



SAPIENZA  
UNIVERSITÀ DI ROMA

# Microcontrollori

Automazione

Vincenzo Suraci



## STRUTTURA DEL NUCLEO TEMATICO

- MICROCONTROLLORI
- CARATTERISTICHE GENERALI
- PERIFERICHE INTEGRATE
- PROGRAMMAZIONE
- PROGRAMMABLE INTERFACE COMPUTER (PIC)
- ARDUINO



SAPIENZA  
UNIVERSITÀ DI ROMA

Corso di Laurea: INGEGNERIA  
Insegnamento: AUTOMAZIONE  
Docente: DR. VINCENZO SURACI

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

# MICROCONTROLLORI



## MICROCONTROLLORI

### DEFINIZIONE

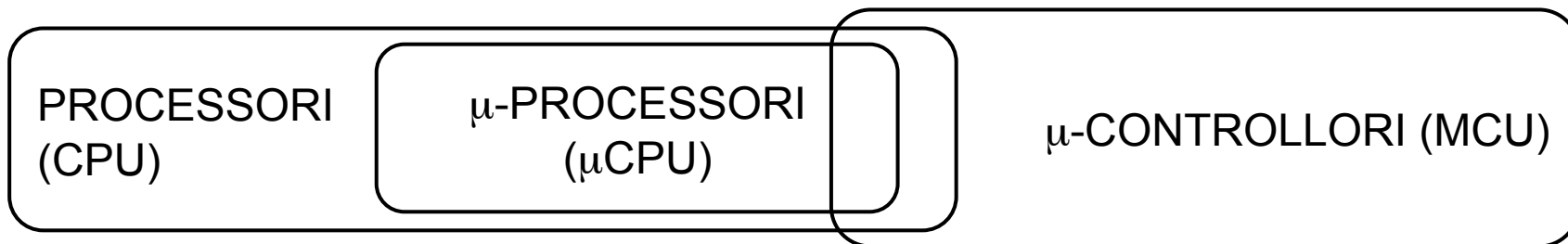
Dicesi **PROCESSORE** una tipologia di **dispositivo hardware** dedicato **all'esecuzione di istruzioni**.

### DEFINIZIONE

Dicesi **MICRO-PROCESSORE** una tipologia di **processore** la cui struttura **hardware è interamente contenuta** in un **circuito integrato**.

### OSSERVAZIONE

Un **MICRO-CONTROLLORE** contiene un **(micro-)processore** con un **set di istruzioni ridotto** ed alcune **periferiche dedicate**.





## MICROCONTROLLORI

### REQUISITI

Un **microcontrollore** è chiamato in generale a rispondere ai seguenti **requisiti**.

Requisito	Caratteristica	Beneficio
Gestione ingressi / uscite	Porte di I/O con controllo fino al singolo bit	Controllo efficiente di dispositivi esterni quali <b>attuatori</b> , teleruttori etc.
Comunicazione con Periferiche esterne	Porta seriale, SPI, I <sup>2</sup> C, UART, CAN, etc.	Estensione delle funzionalità con l'uso di <b>periferiche esterne</b>
Controllo di motori ed attuatori	Timer, Contatori, PWM	Facilità di <b>programmazione</b>
Gestione di programmi logico sequenziali	Salti condizionati, istruzioni logiche, etc.	Facilità di <b>realizzazione</b> del software <b>logico-sequenziale</b>
Reazione ad eventi	Gestione degli IRQ prioritari	Facilità nella <b>realizzazione</b> di sistemi <b>Real Time</b>
Acquisizione dati da sensori	ADC	Facilità di <b>installazione</b> in ambienti <b>pre-esistenti</b>



# MICROCONTROLLORI

## SETTORI DI APPLICAZIONE

### Consumer Electronics

Telefoni cellulari, tablet, orologi, registratori, calcolatrici, mouse, tastiere, modem, fax, schede sonore, caricatori di batterie

### Building Automation

serrature per porte, sistemi di allarme, termostati, condizionatori, telecomandi, VCR, frigoriferi, exercise equipment, lavatrici, forni a micro-onde, consolle, inverter fotovoltaici

### Automotive

Centraline elettroniche, ABS, navigatore satellitare, entertainment, etc.

### Settore industriale

Controllo di assi (posizione, velocità), Regolatori ON-OFF, Regolatori PID, etc.



## MICROCONTROLLORI

### VANTAGGI RISPETTO AI PROCESSORI GENERAL PURPOSE

1. I microcontrollori sono **derivati dai microprocessori**, mantenendone le caratteristiche peculiari, ma con un **set di istruzioni ridotto**.
2. I microcontrollori permettono un **utilizzo più semplice** e specifico nelle applicazioni industriali dove **molte istruzioni dei microprocessori non vengono utilizzate**.
3. I microcontrollori hanno subito grandi evoluzioni tanto da diventare anche «più potenti» dei microprocessori, mantenendo un **costo minore** o uguale e un **utilizzo più rapido ed intuitivo**.
4. La **velocità di esecuzione** delle operazioni integrate nei microcontrollori è **nettamente maggiore** rispetto a quelle eseguite via software dai microprocessori.



## MICROCONTROLLORI

### VANTAGGI RISPETTO AI CONTROLLORI A BUS

1. Sono richiesti un **numero inferiore di dispositivi "discreti"** per la realizzazione di un sistema di controllo o di automazione
2. Il sistema riesce ad avere **dimensioni ridotte**
3. **Costi contenuti** (i dispositivi ed il core costano qualche €)
4. **Consumo di energia inferiore** (i device on chip hanno un consumo minore dei device esterni)
5. Si abbassa la **sensibilità ad interferenze EM** data la minor estensione della circuiteria di connessione (che fa da antenna ricettiva)
6. **il sistema nel complesso è più affidabile** dato che sono interconnessi **pochi componenti** (saldature, gradienti di temperatura locali, auto-interferenze, etc.)





## MICROCONTROLLORI

### VANTAGGI RISPETTO AI CONTROLLORI DEDICATI

1. A differenza dei controllori dedicati, **eseguono istruzioni** pertanto:
  - possono **eseguire elaborazioni complesse**
  - possono **comunicare con altri dispositivi**
  - possono essere **ri-programmati**;
2. Come i controllori dedicati garantiscono **protezione contro le copie**
  - la maggiore parte del single-chip offre la possibilità di proteggere da lettura il programma contenuto nella ROM
3. A differenza dei controllori dedicati, hanno funzioni avanzate di **risparmio energetico**
  - le versioni CMOS supportano il modo di funzionamento **stand-by**: è possibile bloccare, via software, attività della CPU e quindi ottenere correnti di alimentazione molto basse



SAPIENZA  
UNIVERSITÀ DI ROMA

Corso di Laurea: INGEGNERIA  
Insegnamento: AUTOMAZIONE  
Docente: DR. VINCENZO SURACI

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

# CARATTERISTICHE GENERALI



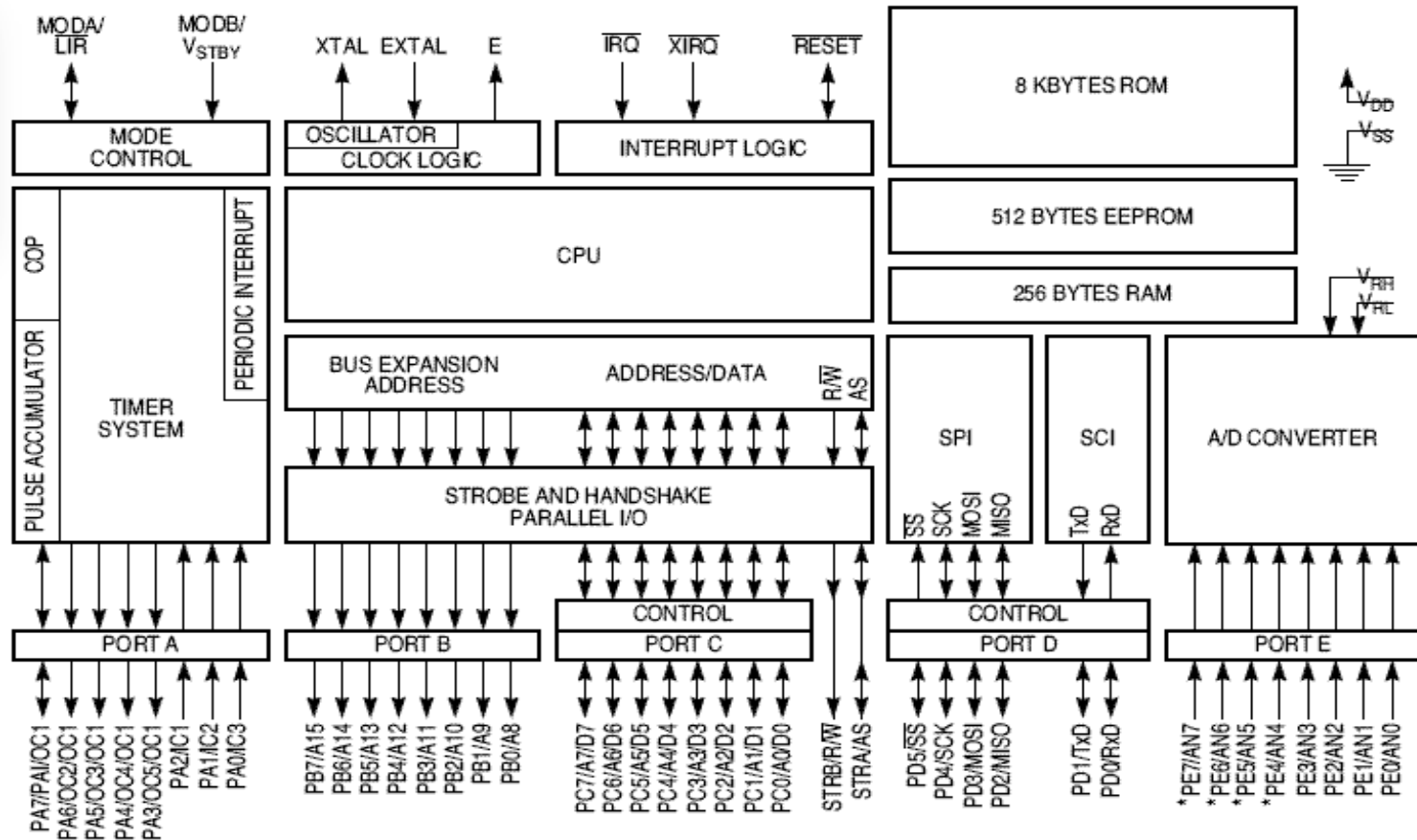
# LAYOUT LOGICO



MOTOROLA  
68HC11x Family



Approx 70 MCU  
68HC11x  
per modello



\* NOT BONDED ON 48-PIN VERSION.

A8 BLOCK



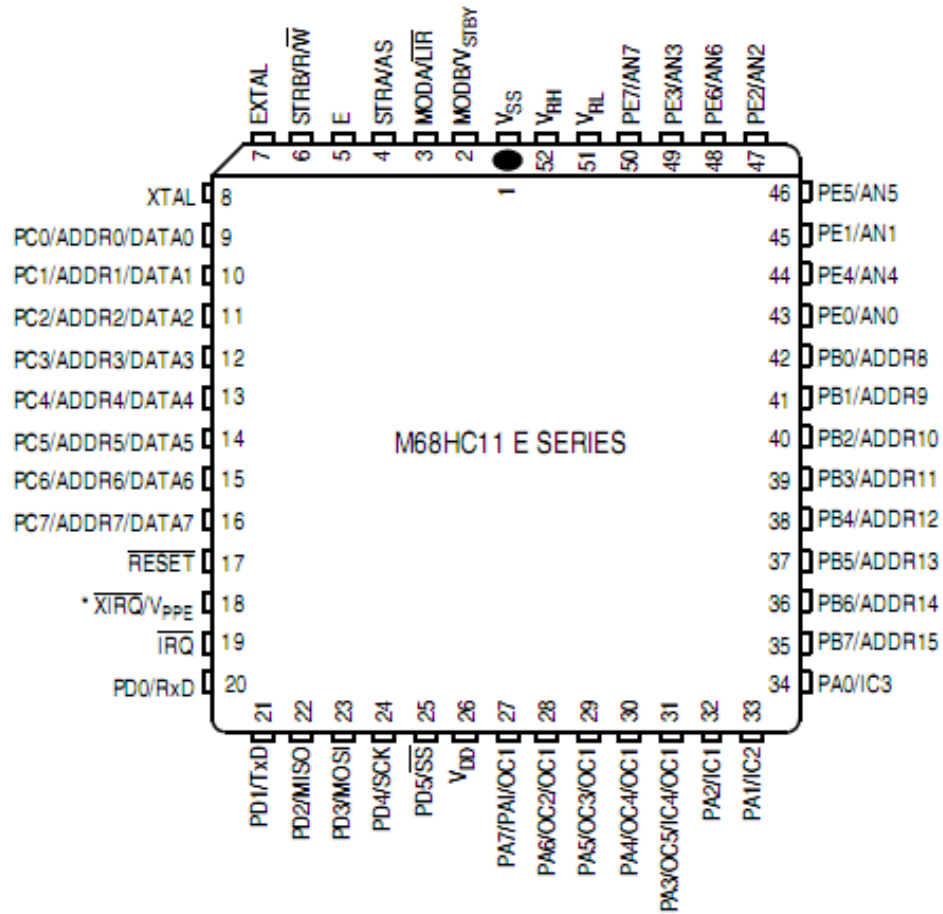
# LAYOUT FISICO



MOTOROLA  
68HC11x Family



Approx 70 MCU  
68HC11x  
per modello



\* V<sub>PPE</sub> applies only to devices with EPROM/OTPROM.



## PROCESSORE

- **frequenza di clock:** da pochi Khz a qualche Ghz
- **numero di core:** 1, 2, anche 3 o 4 nei modelli più recenti
- **numero di bit:** 4, 8, 16 e 32
- SET DI ISTRUZIONI:
  - **RISC** (*Reduced Instruction Set Computer*)
  - **CISC** (*Complex Instruction Set Computer*)

### RISC

- Clock ELEVATI
- Set di istruzioni per funzioni complesse
- Durata istruzione = 1 clock

### CISC

- Clock BASSI
- Istruzioni dedicate per funzioni complesse
- Durata istruzione > 1 clock

Se si vuole realizzare un **sistema real time** è necessario avere un **sistema deterministico**. Pertanto è fondamentale evitare di usare MCU dotate di:  
PIPELINE, BRANCH PREDICTION (per istruzioni di salto),  
ESECUZIONE SPECULATIVA (di istruzioni condizionate), CACHE



## PROCESSORE – ESEMPIO DI SET DI ISTRUZIONI



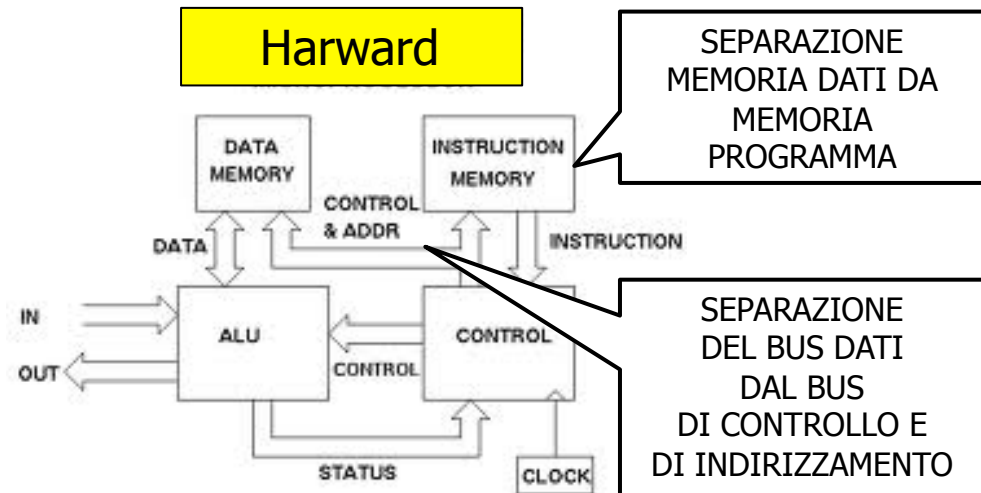
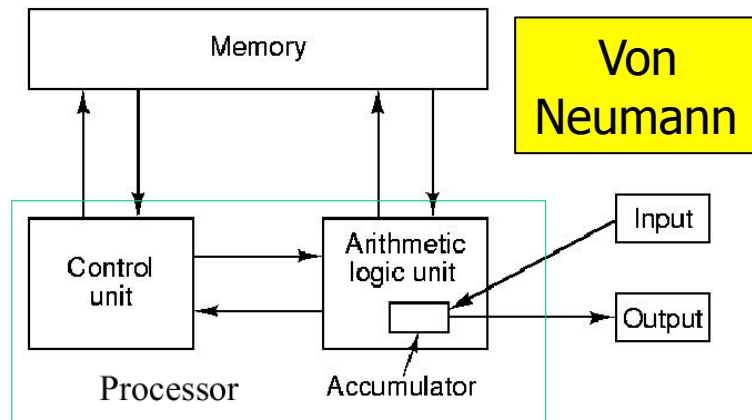
MOTOROLA  
68HC11x Family

Mnemonic	Operation	Description	Addressing Mode	Instruction			Condition Codes							
				Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
ABA	Add Accumulators	$A + B \Rightarrow A$	INH	1B	—	2	—	—	Δ	—	Δ	Δ	Δ	Δ
ABX	Add B to X	$IX + (00 : B) \Rightarrow IX$	INH	3A	—	3	—	—	—	—	—	—	—	—
ABY	Add B to Y	$IY + (00 : B) \Rightarrow IY$	INH	18 3A	—	4	—	—	—	—	—	—	—	—
ADCA (opr)	Add with Carry to A	$A + M + C \Rightarrow A$	A IMM	89	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			A DIR	99	dd	3								
			A EXT	B9	hh 11	4								
			A IND,X	A9	ff	4								
			A IND,Y	A9	ff	5								
ADCB (opr)	Add with Carry to B	$B + M + C \Rightarrow B$	B IMM	C9	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			B DIR	D9	dd	3								
			B EXT	F9	hh 11	4								
			B IND,X	E9	ff	4								
			B IND,Y	E9	ff	5								
ADDA (opr)	Add Memory to A	$A + M \Rightarrow A$	A IMM	8B	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			A DIR	9B	dd	3								
			A EXT	BB	hh 11	4								
			A IND,X	AB	ff	4								
			A IND,Y	AB	ff	5								
ADDB (opr)	Add Memory to B	$B + M \Rightarrow B$	B IMM	CB	ii	2	—	—	Δ	—	Δ	Δ	Δ	Δ
			B DIR	DB	dd	3								
			B EXT	FB	hh 11	4								
			B IND,X	EB	ff	4								
			B IND,Y	EB	ff	5								
ADDD (opr)	Add 16-Bit to D	$D + (M : M + 1) \Rightarrow D$	IMM	C3	jj kk	4	—	—	—	—	Δ	Δ	Δ	Δ
			DIR	D3	dd	5								
			EXT	F3	hh 11	6								
			IND,X	E3	ff	6								
			IND,Y	E3	ff	7								
ANDA (opr)	AND A with Memory	$A * M \Rightarrow A$	A IMM	84	ii	2	—	—	—	—	Δ	Δ	0	—
			A DIR	94	dd	3								
			A EXT	B4	hh 11	4								
			A IND,X	A4	ff	4								
			A IND,Y	A4	ff	5								



## PROCESSORE

- ARCHITETTURE:



### ESEMPI

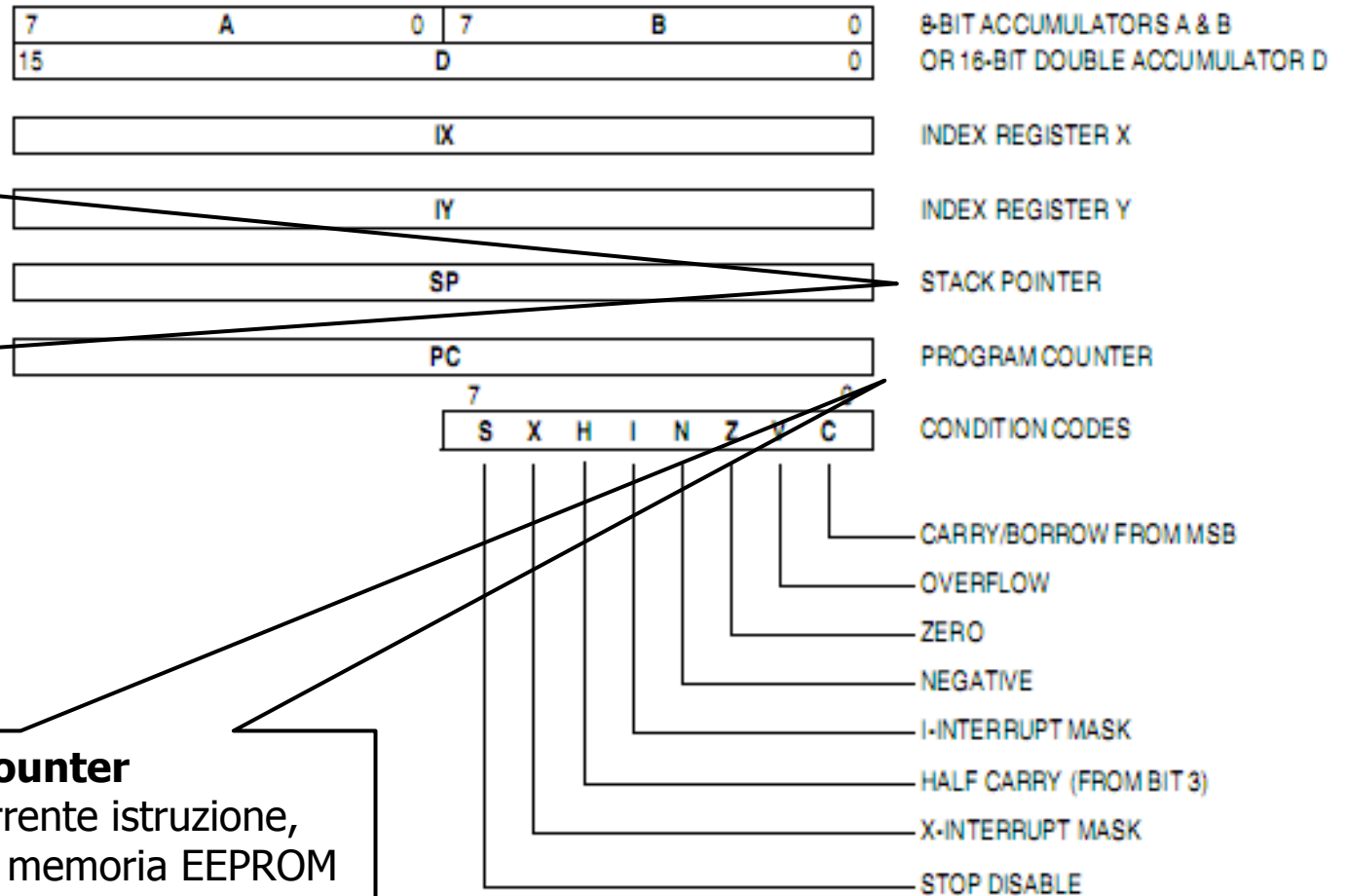


- ARM7: Van Neumann con spazio di indirizzi unico
- ARM9: Harward con spazio di indirizzi unico
- ATMEL 8051 e derivati: Harward con spazio di indirizzi separato
- ST10: Van Neumann
- ST40: Harward con spazio di indirizzi unico



# PROCESSORE

## REGISTRI



**Stack Pointer**  
È un **puntatore** ad una locazione di memoria RAM che tiene traccia dell'**occupazione** di memoria RAM per le istruzioni;  
**programmi ricorsivi** potrebbero far terminare tale memoria

**Program Counter**  
Tiene traccia della corrente istruzione, puntando all'indirizzo di memoria EEPROM





SAPIENZA  
UNIVERSITÀ DI ROMA

Corso di Laurea: INGEGNERIA  
Insegnamento: AUTOMAZIONE  
Docente: DR. VINCENZO SURACI

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

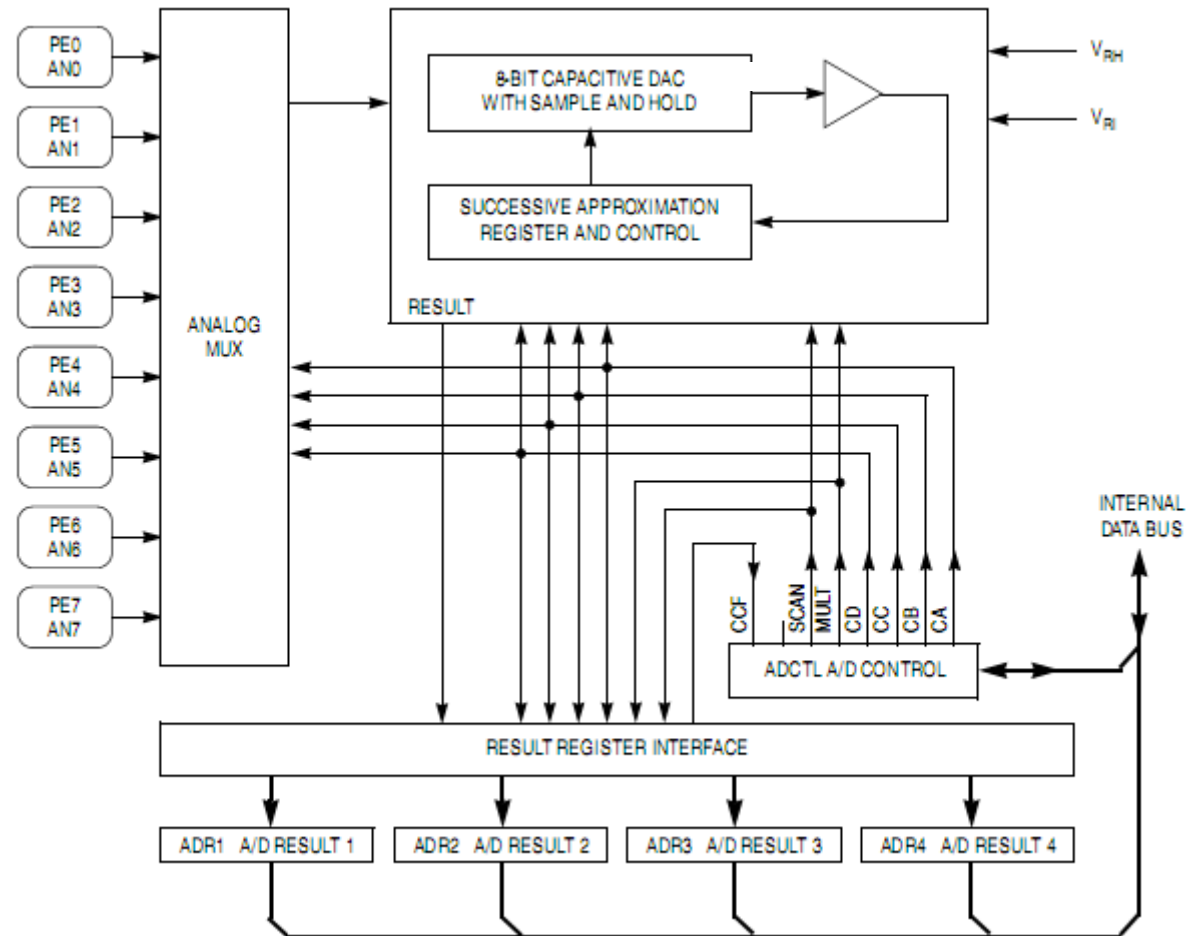
# PERIFERICHE INTEGRATE



## CONVERTITORE ANALOGICO DIGITALE



MOTOROLA  
68HC11x Family





## RESET

Per **supervisionare il flusso** di esecuzione delle istruzioni, ogni microcontrollore è dotato di opportuni strumenti per **azzerare** in maniera **condizionale** il **program counter**.

- **Power-on reset (POR)** Esiste in tutti i microcontrollori, è chiamato ad ogni **accensione** del sistema.
- **External reset (RESET)** Esiste in tutti i microcontrollori, può essere attivato **manualmente** con un pulsante esterno al sistema (il classico **tasto di RESET**).
- **Computer operating properly (COP) reset** è anche detto di **WATCH-DOG**. Il sistema interroga periodicamente questo PIN, se esso non è stato settato opportunamente, vuol dire che la CPU si è bloccata e viene chiamato un reset.
- **Clock monitor reset** è anche detto di **WATCH-DOG**. Il sistema verifica che il clock funzioni correttamente, altrimenti viene mandato un reset.



## TIMER & CONTATORI

Un **microcontrollore** realizza spesso **sistemi di controllo (hard/soft) real time** pertanto **integra strutture dedicate** alla **sincronizzazione di task**.

- **MAIN TIMER** – è un registro (normalmente a 16 bit) connesso attraverso un PRESCALER (che divide per  $2^n$ ) al clock di sistema. Il MAIN TIMER non VIENE MAI INTERROTTO e ricomincia da capo quando va in OVERFLOW.
- **CONTATORE**– è un registro (normalmente a 8 bit) che incrementa di una unità ad ogni evento rilevato.
- **CONTATORE DI IMPULSI** – è un CONTATORE in cui l'evento è generato da una rilevazione di un fronte di salita/discesa.
- **TIMER** - è un CONTATORE DI IMPULSI connesso al clock di sistema. Quando va in overflow genera un evento.
- **REAL TIME INTERRUPT (RTI)** – è un TIMER che genera un INTERRUPT periodico programmabile, quando il timer va in OVERFLOW. Una interruzione al flusso di esecuzione delle istruzioni (dipendente dalla PRIORITÀ associata al RTI) fa «saltare» il PROGRAM COUNTER alla istruzione della sub-routine real-time che deve essere eseguita.



## INTERRUPT

Per **controllare il flusso** di esecuzione delle istruzioni, ogni microcontrollore è dotato di opportuni strumenti per **cambiare in maniera condizionale il program counter**, quando specifici **eventi** occorrono.

Gli **interrupt** possono essere generati da una serie di **possibili eventi**, ad ognuno dei quali può essere associata una **priorità**.

Ad esempio:

- Timer/Counter Overflow
- PWM Counter Overflow
- Dati pronti sul canale di comunicazione
- Reset
- Software Interrupt
- Eventi esterni



MOTOROLA  
68HC11x Family

Vector Address	Interrupt Source	CCR Mask Bit	Local Mask
FFC0, C1 – FFD4, D5	Reserved	—	—
FFD6, D7	SCI serial system • SCI receive data register full • SCI receiver overrun • SCI transmit data register empty • SCI transmit complete • SCI idle line detect	I	RIE RIE TIE TCIE ILIE
FFD8, D9	SPI serial transfer complete	I	SPIE
FFDA, DB	Pulse accumulator input edge	I	PAIE
FFDC, DD	Pulse accumulator overflow	I	PAOVI
FFDE, DF	Timer overflow	I	TOI
FFE0, E1	Timer input capture 4/output compare 5	I	I4/O5I
FFE2, E3	Timer output compare 4	I	OC4I
FFE4, E5	Timer output compare 3	I	OC3I
FFE6, E7	Timer output compare 2	I	OC2I
FFE8, E9	Timer output compare 1	I	OC1I
FFEA, EB	Timer input capture 3	I	IC3I
FFEC, ED	Timer input capture 2	I	IC2I
FFEE, EF	Timer input capture 1	I	IC1I
FFF0, F1	Real-time interrupt	I	RTII
FFF2, F3	IRQ (external pin)	I	None
FFF4, F5	XIRQ pin	X	None
FFF6, F7	Software interrupt	None	None
FFF8, F9	Illegal opcode trap	None	None
FFFA, FB	COP failure	None	NOCOP
FFFC, FD	Clock monitor fail	None	CME
FFFE, FF	RESET	None	None



## PORTE DI I/O

Le porte di **I/O** permettono al **microcontrollore** di **interagire** con **sensori, attuatori** e altre **periferiche esterne**, in maniera additiva e programmabile.

Spesso le **porte di I/O** sono **POLI FUNZIONALI** e possono essere usate in maniera **mutuamente esclusiva** per svolgere **specifiche funzionalità** al fine di **personalizzare** il microcontrollore.

Port	Input Pins	Output Pins	Bidirectional Pins	Shared Functions
Port A	3	3	2	Timer
Port B	—	8	—	High-order address
Port C	—	—	8	Low-order address and data bus
Port D	—	—	6	Serial communications interface (SCI) and serial peripheral interface (SPI)
Port E	8	—	—	Analog-to-digital (A/D) converter



MOTOROLA  
68HC11x Family

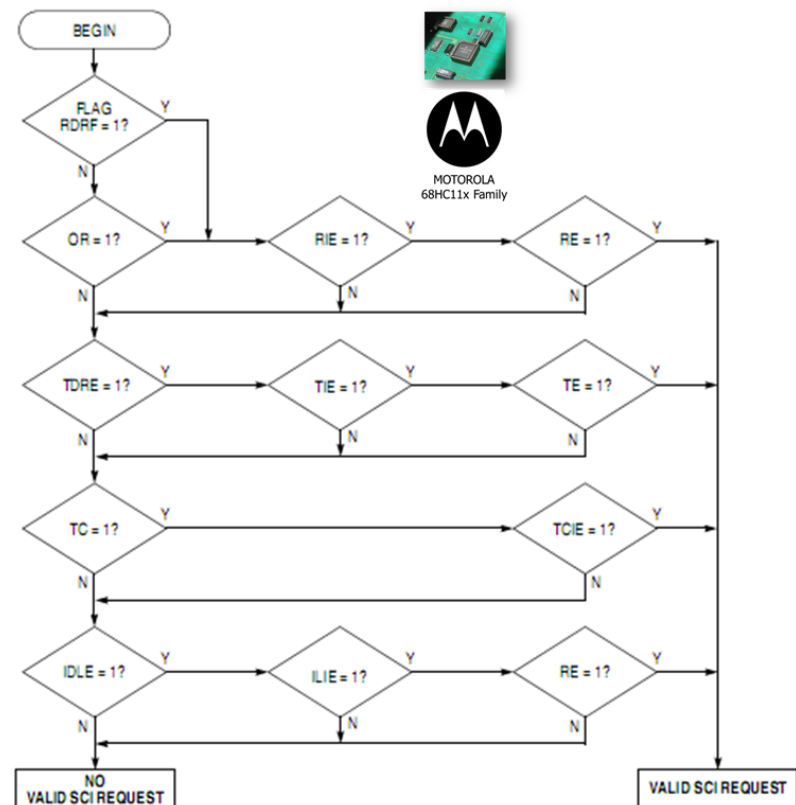


## COMUNICAZIONE SERIALE (SCI)

Per comunicare con **altri microcontrollori** o **dispositivi esterni**, ogni microcontrollore è dotato di una **interfaccia di comunicazione seriale**.

Le principali caratteristiche di una **comunicazione seriale** sono:

- **FORMATO DATI** – bit di START, bit di STOP, Least Significant Bit (LSB), ecc.
- **BAUD RATE** – configurando opportunamente il **baud rate register** e tenendo conto del clock di sistema, si possono ottenere rate variabili fino a 115200 simboli (baud) al secondo
- **CONTROL & STATUS REGISTER** – per configurare e controllare la porta seriale





SAPIENZA  
UNIVERSITÀ DI ROMA

Corso di Laurea: INGEGNERIA  
Insegnamento: AUTOMAZIONE  
Docente: DR. VINCENZO SURACI

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

# PROGRAMMAZIONE





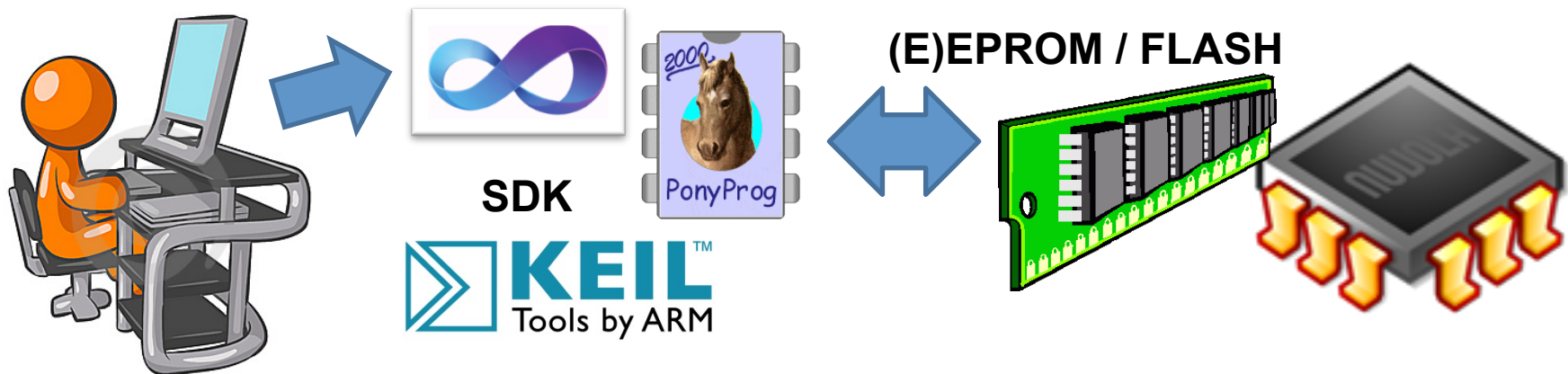
## SISTEMA DI SVILUPPO DEL SOFTWARE

### OSSERVAZIONE

Ogni **microcontrollore** esegue un **set di istruzioni** (codice macchina) definito dall'utente. È pertanto necessario utilizzare opportuni **sistemi di sviluppo** per caricare il software nei microcontrollori.

### DEFINIZIONE

Per **SISTEMA DI SVILUPPO** s'intende l'insieme di strumenti (**kit**) **software e hardware** necessari alla **generazione del codice macchina** che deve essere eseguito dal processore (implementazione del software), al suo collaudo e messa a punto (debug).





## IMPLEMENTAZIONE DEL SOFTWARE

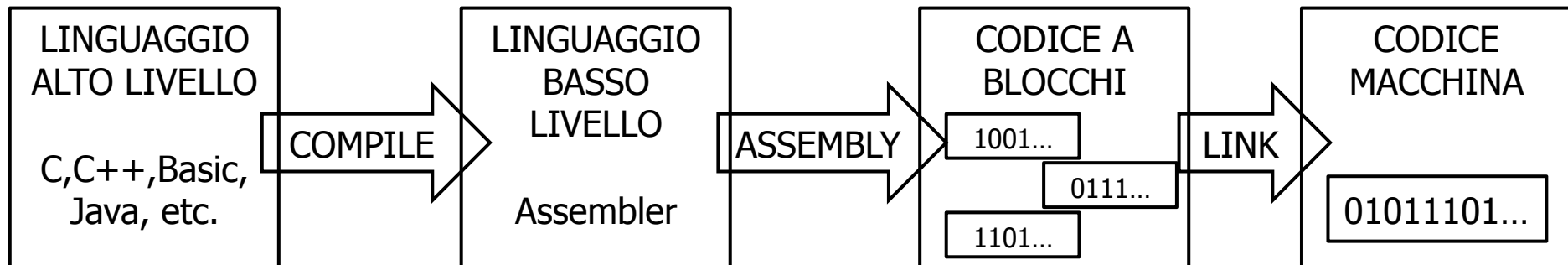
- L'**implementazione** consiste nella **stesura del programma in linguaggio assembly o di alto livello** (tipicamente il C), utilizzando un editor di testo generico o specifico per quel linguaggio.
- Una volta scritto il programma Assembly deve essere **ASSEMBLATO**, cioè **tradotto** nell'effettivo **codice macchina numerico** (generalmente esadecimale). La conversione viene fatta da un Assemblatore specifico per processore, o famiglia di processori.
- Se **codificato in alto livello**, il programma deve essere **compilato**, per mezzo di un compilatore che lo **converte prima in linguaggio Assembly**, e quindi nell'effettivo codice macchina, in due passaggi successivi. Anche il compilatore deve essere specifico per processore, o famiglia di processori.





## PROGRAMMAZIONE DEL SOFTWARE

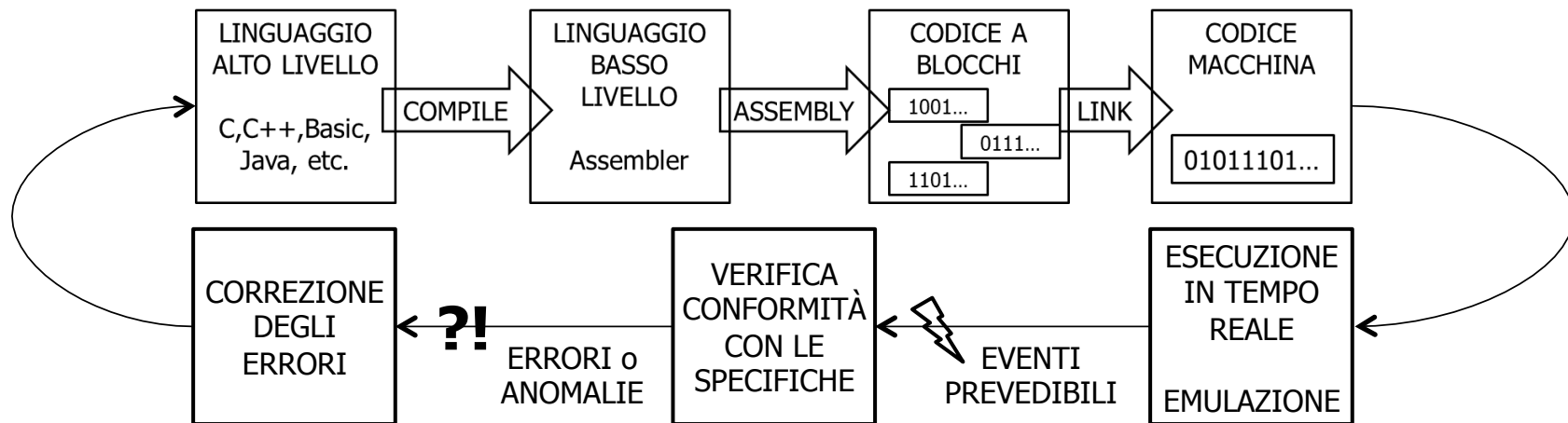
- Nella maggior parte dei casi, la **complessità delle funzioni** di un programma per microprocessore, richiede la **suddivisione in moduli funzionali** (o sottoprogrammi).
- Ciascun modulo viene quindi **assemblato in modalità rilocabile** (ad indirizzi non determinati) generando diversi blocchi di codice.
- L'associazione di tutti i **moduli assemblati agli indirizzi definitivi**, viene effettuata, in un'ulteriore passaggio, da un **LINKER**, che genera il codice macchina definitivo (codice eseguibile), in un formato opportuno per poter essere trasferito nella memoria del processore, ed eseguito.





## DEBUG DEL SOFTWARE

- La fase di **debug** consiste nel **far eseguire il software dal processore**, in condizioni quanto più simili a quelle **reali di funzionamento (emulazione)**, **verificando in tempo reale** che il suo **comportamento** ad ogni **evento prevedibile**, sia **conforme alle specifiche di progetto**.
- Nel caso di **errori o anomalie**, il software viene **corretto**, un nuovo codice eseguibile generato e trasferito in memoria, per essere nuovamente verificato.
- Questo **processo continua iterativamente** fino a che il programma non sia stato **completamente collaudato**.





SAPIENZA  
UNIVERSITÀ DI ROMA

Corso di Laurea: INGEGNERIA  
Insegnamento: AUTOMAZIONE  
Docente: DR. VINCENZO SURACI

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

# PROGRAMMABLE INTERFACE COMPUTER (PIC)



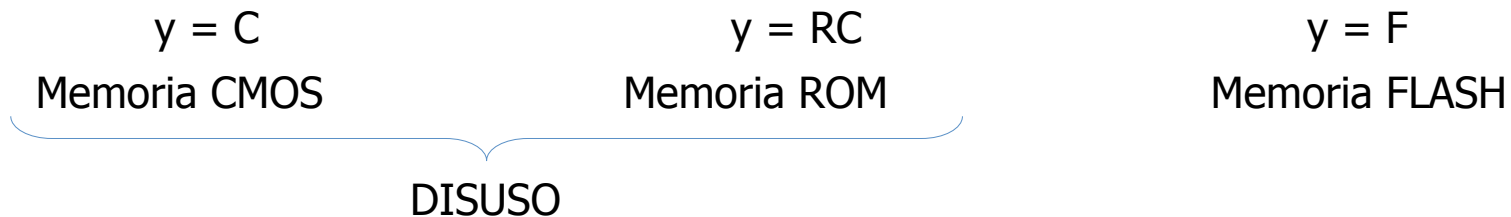
## STORIA

**1975** – La **General Instrument** progetta un nuovo modello di MICROCONTROLLORE che chiama PIC (PROGRAMMABLE INTELLIGENT COMPUTER).

**1987** – La General Instrument fonda la **Microchip Technology**, uno Spin-Off aziendale cui delega la produzione dei PIC, con il trademark «**PICmicro**».

**1989** – La Microchip Technology ha successo, vende milioni di PIC e si separa definitivamente dalla General Instrument.

**Oggi** – Il termine «PIC» fa riferimento all'acronimo «**PROGRAMMABLE INTERFACE COMPUTER**» e comprende le seguenti famiglie di microcontrollori: PIC10yxx - PIC12yxx - PIC16yxx - PIC18yxx - PIC24yxx - PIC32yxx

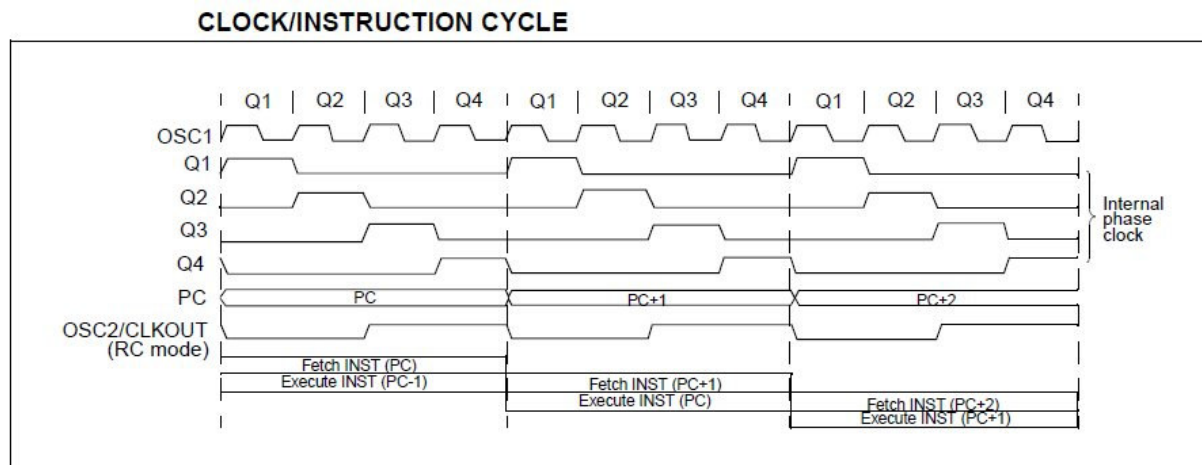




## CARATTERISTICHE DI BASE

Un PIC è un MICROCONTROLLORE caratterizzato da:

- Un set di istruzioni di tipo **RISC** (*Reduced Instruction Set Computer*) - 33 nelle versioni più semplici e fino a 77 in quelli con prestazioni più elevate
- Una struttura di esecuzione a **pipeline** di tipo **deterministico**: ogni istruzione dura **4 CICLI DI CLOCK**, tranne quelle di SALTO che impiegano **8 CICLI DI CLOCK**



- L'architettura è di tipo **Harvard a bus separati**; i bus dati e controllo sono a 8/16 bit le istruzioni hanno un formato a 12, 14 o 16 bit



## MEMORIA

**RAM** - ha una **larghezza di parola di 8 bit** e una profondità che varia da pochi byte (25 nei PIC16xxx) fino a qualche kilobyte.

**EEPROM** - Nelle **versioni con memoria flash** può essere presente una **memoria interna di tipo eeprom** accessibile come fosse una periferica per potervi MEMORIZZARE in maniera indelebile **parametri di configurazione** del software da elaborare. La sua profondità varia da 64 a 1024 byte.

- **Memoria Istruzioni** - La larghezza della parola di programma varia da 12 bit (ad esempio, nel PIC16C54) a 14 bit (ad esempio, nel PIC16F628) a 16 bit (ad esempio, nel PIC18F4520). La sua profondità arriva fino 128 kbyte.
- **Memoria Dati (stack)** - Lo stack è un tipo di memoria, separata da quella principale. È dotato di un suo bus che va da 2 fino a 31 linee (PIC della serie 18).





## PERIFERICHE INTEGRATE

### PORTE I/O

- La funzione di **ingresso e uscita di dati digitali** è stata la prima funzione implementata nei PIC.
- Ogni **porta** è costituita da **8 (o meno) bit** ognuna.
- È possibile **programmare ogni bit** come ingresso o come uscita singolarmente.
- In alcune versioni è possibile avere degli **ingressi con conversione analogico digitale (ADC) da 10 o 12 bit**.
- In alcuni casi è possibile attivare dei resistori interni (weak pull-up) per **facilitare il collegamento con pulsanti ed interruttori**.

### RETE

- Può essere presente una grande varietà di porte seriali: I<sup>2</sup>C, USART, SPI, CAN, USB.

### PWM

- Si arriva fino a 5 canali PWM a 10 bit.



## PERIFERICHE INTEGRATE

### TIMER

- Su tutte le versioni è implementato **almeno un temporizzatore a 8 bit**.
- Si arriva **fino a 5 temporizzatori** con larghezze a 8 o 16 bit.
- Il **timer** funziona in base alla frequenza di lavoro del PIC e/o può esserne data, tramite un apposito piedino, una diversa da quella di lavoro tramite un oscillatore esterno.
- Il **prescaler** divide la frequenza di lavoro di: 2, 4, 8, 16, 32, 64, 128, 256 volte.

### WATCH DOG TIMER

- Su tutti i PIC è inoltre implementato un temporizzatore speciale chiamato WDT (**Watch Dog Timer**) che serve (se utilizzato) a far ripartire il microcontrollore in caso di **blocco del programma**.





## PROGRAMMAZIONE

### LINGUAGGIO DI PROGRAMMAZIONE

- Il linguaggio di programmazione dei PICmicro è l'**assembly**, ma sono stati implementati alcuni compilatori per semplificarne la programmazione.
- Sono disponibili infatti molti compilatori di linguaggi con sintassi simili al **BASIC** oppure compilatori di **C** o **Pascal**. Esistono anche **linguaggi gratuiti** come **Jal** (Just Another Language) e **SDCC** (Small Device C Compiler).

### COMPILATORI PIÙ USATI

-  **mikroElektronika**  
TOOLS | COMPILERS | BOOKS | MAGAZINE
-  **CCS** Inc  
Custom Computer Services, Inc. **PICC** di CCS (riconosciuto come compilatore di terze parti da Microchip)
- Microchip mette a disposizione i compilatori C per le famiglie più avanzate: C18, C24, C30 (16bit) e PIC32 (32bit), e una **libreria di codice open source** per una grande **varietà di applicazioni**.



SAPIENZA  
UNIVERSITÀ DI ROMA

Corso di Laurea: INGEGNERIA  
Insegnamento: AUTOMAZIONE  
Docente: DR. VINCENZO SURACI

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

# ARDUINO





## Un po' di **STORIA**

**2001** - Olivetti e Telecom Italia creano l'Interaction Design Institute di Ivrea.

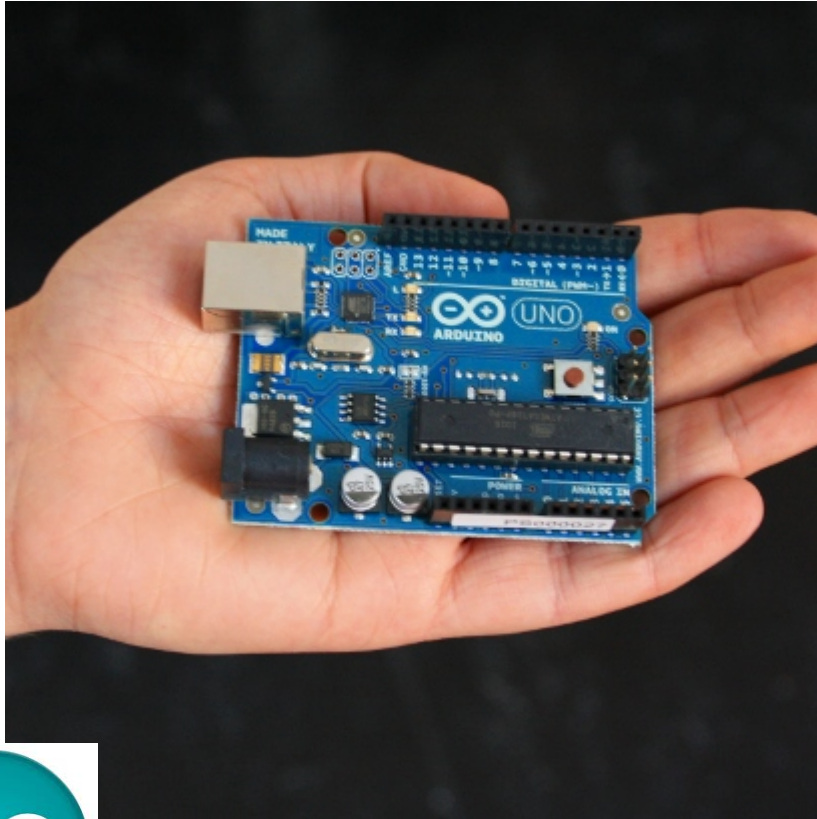
**2005** – Massimo Banzi crea Arduino, come strumento di prototipazione elettronica.

**Oggi** – Arduino ha un successo planetario e diventa uno strumento potentissimo per costruire **facilmente** e **rapidamente** prototipi funzionanti di **controllori embedded** a **basso costo**.





## COSA È ARDUINO SECONDO IL SUO CREATORE?



Arduino is an **open-source** electronics prototyping platform based on flexible, easy-to-use hardware and software.

It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

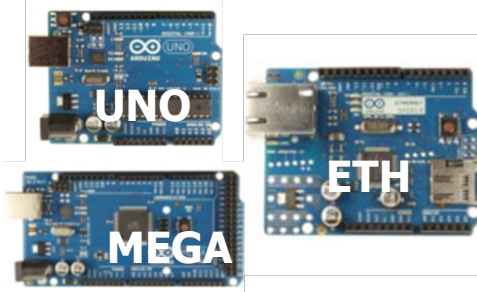




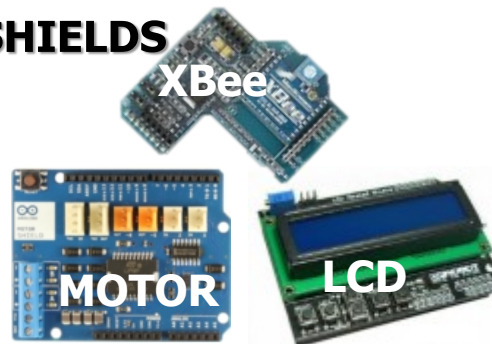
## COSA È ARDUINO IN ESTREMA SINTESI?

- Arduino è una **iniziativa** finalizzata alla definizione di **requisiti hw/sw** per la costruzione **open-source** di **controllori embedded**.
- Il successo dell'iniziativa si basa su:
  - Possibilità di scaricare **gratuitamente** gli schemi hardware da [www.arduino.cc](http://www.arduino.cc) di tutti i modelli delle **BOARD** Arduino
  - Possibilità di costruire **gratuitamente** (no royalties) delle **SHIELD** Arduino da collegare meccanicamente con le BOARD al fine di estendere le capacità e le funzionalità della stessa
  - Possibilità di scaricare **gratuitamente** l'**SDK** per iniziare da subito a **programmare** le **BOARD** Arduino e le sue **SHIELD** ufficiali

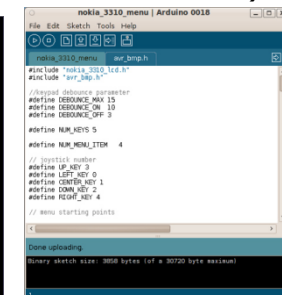
### BOARDS



### SHIELDS



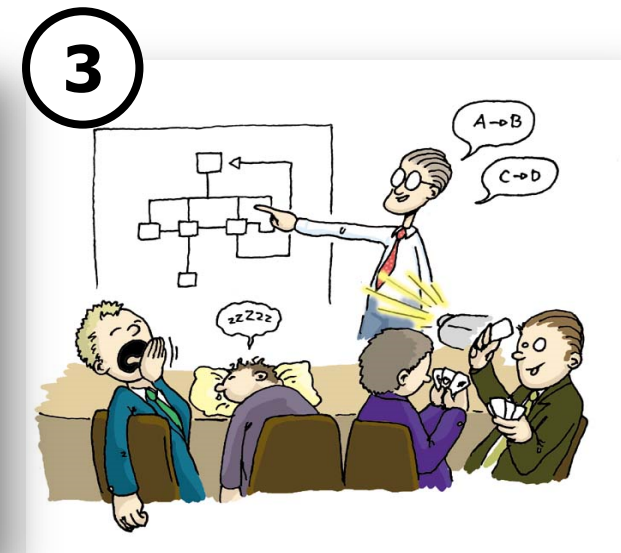
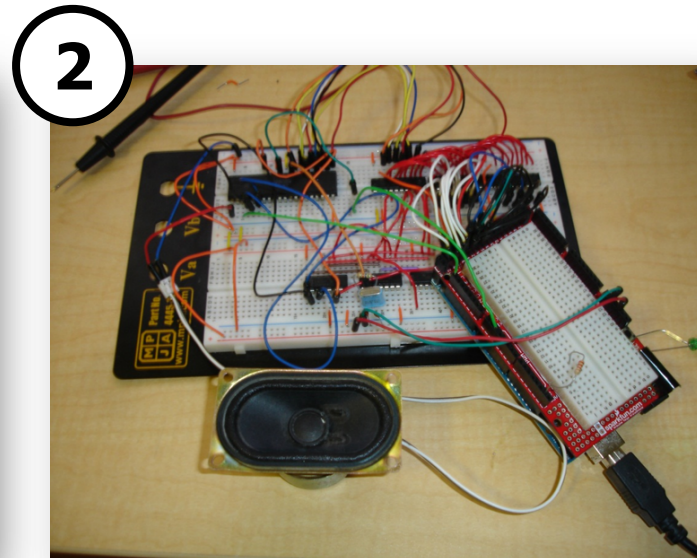
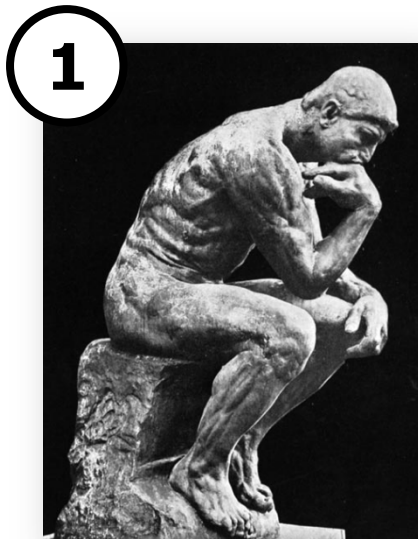
### IDE (Integrated Development Environments)





## COME USARE ARDUINO IN AUTOMAZIONE ?

1. Trovate un **caso d'uso** applicabile al settore dell'**automazione** (non necessariamente industriale)
2. **Progettate l'idea** e **realizzatela** (spendendo il meno possibile)
3. **Presentate l'idea** come **prova scritta di questo nucleo didattico**







## BIBLIOGRAFIA

- P. Foglia «Microcontrollori» – slide disponibili on-line
- M68HC11E Family - Data Sheet – documentazione disponibile on-line
- PIC – documentazione rielaborata da Wikipedia
- Arduino – documentazione disponibile on-line ([www.arduino.cc](http://www.arduino.cc))