abbiamo una quantità di memoria ben definita per i dati ed una per il programma, a fronte però di bus differenti che consentono **l'accesso contemporaneo alle due memorie**.*



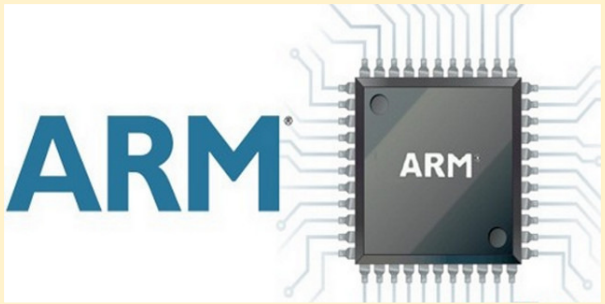
Il Microcontrollore

*Tra le marche più note di Microcontrollori troviamo la **MICROCHIP**, o la **ATMEL** (oggi acquisita dalla prima).*

*La Microchip è la famiglia dei famosi **PIC**, famosissimo lo storico **PIC16F84** uno dei microcontrollori ad 8 bit più conosciuto.*

*La **ATMEL** è invece la famiglia dei microcontrollori utilizzati nelle schede che oggi conosciamo tutti con il nome di **ARDUINO**.*

*Ovviamente esistono moltissime altre marche di microcontrollori e diverse tecnologie costruttive come ad esempio la più recente tecnologia **ARM**, utilizzata per le sue elevate prestazioni e bassi consumi, nel monto degli smartphone e tablet.*



Il Microcontrollore

Un'altra differenza sostanziale tra il Microprocessore ed il Microcontrollore è il set di istruzioni.

- *Nei Microprocessori, si utilizza la tecnologia CISC (Complex Instruction Set Computer) che è rappresentata da un set di istruzioni in grado di eseguire anche operazioni complesse. Il set di istruzioni in questo caso è composto anche da centinaia di istruzioni.*

- *Nei Microcontrollori si utilizza la tecnologia RISC (Reducted Instruction Set Computer) che è invece rappresentata da un set di istruzioni ridotto, in grado di eseguire solo operazioni basilari. Nel Microcontrollore menzionato prima il PIC16F84, ci sono in tutto 33 istruzioni.*

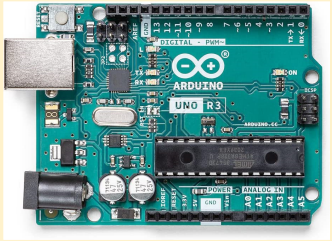
CISC		RISC
• Molte Istruzioni • Pochi registri • Numero di cicli di esecuzione per ogni istruzione	***VS***	• Poche Istruzioni • Molti registri • 1 o 2 cicli di esecuzione per ogni istruzione

Il Microcontrollore

Un'altra differenza la troviamo che il Microcontrollore trova facilmente utilizzo in applicazioni EMBEDDED, cioè applicazioni specifiche per il controllo di singoli dispositivi (es. router, antifurto, centralina telefonica, centralina automobile ecc…).

Il Microprocessore invece opera spesso all'interno di sistemi di elaborazione complessi insieme ad altri processori e componenti, con memorie di massa differenti (Hard disk, SSD) e quasi sempre con un Sistema Operativo (Windows 10, Linux, IOS ecc…).

SCHEDA ARDUINO UNO



SONO DUE MONDI COMPLETAMENTE DIVERSI CON APPLICAZIONI DIFFERENTI, PER QUESTO ***NON COMPARABILI***

SCHEDA MADRE PER PC

ARDUINO

*Nel 2003 ad Ivrea era operativa la **Interaction Design Institute,** una scuola post-laurea, nata dall'iniziativa di **Olivetti** e **Telecom Italia**. Nel corso della sua attività sono stati portati avanti diversi progetti software ed hardware, tra cui la nascita del progetto **Arduino**.*

Nel 2003 dei membri di questa scuola, cominciano a pensare ad una piattaforma hardware e software con cui realizzare veloci prototipi per scopi hobbystici, ma anche professionali.

*Il team composto da; **Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, e David Melli** realizzano così nel **2005** la **piattaforma hardware e software ARDUINO**. Il nome nasce da un bar ritrovo frequentato dagli ideatori del progetto, nome a sua volta derivante dall'omonimo Re d'Italia di Ivrea del 1002. Non possiamo perciò non considerare ARDUINO una creatura Italiana.*

ARDUINO

La scheda elettronica, è composta dalle seguenti parti:

- *Un microcontrollore ATMEL, cioè un un'unità di elaborazione e calcolo contenente una CPU, una RAM ed una FLASH (cioè una memoria non volatile)*
- *un circuito di alimentazione della scheda composto da un regolatore di tensione, che rende la scheda alimentabile da apposito alimentatore o da porta USB,*
- *un circuito convertitore USB-RS232,*
- *dei PIN (piedini di collegamento) connessi alle porte GPIO (**G**eneral **P**urpose **I**nput ed **O**utput) del microcontrollore*
- *dei PIN connessi agli ingressi analogici del microcontrollore.*

ARDUINO

*La piattaforma ARDUINO è composta da diverse schede, la più conosciuta si chiama ARDUINO UNO dotata di un microcontrollore della ditta ATMEL modello **ATMEGA328** funzionante con:*

- *un clock da **16MHz**,*
- *una RAM da **2kB** (memora volatile per i dati e le variabili)*
- *una Flash (la memoria non volatile che contiene il programma) da **32kB**,*
- *una **EEPROM** da 1kB (la EEPROM è anch'essa una memoria non volatile, utilizzata però solo per i dati).*

Come si può notare dai di frequenza del clock e dimensione della memoria, non possiamo pensare di installare su una scheda ARDUINO UNO, un sistema operativo, le sue risorse rispetto ai classici PC sono sicuramente inferiori anzi irrisorie.
*Questo tipo di scheda non deve comportarsi come un generico PC, ma eseguire delle operazioni ben definite e soprattutto **deve eseguire un solo programma alla volta**.*

ARDUINO

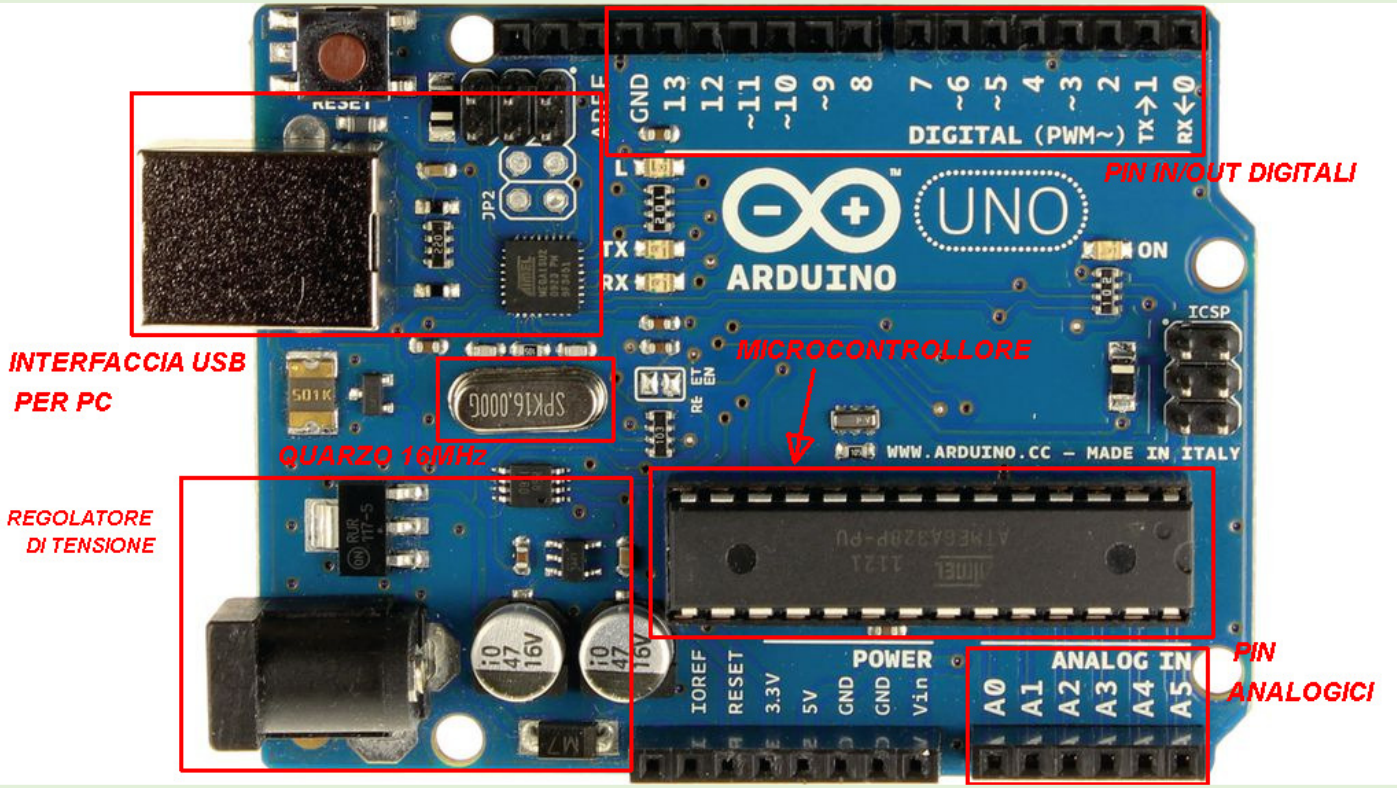
Come abbiamo detto, nel mondo ARDUINO esistono altre schede, e con esse si possono realizzare svariati progetti. Basta girare un po' in rete o anche su youtube e vedere come i tanti sviluppatori e "maker" hanno dato sfogo alla propria fantasia, realizzando a volte dei veri ***piccoli capolavori della tecnologia****.*

Sul sito ufficiale di ARDUINO al seguente link https://www.arduino.cc/en/Main/Products troviamo le varie schede disponibili.

Una parte importante e fondamentale della piattaforma ARDUINO è proprio il suo sito https://www.arduino.cc/ . Qui possiamo trovare tutte le informazioni tecniche ed una completa guida contenente tutte le informazioni hardware e software. Inoltre dal sito possiamo scaricare liberamente la parte software della piattaforma e cioè l'ambiente di sviluppo che andremo più avanti a descrivere https://www.arduino.cc/en/Main/Software .
La versione meno recente che utilizzeremo di seguito è scaricabile al seguente link:
https://drive.google.com/file/d/1NXjyVWlfy2EyqqC7XkOyGPPiagJFnMKC/view?usp=share_link

ARDUINO

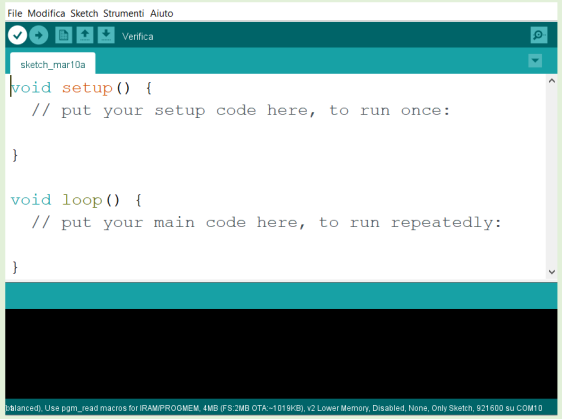
*Una delle schede più utilizzate, considerata un po' l'entry level del progetto, è la scheda **ARDUINO UNO**.*



ARDUINO

*Tutto l'ambiente è basato sul concetto di **open source**, troviamo perciò tutte le informazioni dettagliate sul software e sull'hardware, comprese anche degli schemi elettrici.*

*Perciò se decidiamo di acquistare un kit di ARDUINO ed entrare in questo mondo, il primo passa sarà quello di scaricare ed installare il suo ambiente di sviluppo software definito **I.D.E.** (Integrated Development Environment, ambiente di sviluppo integrato).*



ARDUINO

Cominciamo con il dire che la programmazione con la piattaforma ARDUINO, parte dai linguaggi C/C++, che sono ala base dell'ambiente di sviluppo fornito. Il suo linguaggio di programmazione si chiama ***WIRING****.*

Nessuno ci vieta di programmare la scheda ARDUINO UNO con altri linguaggi, ma ovviamente dovremo utilizzare altri ambienti ed altri programmatori.

Utilizzando l'ambiente di sviluppo fornito, rimane tutto più semplice e veloce, perché come già detto vengono fornite delle librerie già pronte per essere utilizzate sulla scheda, ed si può programmare la scheda senza alcun dispositivo esterno.

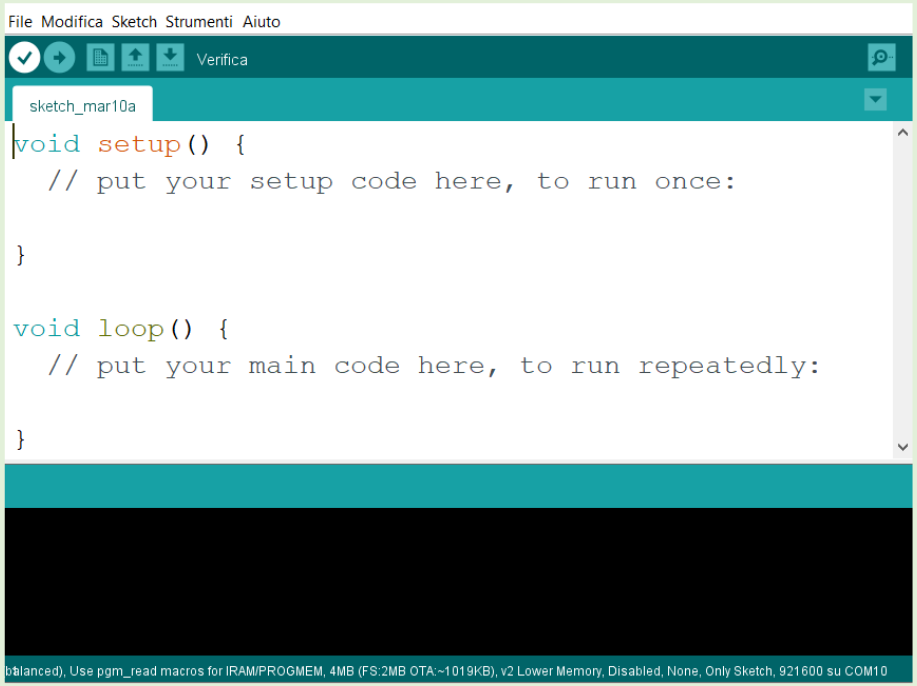
ARDUINO

*Guardando la schermata che appare all'avvio dell'IDE di Arduino, saltano subito all'occhio due parti del programma definite con **setup** e **loop**.*

*Entrambe sono seguite dalla scritta **void**.*

Chi conosce un po' il linguaggio C, conosce il significato di questa dicitura.

Chi non lo conosce può per ora ignorarla e ricopiarla così com'è.



Ci basta sapere che la parola void in inglese significa, vuoto, nullo. Nel caso specifico questo termine ci indica che la parte di programma che viene descritta successivamente tra le parentesi { }, non restituisce nessun risultato, viene eseguita e basta.

ARDUINO

*La parte di programma chiamata **setup,** dovrà essere scritta tra le prime due parentesi graffe e, come recita la scritta in inglese, verrà eseguita una volta sola all'avvio.*

"put your setup code here, to run once"

*Ci viene perciò consigliato di inserire tra le due parentesi tutto ciò che vogliamo far **eseguire una sola volta all'avvio del programma**, come ad esempio i settaggi o le impostazioni iniziali.*

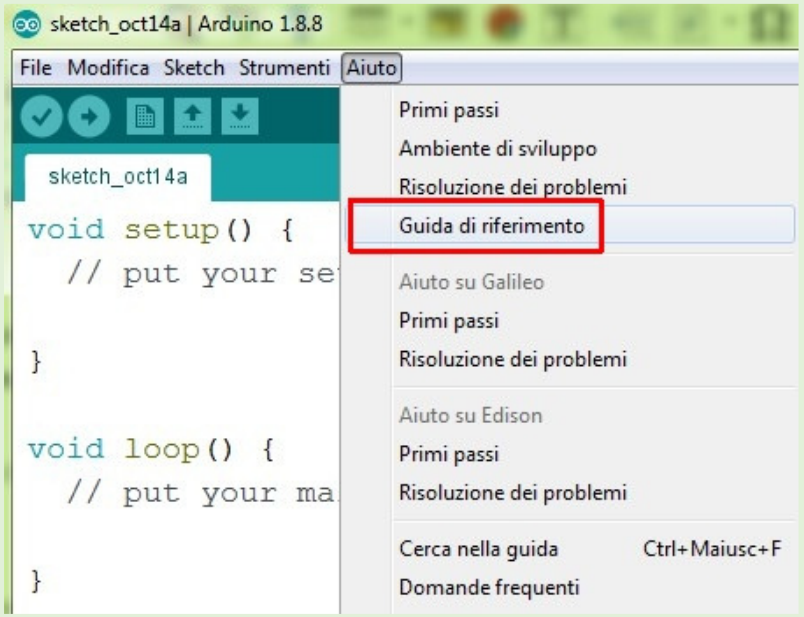
*La parte di programma chiamata **loop**, viene invece eseguita ripetutamente e ciclicamente dopo l'avvio.*

"put your main code here, to run repeatedly"

*Quello che inseriremo tra le due parentesi indicate dal ciclo loop, **verrà invece eseguito ciclicamente**.*

ARDUINO

Possiamo a questo punto accedere alla guida online messa a disposizione e studiare i vari comandi e le funzioni.
Per fare questo basta accedere alla guida dal menu help.

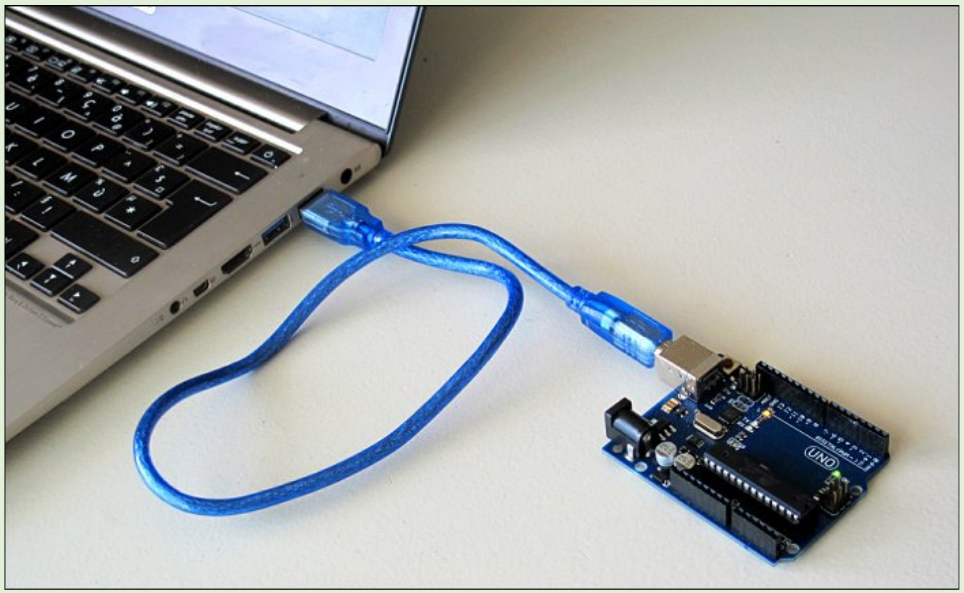


Ma prima di consultare la preziosa guida, consiglio di provare a lanciare il più semplice programma di esempio mandandolo in esecuzione sula scheda.

ARDUINO

Le operazioni da fare sono perciò le seguenti:

- *Dopo aver installato il software, collegare la scheda ARDUINO UNO al PC tramite il cavetto USB.*
- *Lanciare il software di ARDUINO ed aprire l'esempio **Blink**, sotto alla sezione **Basic**.*



*A questo punto verrà aperta una nuova finestra contenente il programma **Blink**. Nel linguaggio C/C++ tutto ciò che è compreso tra i caratteri /* e */ rappresenta un commento, come anche tutto ciò che in una riga viene preceduto dai caratteri //.*

All'inizio del programma troveremo perciò una premessa per descrivere il programma compresa tra i caratteri / e */ . Successivamente troveremo altri commenti preceduti dai caratteri //, tutti i commenti avranno un colore grigio per indicare che non sono istruzioni del programma.*

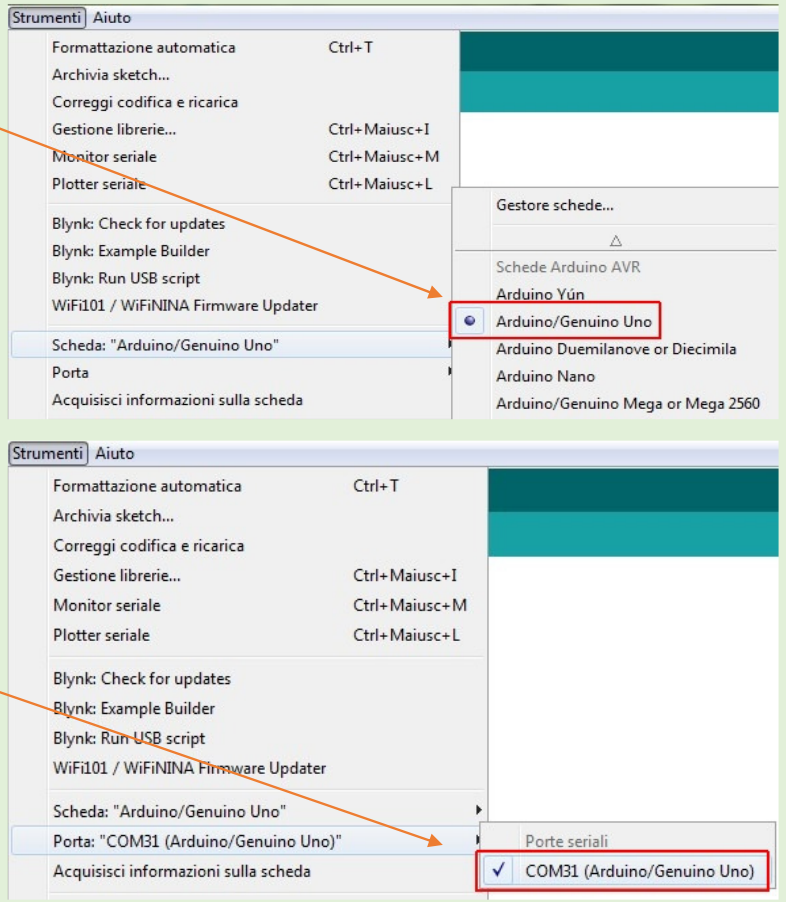
ARDUINO

Prima di caricare il programma dobbiamo indicare sull'IDE la scheda che stiamo utilizzando e la porta seriale utilizzata.

Il PC comunicherà con la scheda, tramite la porta USB collegata al microcontrollore con un convertitore USB-Seriale RS232.

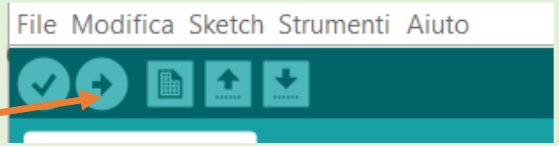
Una volta collegata la scheda, il PC vedrà perciò una porta Seriale RS232 indicata con la scritta **COM** seguita da un numero.

Dobbiamo perciò innanzitutto verificare che nel menù strumenti sia selezionata la scheda giusta e la COM indicata con ARDUINO UNO.

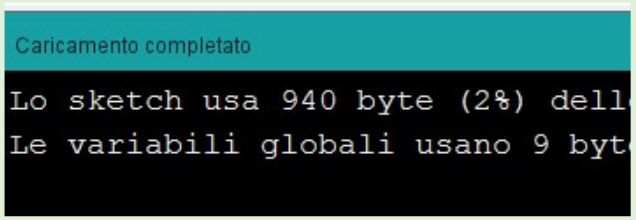


ARDUINO

Una volta eseguite queste impostazioni, possiamo avviare la compilazione (traduzione del programma in linguaggio macchina cioè codice binario) e la programmazione della scheda con l'apposito pulsante.



Se non ci saranno errori di sintassi, o di altro genere, nella finestra in basso vedremo un riepilogo della memoria usata, e non verrà indicato nessun errore.



A questo punto vedremo il led lampeggiare sulla scheda.

ARDUINO

*Analizziamo ora il programma. Cominciamo con il dire che la parola **LED_BUILTIN** è associata al numero **13**, nel senso che il compilatore sostituirà a questa parola il numero associato, che corrisponde ad uno dei terminali della scheda e che è internamente connesso ad un LED di test presente sulla scheda.*

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

I due programmi infatti darebbero lo stesso identico risultato.

ARDUINO

Nella parte SETUP viene messa la seguente istruzione. `pinMode(13, OUTPUT);`

Andando a cercare sulla guida troviamo quanto segue.

Possiamo facilmente intuire che in questo modo abbiamo impostato il piedino numero 13 come un'uscita.
Perciò d'ora in avanti potremo fare in modo, con le apposite istruzioni, che sul terminale 13 della scheda ci possa essere un livello di tensione basso oppure alto.

Description

Configures the specified pin to behave either as an input or an output. See the description of digital pins for details on the functionality of the pins.

Syntax

pinMode(pin, mode) **sintassi corretta per eseguire la funzione pinMode**

Parameters

pin: the number of the pin whose mode you wish to set **(numero del piedino che vogliamo impostare)**

mode: INPUT, OUTPUT, or INPUT_PULLUP. (see the digital pins page for a more complete description of the functionality.) **(tipo di impostazione: INPUT, OUTPUT, INPUT_PULLUP)**

Returns

None **(risultato restituito, nessuno)**

ARDUINO

All'interno del ciclo LOOP troviamo invece queste istruzioni `digitalWrite(13, HIGH);` `digitalWrite(13, LOW);`

Sempre cercando sulla guida troviamo quanto segue.

Description

Write a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

La descrizione in inglese è molto chiara,
*in pratica **digitalWrite**, consente di mettere*
*a livello alto o basso un terminale definito precedentemente con **pinMode,** come uscita.*

Syntax

digitalWrite(pin, value)

Parameters

pin: the pin number

value: HIGH or LOW

Returns

none

Il livello alto corrisponde ad una tensione di 5V o 3,3V a seconda del tipo di scheda, il livello basso corrisponde invece a 0 Volt.
La sintassi prevede che all'interno delle parentesi venga indicato il numero del terminale (nel nostro caso 13) e dopo la virgola il suo stato HIGH o LOW (alto o basso).

ARDUINO

*Un'altra funzione contenuta nel programma è il **delay**.*

```
delay(1000);
```

La spiegazione presente nella guida è la seguente.

In pratica è un ritardo inserito tra una riga e l'altra del programma. Quando il microcontrollore esegue questa istruzione, il programma è come se si fermasse per non fare nulla, fino alla scadenza del tempo impostato.

Description

Pauses the program for the amount of time (in miliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

Ferma il programma per il tempo indicato tra le parentesi della funzione. Espresso in millisecondi.

Syntax

delay(ms)

Parameters

ms: the number of milliseconds to pause (*unsigned long*)

Returns

nothing **La funzione viene eseguita e non vene restituito alcun risultato.**

ARDUINO

Proviamo ora ad effettuare una piccola modifica, aggiungiamo un secondo LED da far lampeggiare sul terminale 12, e facciamo in modo che venga fatto lampeggiare il primo LED per 5 volte ed il secondo per 3 volte.

Per fare questo, viene utilizzata una variabile chiamata «conteggio» di tipo intero (min -32768 max +32767).

Dentro al loop sono stati inseriti due cicli while per fare in modo di eseguire il lampeggio del primo e secondo LED un numero di volte definito dalla variabile.

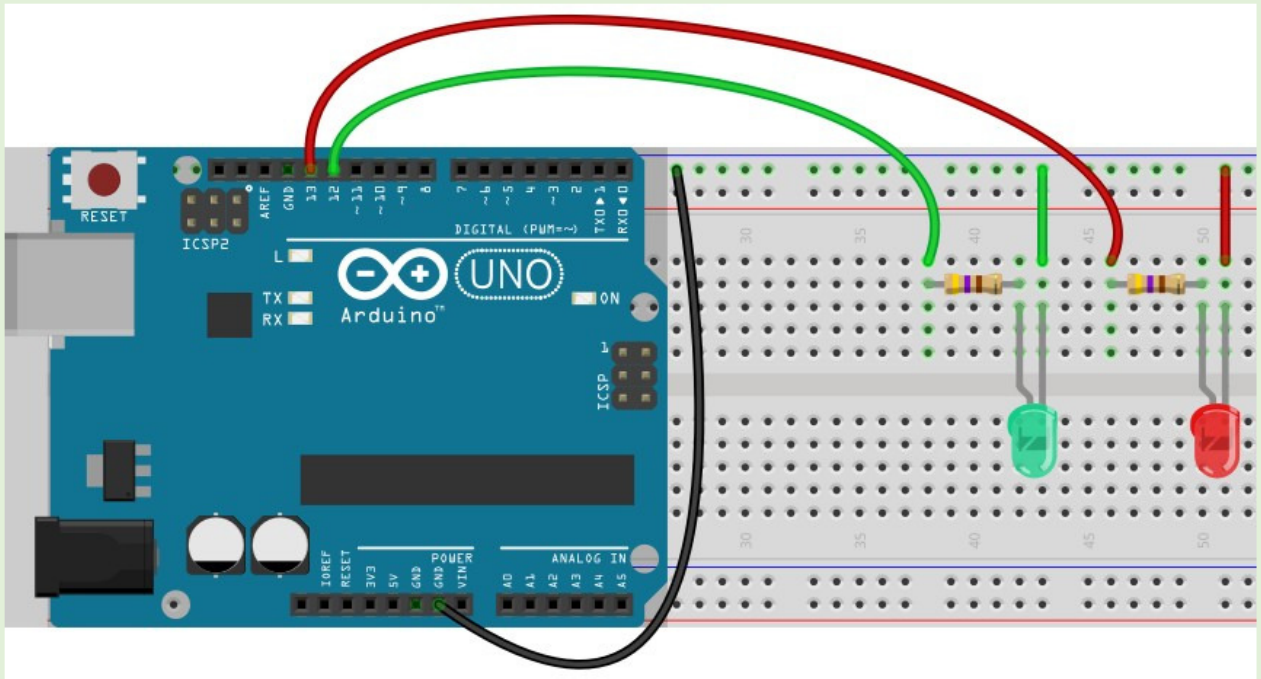
```
int conteggio;  //dichiaro una variabile intera

void setup() {
  pinMode(13, OUTPUT); //impostazione pin 13 come uscita
  pinMode(12, OUTPUT); //impostazione pin 12 come uscita
}

void loop() {
  conteggio=5;       //inizializzo il valore della variabile
  while(conteggio>0){ //controllo se eseguire il ciclo while
     digitalWrite(13, HIGH);   //accendo il LED1
     delay(500);               //attendo un secondo
     digitalWrite(13, LOW);    //spengo il LED1
     delay(500);               //attendo un secondo
     conteggio--;     //decremento la variabile conteggio
   }//while
  conteggio=3;       //inizializzo il valore della variabile
  while(conteggio>0){  //controllo se eseguire il ciclo while
     digitalWrite(12, HIGH);   //accendo il LED2
     delay(500);               //attendo un secondo
     digitalWrite(12, LOW);    //spengo il LED2
     delay(500);               //attendo un secondo
     conteggio--;    //decremento la variabile conteggio
   }//while
}
```

ARDUINO

Lo schema di collegamento del precedente esempio è il seguente. Una volta programmata la scheda con la stessa modalità vista prima, il programma rimarrà caricato nella scheda ***fino alla prossima riprogrammazione.***



MICROCONTROLLORI PIC

La scheda Arduino vista precedentemente, potrebbe essere il primo approccio per entrare nel mondo dei sistemi Embedded, anche se di fatto l'utilizzo che viene fatto della scheda non ci permette di gestire il microcontrollore ad un livello più basso, lavorando con i registri interni.

Utilizzando invece il componente «nudo» senza una scheda di supporto e magari senza una programmazione evoluta come quella di Arduino, i problemi da affrontare sono diversi e la conoscenza del microcontrollore è più approfondita.

In questa parte vedremo un tipo di microcontrollore molto diffuso che è quello della famiglia Microchip cioè il PIC.

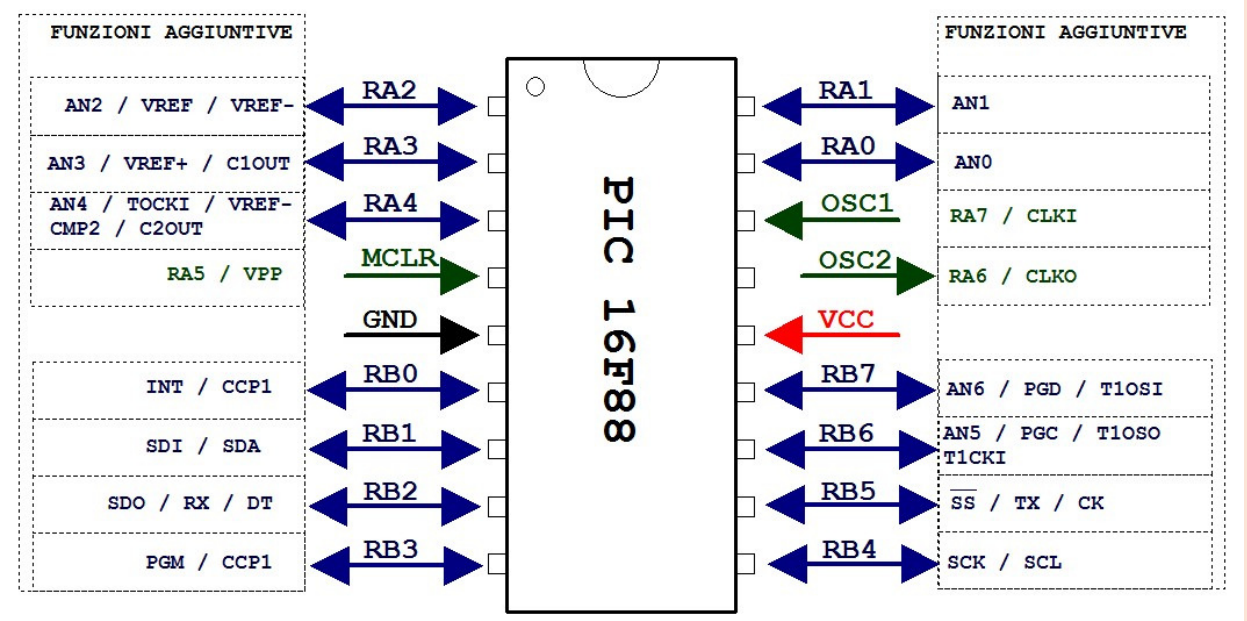
MICROCONTROLLORI PIC

*Come detto precedentemente in un microcontrollore, troviamo un **sistema di elaborazione completo** corredato anche di circuiti hardware con funzioni specifiche.*

- ***CPU C**entral **P**rocessing **U**nit ovvero unità centrale di elaborazione.*
- ***Memoria FLASH** Una memoria **non volatile** e riscrivibile dove allocare il programma da eseguire.*
- ***EEPROM E**rasable **El**ectrically **P**rogrammable **R**ead **O**nly **M**emory ovvero una memoria **non volatile** e riscrivibile a cui si può accedere per il salvataggio di dati, utilizzabile dal programma.*
- ***RAM R**andom **A**ccess **M**emory, ovvero la memoria **volatile** dove salvare le variabili utilizzate. In questa memoria ci sono allocati anche i registri per applicazioni specifiche, come **PORTA**, **PORTB**, **TRISA**, **TRISB**, **STATUS**, ecc...*
- ***GPIO** Linee di Input Output (**G**eneral **P**urpose **I**nput **O**utput). Nel microcontrollore sono già implementati dei circuiti che consentono di controllare direttamente i piedini del componente, e di configurarli, come digitali o analogici, o come ingresso o uscita.*
- ***DISPOSITIVI AUSILIARI** Nel microcontrollore ci sono altri circuiti elettronici già realizzati, quali; timer, contatori, whatchdog (timer che resetta periodicamente il programma), circuiti per la comunicazione seriale, convertitori analogico- digitali ecc....*

MICROCONTROLLORI PIC

In figura è rappresentato il PIC16F88, un Microcontrollore ad 8 bit con 18 pin.



*I piedini **BLU** rappresentano le linee di ingresso/uscita **GPIO (General Purpose Input Output)** e possono anche avere più di una funzione.*

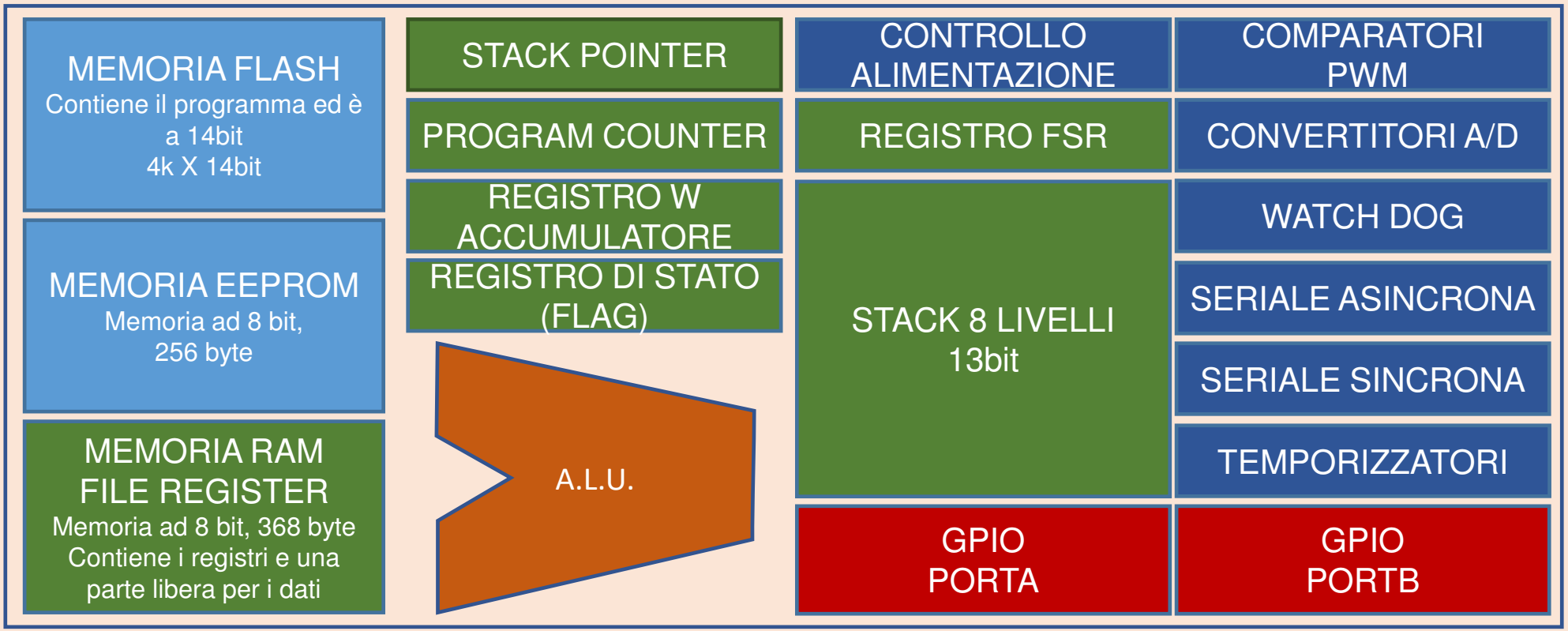
*I piedini **VERDI** si utilizzano per collegare un **CLOCK** esterno e un segnale di reset, anch'essi possono hanno la possibilità di essere utilizzati come semplici input o output.*

*Ai piedini chiamati **OSC1** e **OSC2** va collegato un circuito di clock, spesso realizzato con un quarzo. Il piedino **MCLR** rappresenta il **Master Clear** e cioè il **reset.***

*Il pin **Vdd** rappresenta l'alimentazione di 5Vdc e **Vss** il negativo dell'alimentazione.*

MICROCONTROLLORI PIC

Di seguito il contenuto di un microcontrollore PIC16F88, in azzurro chiaro le memorie non volatili in verde quelle volatili.



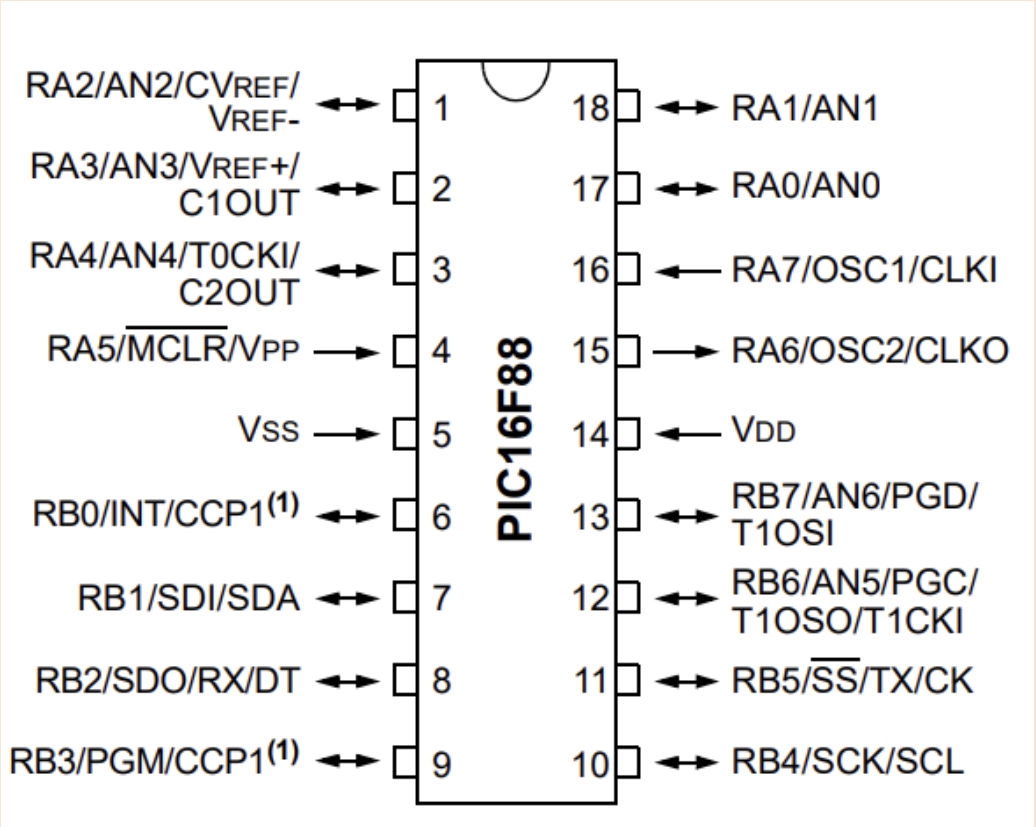
MICROCONTROLLORI PIC

Il punto di forza del Microcontrollore è proprio quello di avere un sistema di elaborazione completo di circuiti hardware controllati sempre a livello software.

In pratica scegliendo il microcontrollore adatto, possiamo realizzare una scheda elettronica su misura per la nostra applicazione.

Inoltre essendo ormai tutti i microcontrollori dotati di un clock interno, il dispositivo può essere semplicemente alimentato senza la necessità di altri componenti esterni.

Questo significa anche che possiamo realizzare qualsiasi funzione che lega degli ingressi a delle uscite, con un unico dispositivo integrato.



MICROCONTROLLORI PIC

Vediamo ora la struttura della memoria RAM che viene definita FILE REGISTER.

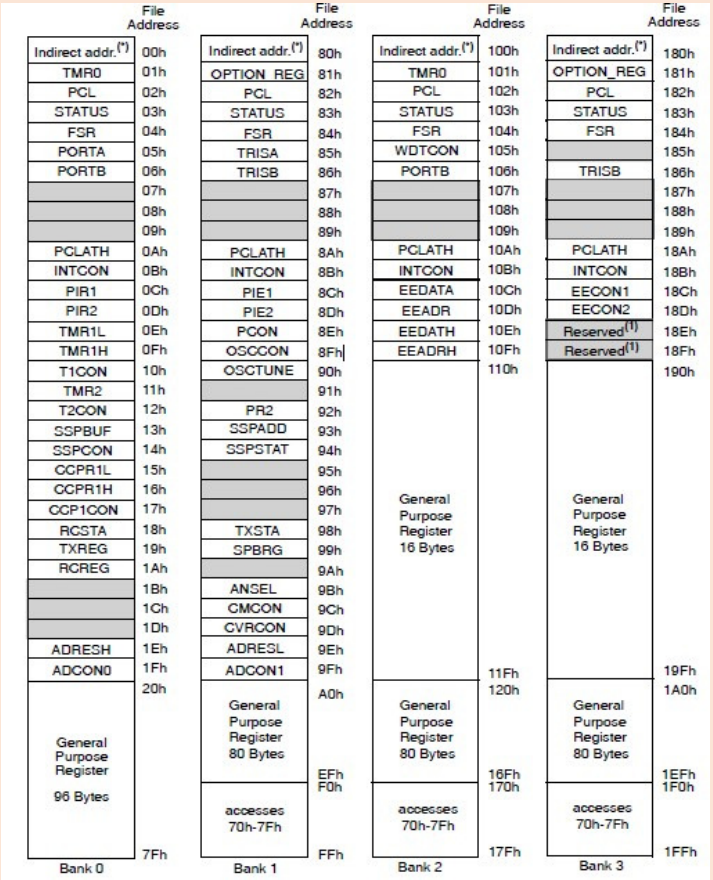
*Nel PIC16F88, la RAM è divisa in **4 banchi**, ognuno dei quali contiene dei registri ad 8 bit di uso specifico, e delle zone libere da utilizzare per le variabili del programma.*

Per spostarsi sui 4 banchi, occorre agire sui bit RP1 e RP0 del registro **STATUS**.

STATUS: ARITHMETIC STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

RP1 – bit 6 di STATUS	RP0 – bit 5 di STATUS	Banco di memoria accessibile
1	1	Bank3 (180 Hex – 1FF Hex)
1	0	Bank2 (100 Hex – 17F Hex)
0	1	Bank1 (80 Hex – FF Hex)
0	0	Bank0 (0 Hex – 7F Hex)



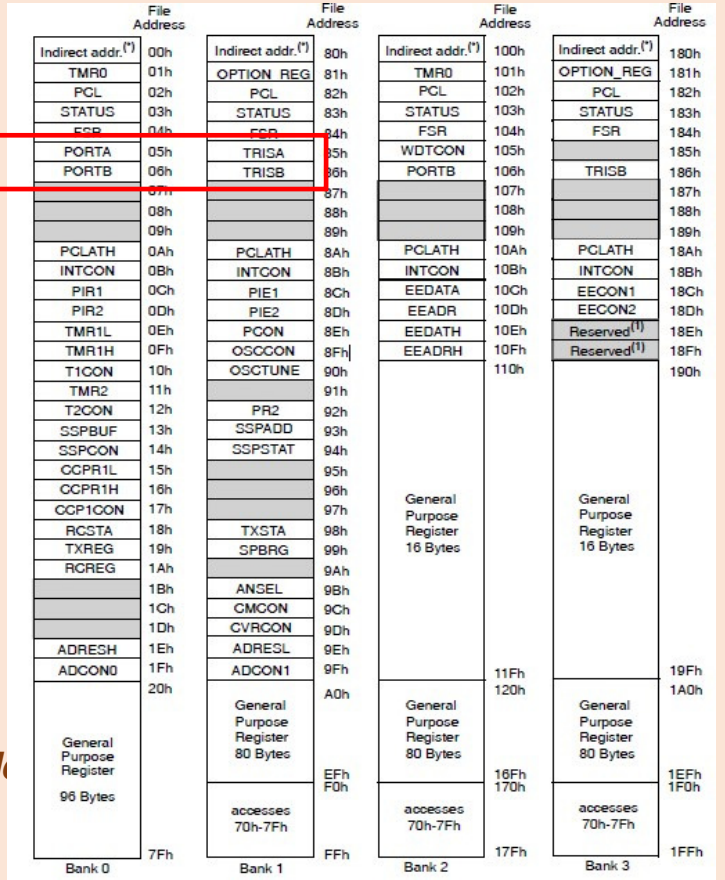
MICROCONTROLLORI PIC

*Tra i vari registri troviamo i 4 registri che ci consentono di gestire i terminali GPIO, e cioè **PORTA**, **PORTB**, **TRISA** e **TRISB**.*

E più esattamente sui registri TRIS andiamo a definire la direzione dei vari GPIO (ingresso o uscita) e sui registri PORT andiamo a scrivere o leggere il valore logico (e di conseguenza di tensione) presente sui terminali del componente.

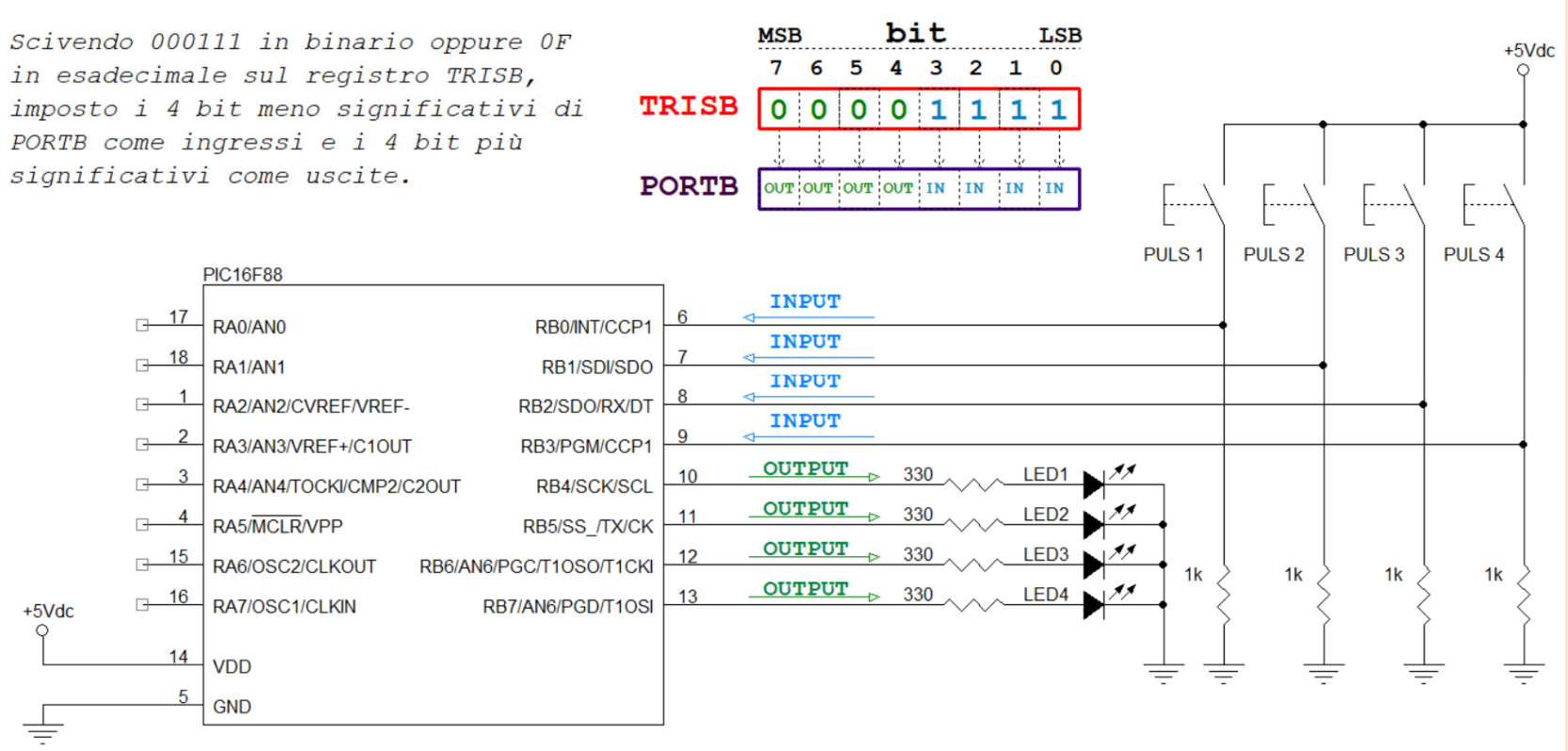
Un valore 0 scritto su un bit del registro TRIS, fa diventare il corrispondente bit del registro PORT un uscita, e viceversa un valore 1 scritto su TRIS, fa diventare Il corrispondente bit del registro PORT un ingresso.

Il valore dei pin identificati con PORTA e PORTB sono esattamente corrispond con il valore dei due registri



MICROCONTROLLORI PIC

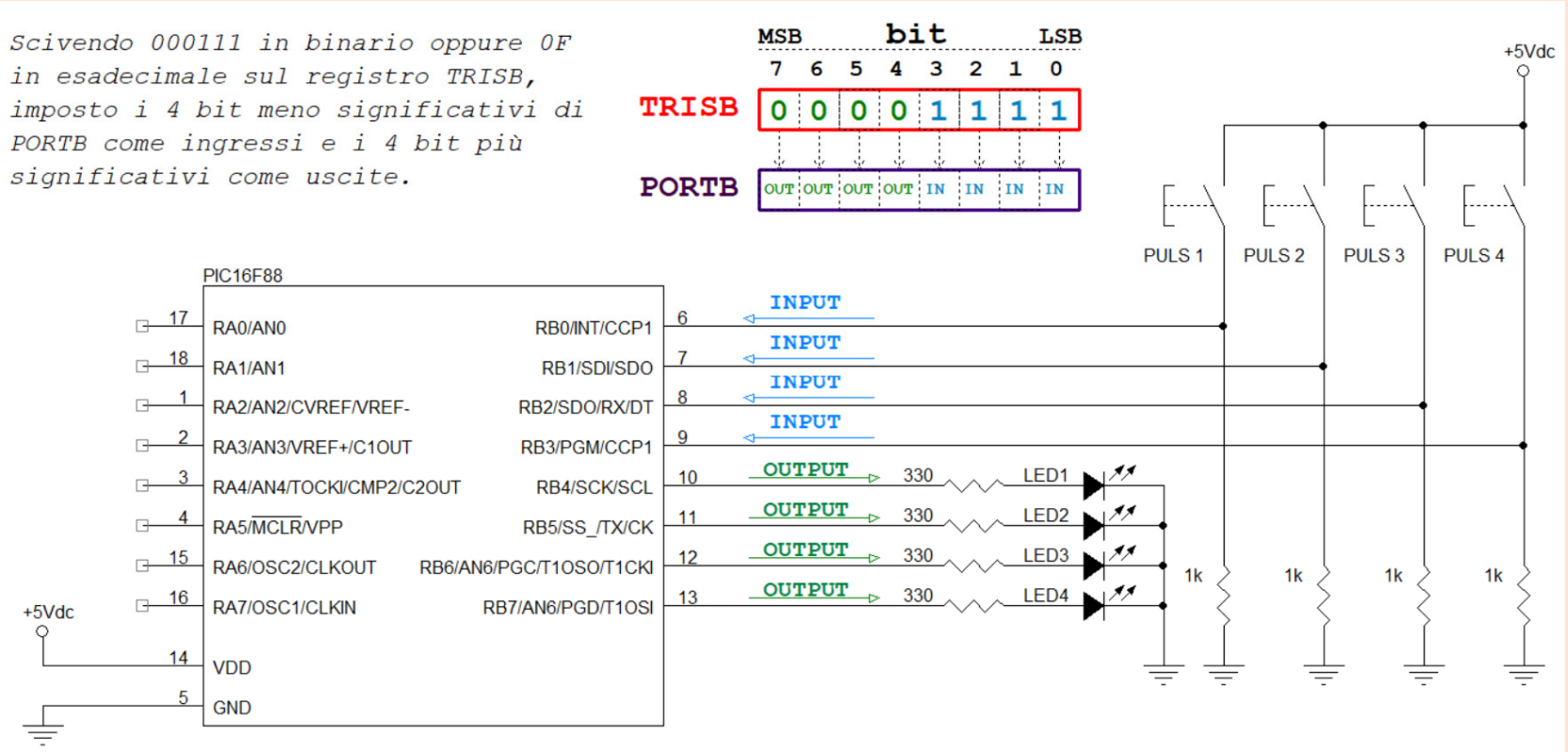
Nello schema seguente ad esempio, dovremo configurare come INGRESSI i terminali collegati ai PULSANTI e come USCITE i terminali collegati ai LED. In questo caso per configurare correttamente PORTB dovremo scrivere 00001111 in TRISB.



MICROCONTROLLORI PIC

Una volta configurato il registro TRISB, posso leggere i 4 bit RB0, RB1, RB2 ed RB3 per conoscere lo stato deu 4 pulsanti (RB0=1 se PULS 1 è premuto). E posso accendere o spegnere i LED scrivendo 1 o 0 su RB4, RB5, RB6 o RB7.

Scivendo 000111 in binario oppure 0F in esadecimale sul registro TRISB, imposto i 4 bit meno significativi di PORTB come ingressi e i 4 bit più significativi come uscite.



MICROCONTROLLORI PIC

*Un altro REGISTRO importante per la gestione delle GPIO, è **ANSEL**.*

REGISTER 12-1: ANSEL: ANALOG SELECT REGISTER (ADDRESS 9Bh) PIC16F88 DEVICES ONLY

U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
bit 7							bit 0

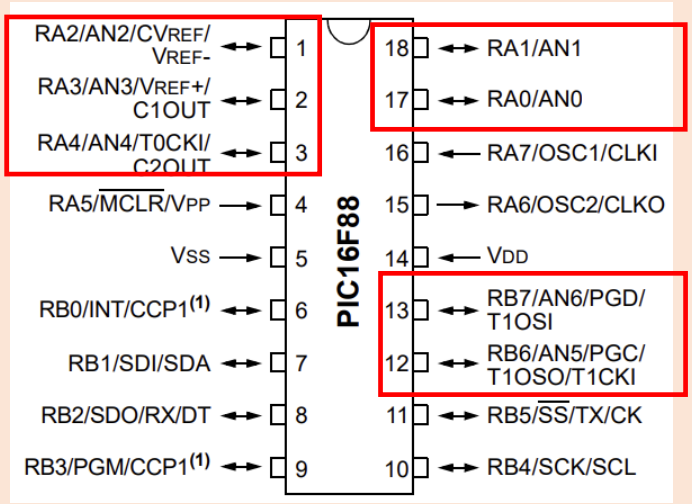
bit 7 **Unimplemented:** Read as '0'

bit 6-0 **ANS<6:0>:** Analog Input Select bits
Bits select input function on corresponding AN<6:0> pins.
1 = Analog I/O(1,2)
0 = Digital I/O

TXREG	19h	SPBRG	99h	16 Bytes	16 Bytes
RCREG	1Ah		9Ah		
	1Bh	ANSEL	9Bh		
	1Ch	CMCON	9Ch		
	1Dh	CVRCON	9Dh		
ADRESH	1Eh	ADRESL	9Eh		

*Con questo registro andiamo a definire il funzionamento dei PIN che hanno la possibilità di funzionare come ingressi **ANALOGICI**, cioè tutti quei pin dove troviamo la scritta **AN**.*

Scrivendo 1 su un bit di ANSEL configureremo il relativo pin come ingresso analogico, scrivendo invece 0 il pin sarà un ingresso/uscita digitale, la cui direzione verrà impostata come visto prima con i registri TRIS.



MICROCONTROLLORI PIC

La presentazione proseguirà con la parte relativa all'hardware del microcontrollore e del software.

In attesa di completamento, vi invito a visionare i seguenti tutorial:

https://danielepostacchini.it/wp-content/uploads/2019/04/Microcontrollori.pdf

https://danielepostacchini.it/wp-content/uploads/2019/10/Microcontrollori_IO-2.pdf

https://danielepostacchini.it/wp-content/uploads/2019/10/Hardware-microcontrollori.pdf

https://danielepostacchini.it/wp-content/uploads/2019/01/mikroelektronika.pdf

https://danielepostacchini.it/wp-content/uploads/2019/09/mikroC_.pdf

https://danielepostacchini.it/wp-content/uploads/2019/11/Il-linguaggio-C-per-la-programmazione-dei-microcontrollori-parte1.pdf

https://danielepostacchini.it/wp-content/uploads/2020/04/Interrupt.pdf

https://danielepostacchini.it/wp-content/uploads/2020/05/antirimbalzo.pdf

https://danielepostacchini.it/wp-content/uploads/2020/05/Gestione-di-un-men%C3%B9-su-LCD_completo.pdf