



UNIVERSITÀ DI PISA

CORSO DI LAUREA IN FISICA

LABORATORIO DI FISICA 3

MICROPROCESSORI

Prof. F. Forti

# Un po` di storia: dai primi calcolatori ai microprocessori

2

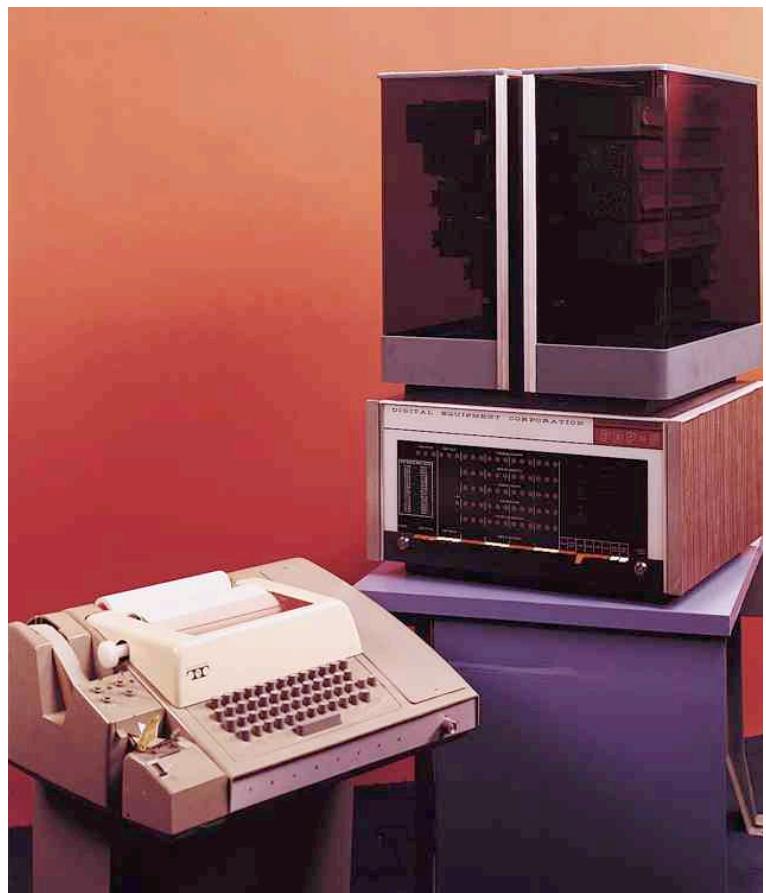
## □ I primi computer a valvole

- 1946 ENIAC (Electronic Numerical Integrator and Calculator): un calcolatore mastodontico costruito per calcoli balistici per l'esercito americano (17000 valvole, 70000 resistenze, 10000 condensatori ... 30 tonnellate) consumava circa 200 kW (200 W per un pentium). Il calore sprigionato era uno delle principali cause di rottura. Dopo 9 anni di funzionamento divenne praticamente impossibile utilizzarlo a causa delle continue rotture. E` durante la costruzione di ENIAC che venne coniato il termine “bit” dalla contrazione delle parole binary-digit.
- Abbiamo visto che l'invenzione che ha rivoluzionato il mondo dei calcolatori e` stato il transistor (1948). Molto piu` resistente e piccolo delle valvole e con consumi molto ridotti.
- Si cominciano a costruire i primi calcolatori a Transistor e si comincia a dover affrontare un altro problema : la memoria, ovvero come immagazzinare dati senza il rischio di perderli ? Come si potevano avere memorie di dimensioni ridotte ma capienti ?

# Un po` di storia...

3

- Prima si utilizzarono valvole, poi memorie magnetiche (nastri e dischi)
- Il primo computer a transistor di dimensioni “umane”, che divento` in effetti un fenomeno di massa, fu il PDP-8 (grande come un frigorifero con una memoria di 4 kbyte)



# Un po` di storia: dai primi calcolatori ai microprocessori

4

- La seconda vera rivoluzione fu l'invenzione dei circuiti integrati ... finalmente le dimensioni potevano diventare davvero piccole. Il primo computer a circuiti integrati fu il Sistema/360 dell'IBM (1963)
- Il P101 della Olivetti (1965) fu un altro passo in avanti: primo pc prodotto in serie e programmabile. Fu il primo personal computer.

Il P101 aveva 10 registri di memoria, un linguaggio di programmazione facilmente assimilabile basato su 15 istruzioni elementari di significato intuitivo, la possibilità di registrare dati e programmi su una scheda magnetica che funziona come un floppy disk, una piccola stampante incorporata. Il tutto in un oggetto che può stare su una scrivania.

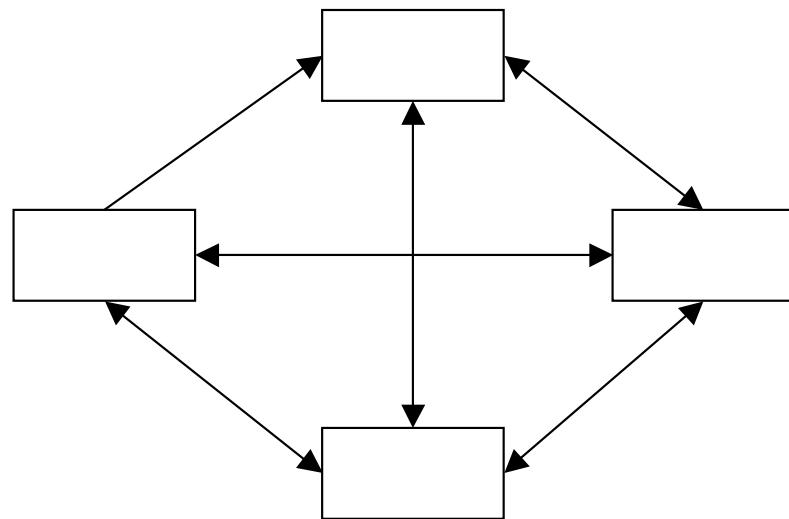




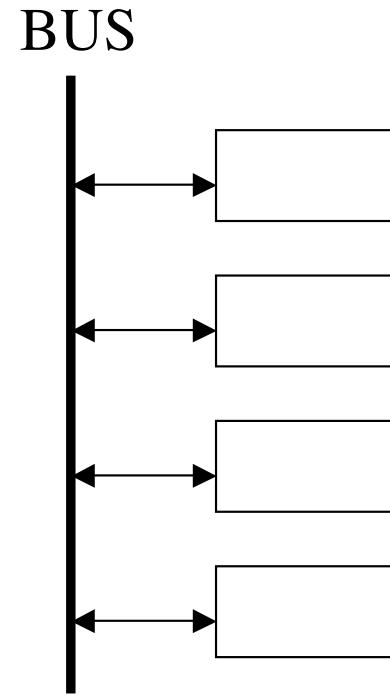
Fisica - Lab3 - Forti

# Comunicazione in sistemi complessi

6



“Tutti con tutti”:  
dispendioso e intricato



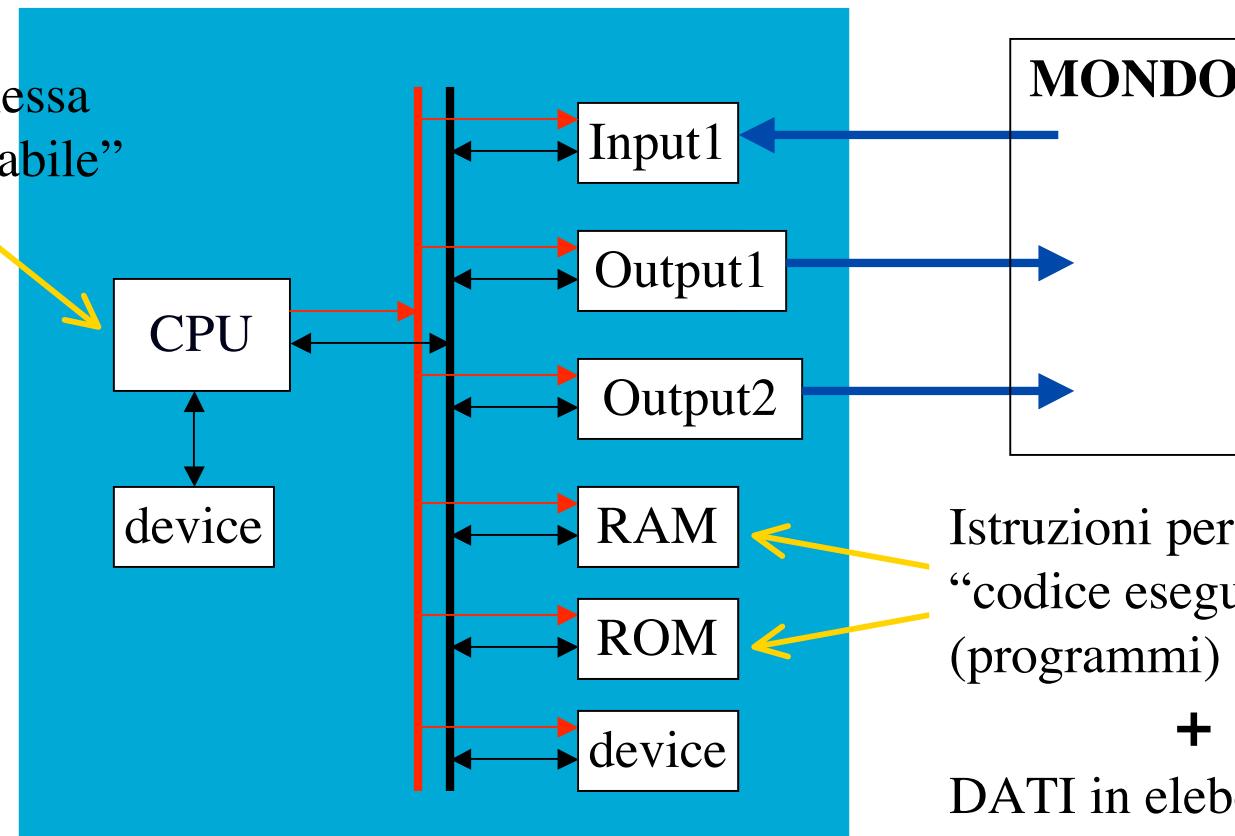
“Tutti con la stessa linea”:  
lineare ed economico

# Architettura di un calcolatore

7

## Microcomputer / Microcontroller

FSM complessa  
“Programmabile”

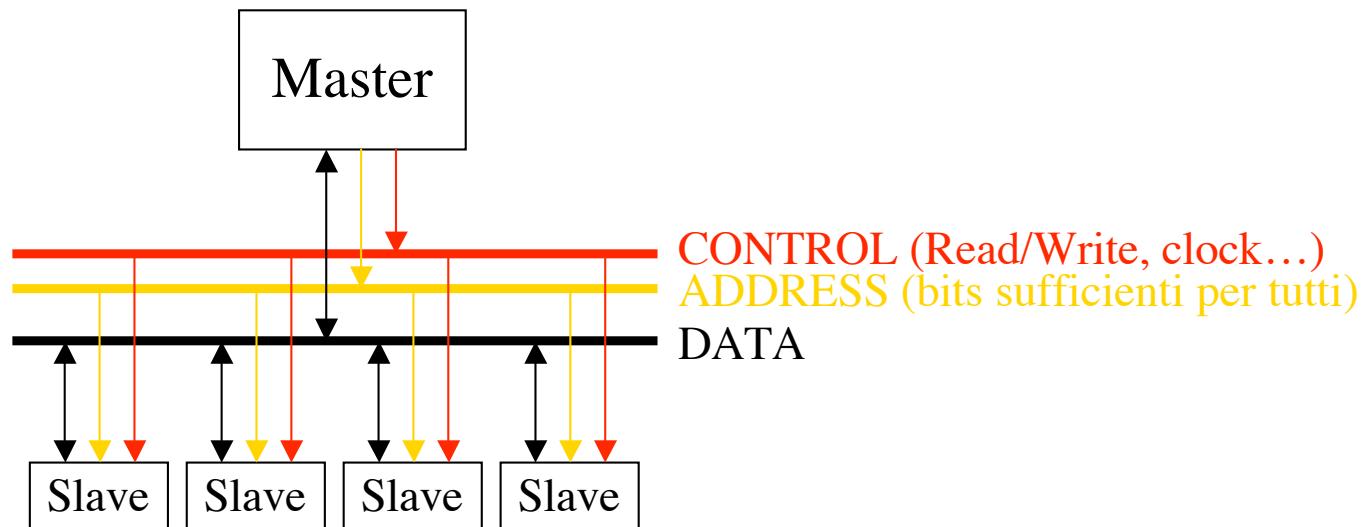


Istruzioni per la CPU:  
“codice eseguibile”  
(programmi)  
+  
DATI in elaborazione

# Bus protocol

8

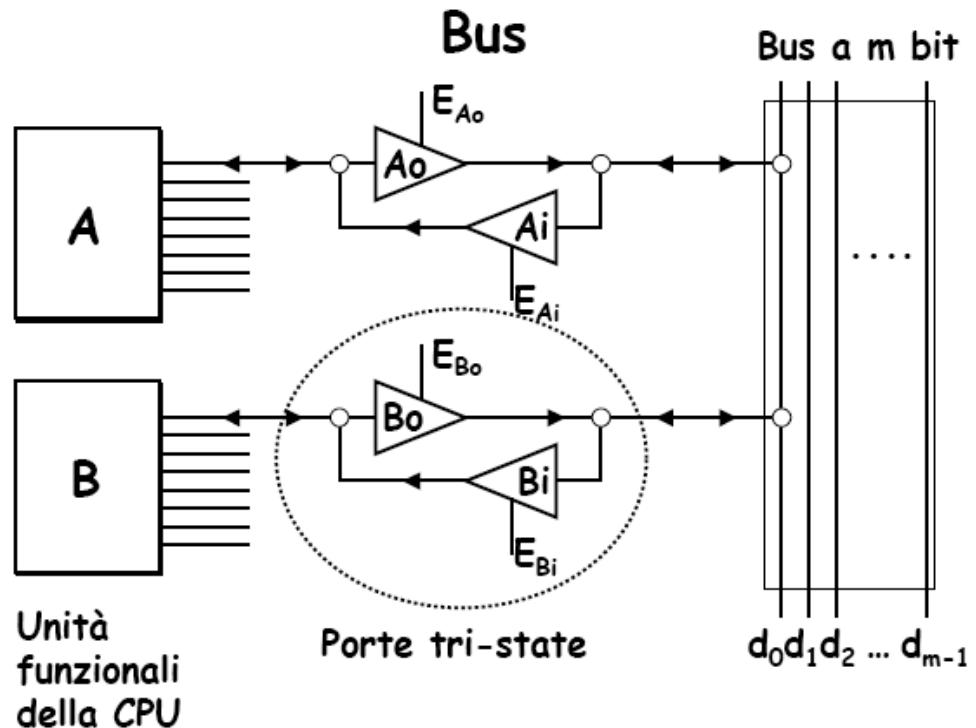
- Un BUS richiede un protocollo per evitare conflitti
  - Ogni device deve essere capace di andare in 3-state
  - Ogni device deve avere un “indirizzo” (o piu’) che lo identifica
  - Il BUS deve comprendere linee per indirizzamento e controllo
  - In genere occorre un clock per sincronizzare la comunicazione
  - Occorre un MASTER che “diriga il traffico” sul BUS
- Puo’ essere necessario cambiare MASTER in certe circostanze: occorre un protocollo per passare il controllo (BUS mastering)



# Bus e porte tri-state

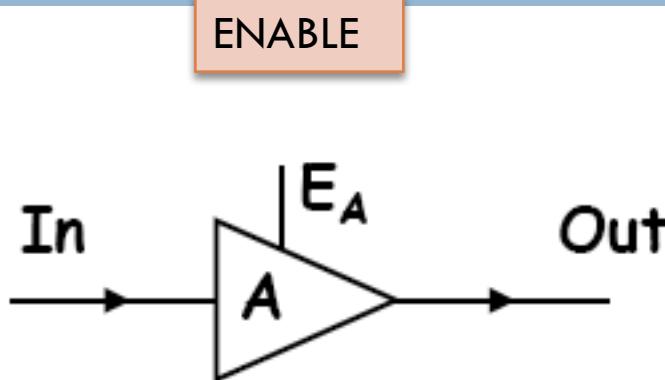
9

Ogni linea puo` assumere il valore 1 o 0 (ALTO o BASSO) si deve pero` tenere anche conto che quando un componente immette dati su un determinato BUS tutti gli altri componenti collegati allo stesso BUS devono apparire disconnessi dal BUS, il BUS deve cioe` vederli come alte impedenze (cioe` come se fossero scollegati). Per far questo si utilizzano delle porte logiche che oltre che negli stati 1,0 possono essere in un terzo stato di ALTA IMPEDENZA: queste porte si chiamano tri-state.

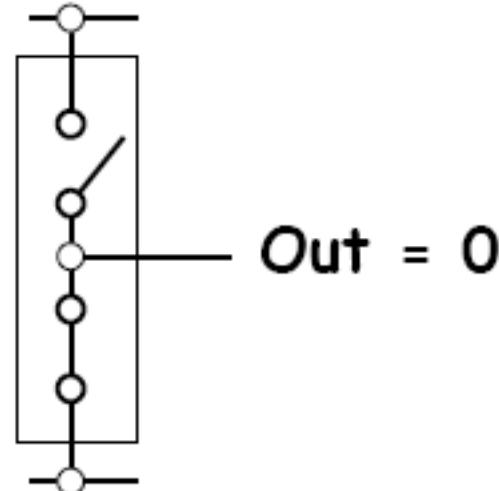


# Simbolo e tabella della verita` porta tri-state

10



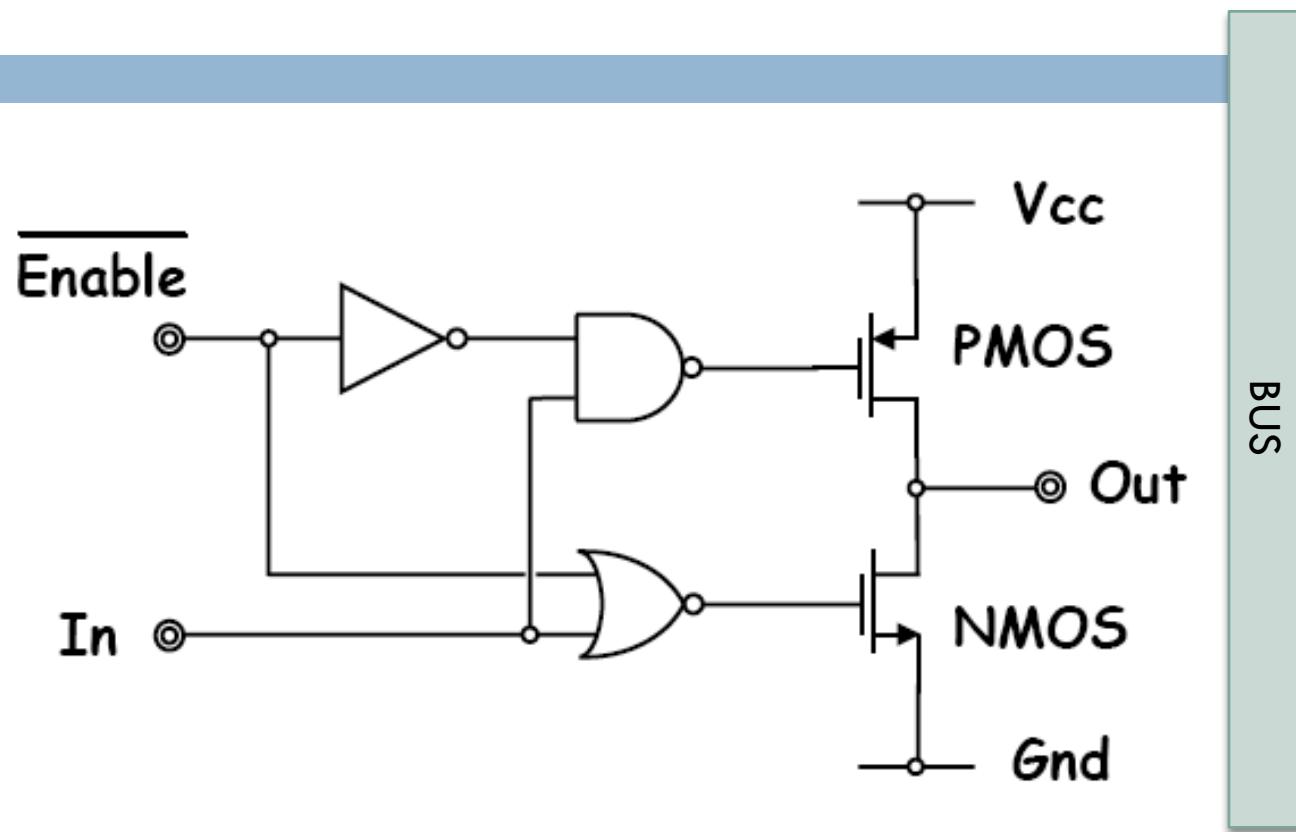
Enable	In	Out	
0	0	X	Floating
0	1	X	Floating
1	0	0	Out = In
1	1	1	Out = In



Porta tri-state a  
livello logico basso

# Porta tri-state CMOS

11

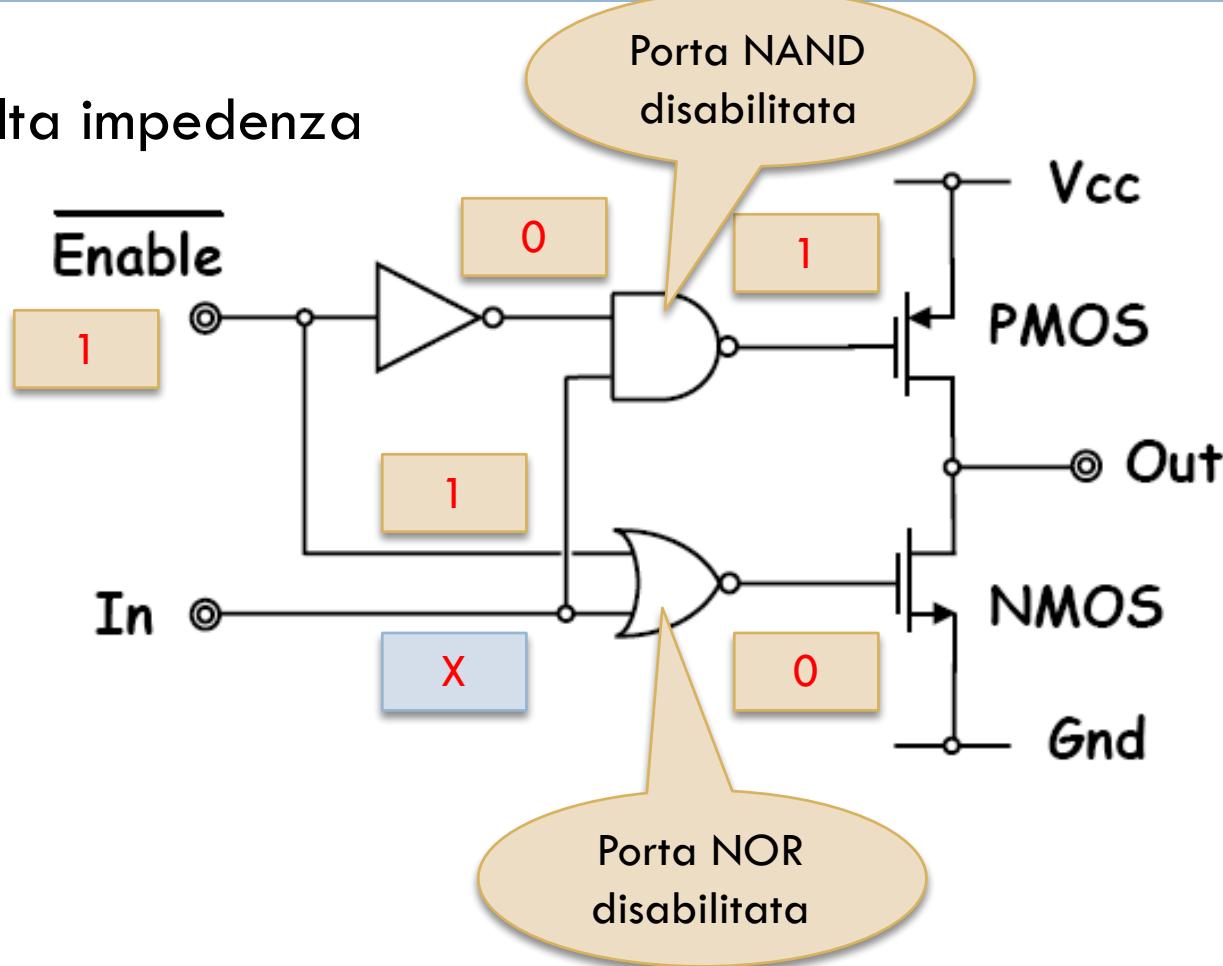


Il segnale di ENABLE permette di definire se la porta si comporta da interruttore chiuso (con  $\text{In}=\text{Out}$ ) oppure da interruttore aperto.  
Logica negativa.

# Porta tri-state CMOS

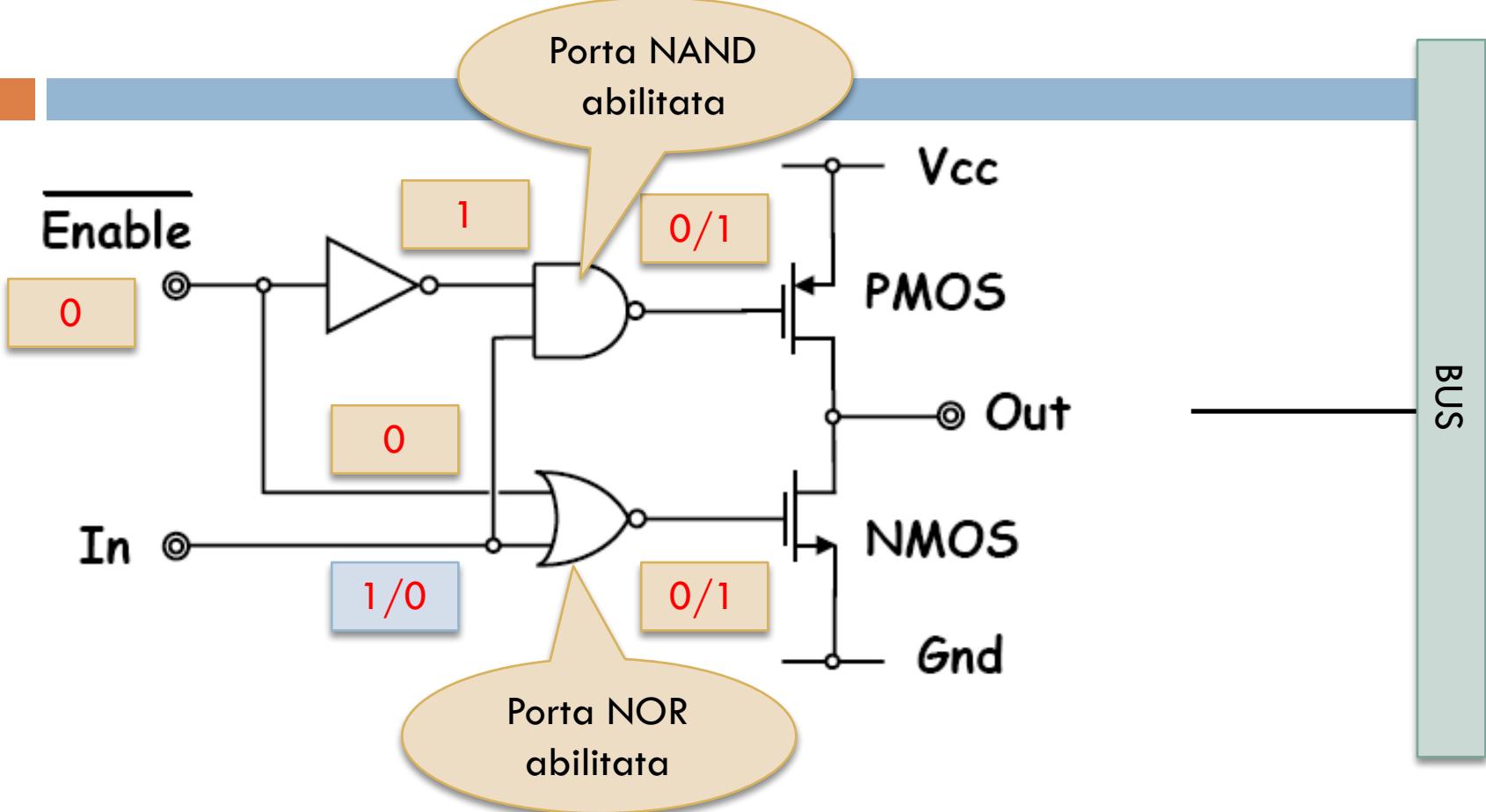
12

Stato alta impedenza



# Porta tri-state CMOS - abilitata

13



Uno dei due transistor e` in saturazione e quindi il BUS vede una bassa resistenza -> interruttore chiuso.

Se ENABLE-Bar e` BASSO il buffer CMOS composto dai due transistor e` abilitato e Out = In.

# Architetture dei calcolatori

14

Le architetture di un calcolatore si dividono in due grosse classi distinte per il modo con cui sono organizzati le “comunicazioni” tra memoria e CPU :

- Architettura di Von Neumann
  - ▣ Il bus dei dati ed il bus delle istruzioni condividono la memoria ed il bus. Architettura semplice e quindi economica ma lenta
- Architettura di Harvard
  - ▣ Il bus dei dati e delle istruzioni è separato e così le memorie. In questo caso si può avere un accesso contemporaneo a memoria dati ed istruzioni riducendo così il tempo di esecuzione delle istruzioni.

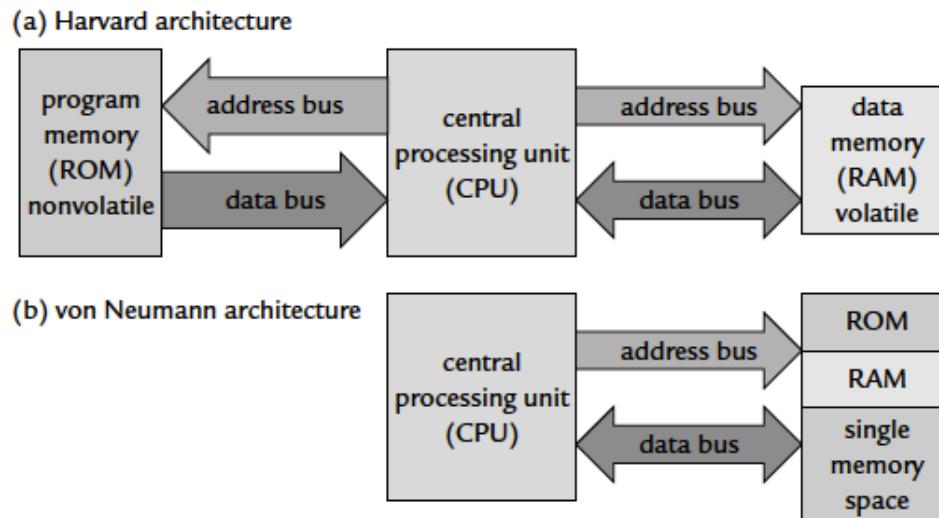
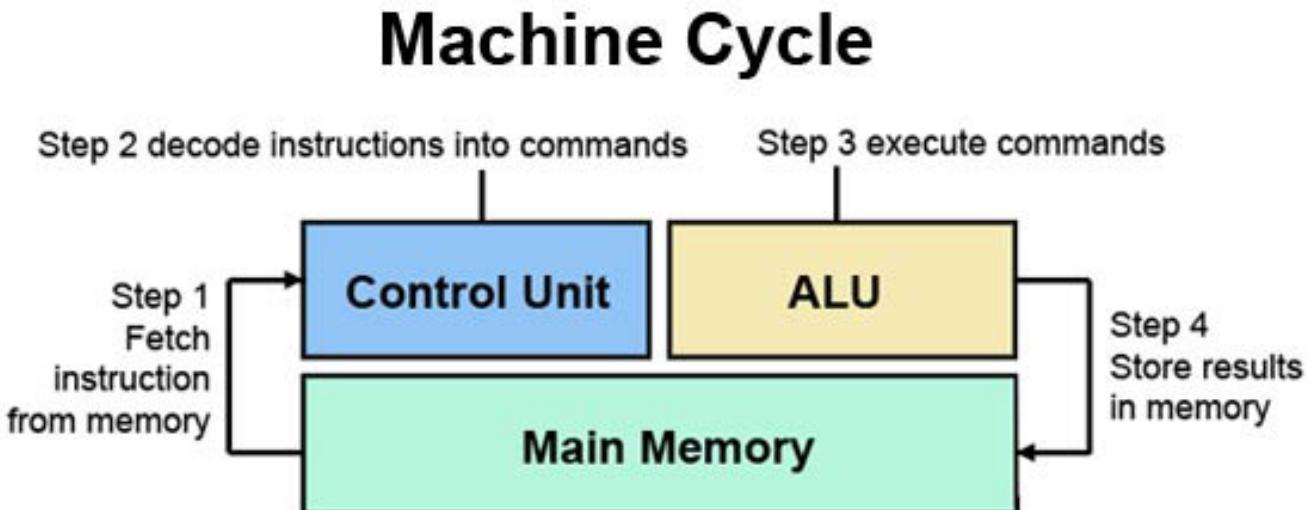


Figure 1.3: Harvard and von Neumann architectures for memory.

# Ciclo di CPU

15

CLOCK

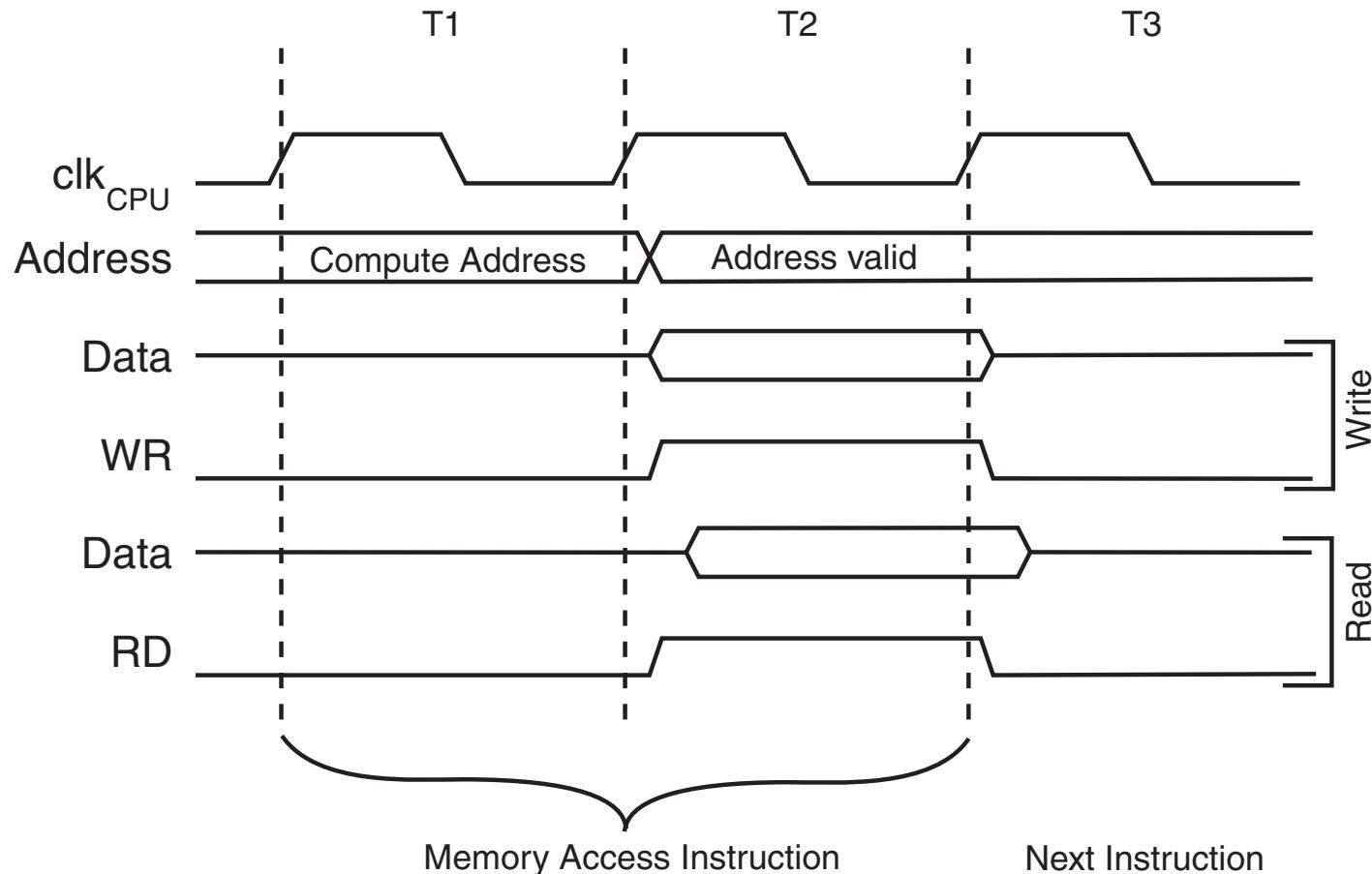


<http://www.computerhope.com>

- Memoria esterna o interna (diversa velocità)
- Esecuzione condizionata da registri di stato
- Input/output memory-mapped: usa certe specifiche locazioni di memoria
- Normalmente a fine ciclo il PC (program counter) viene incrementato
  - A meno che non ci sia un jump

# Memory read/write

16

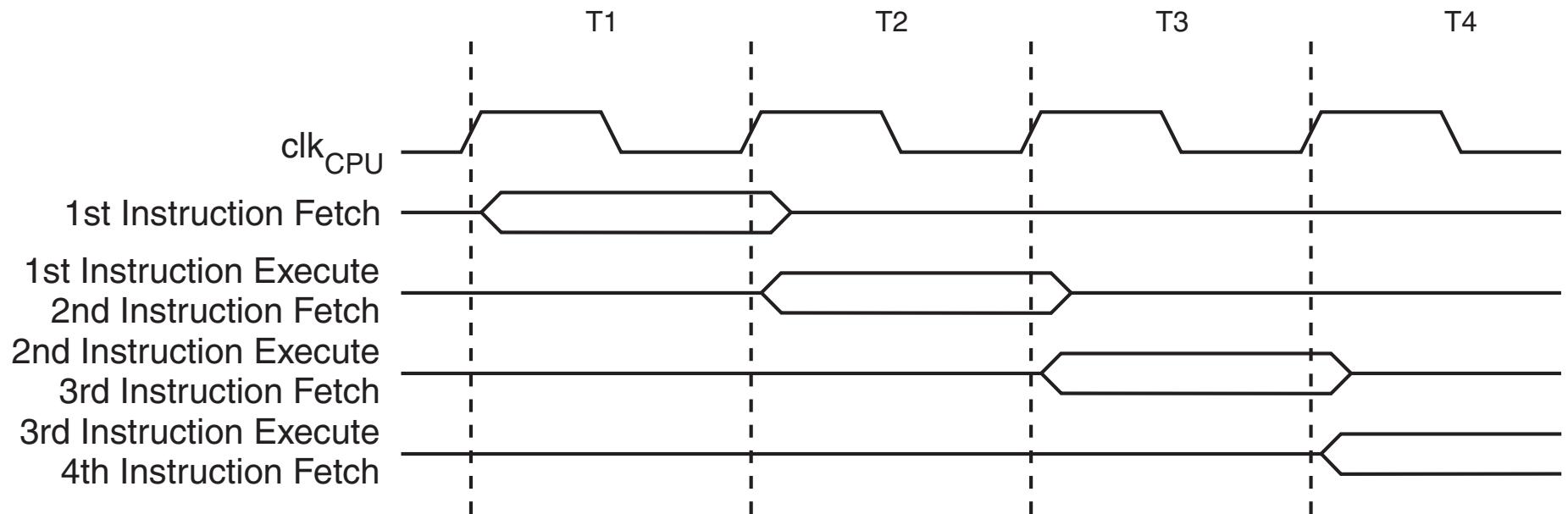


Address: 16-64 bits   Data: 8-64 bits

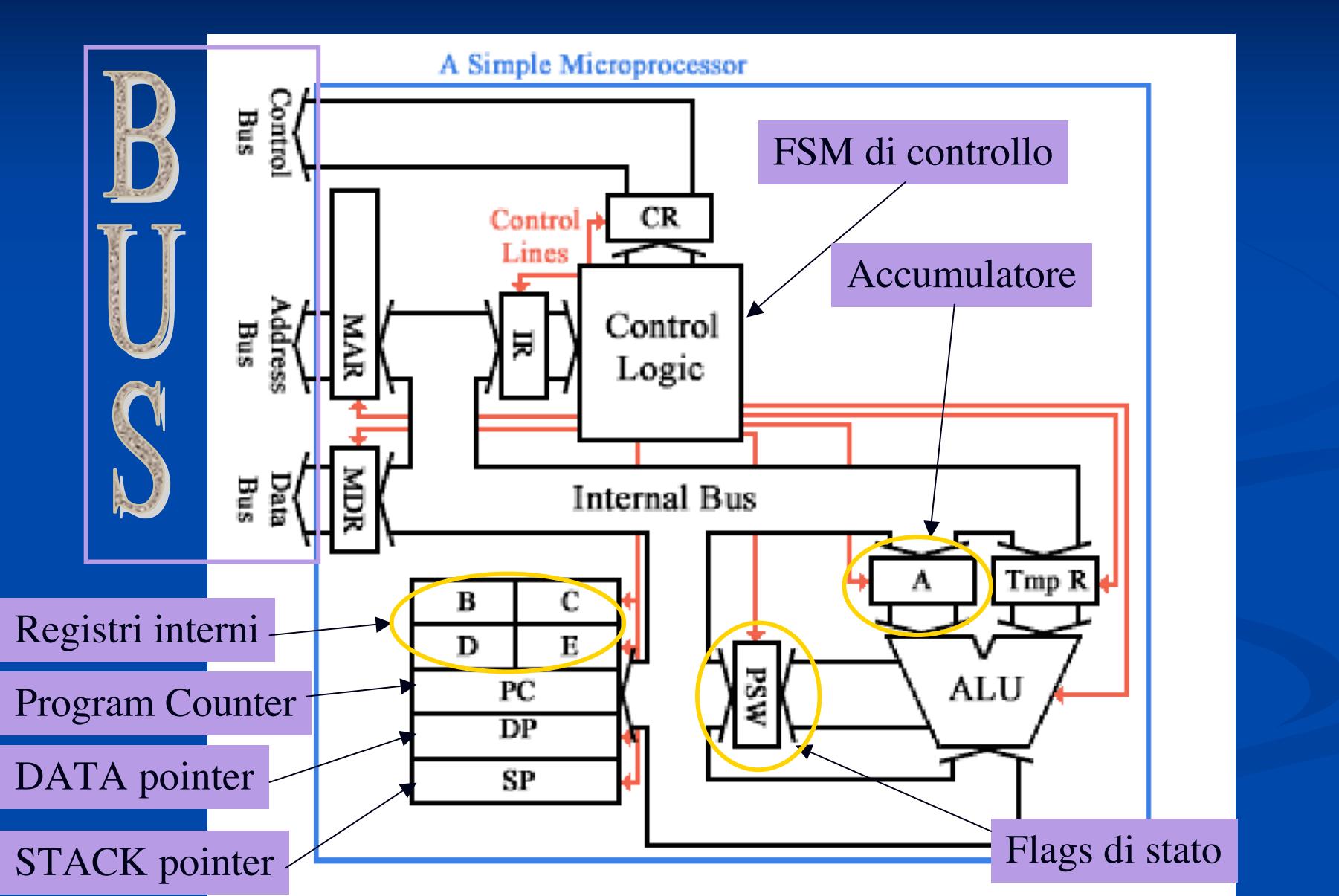
# Instruction timing

17

The Parallel Instruction Fetches and Instruction Executions

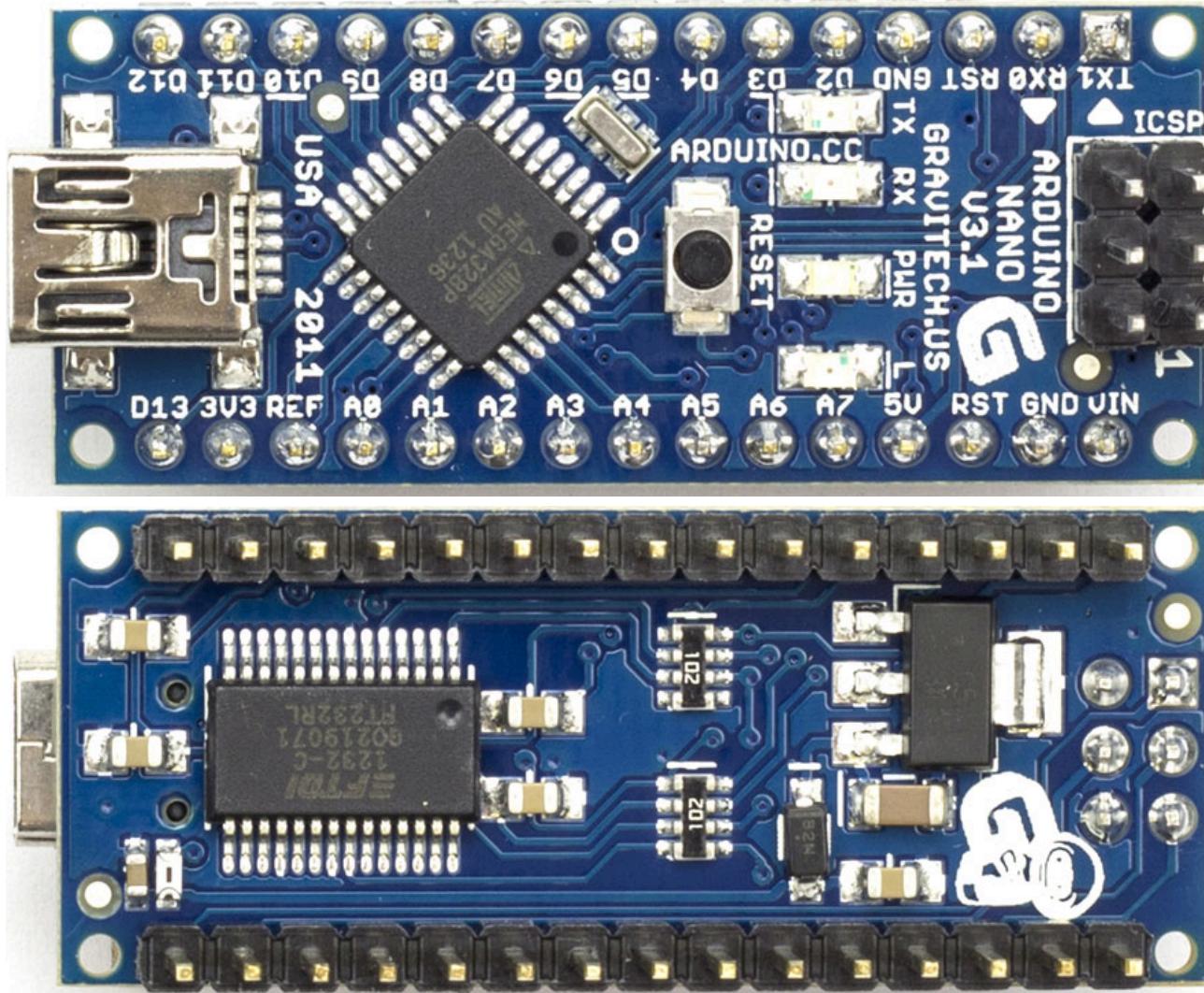


# Struttura di una CPU tipica



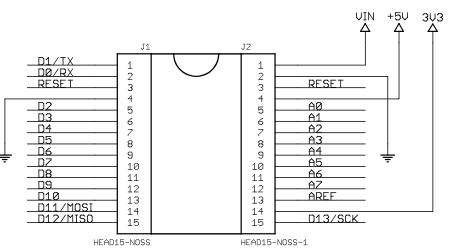
# Arduino nano / ATmega 328

19



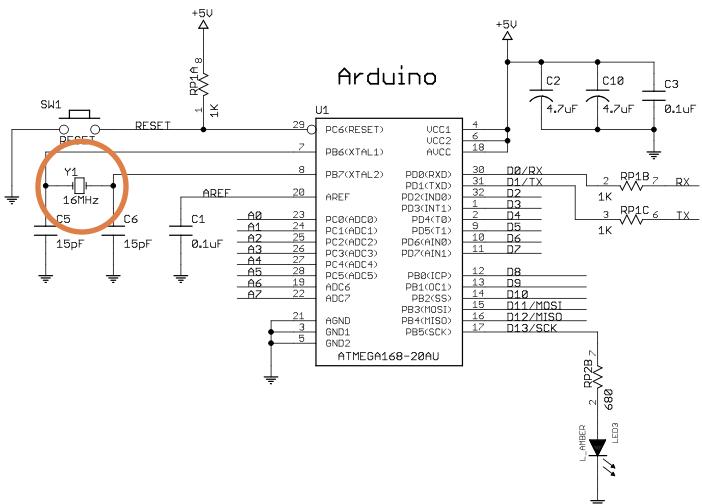
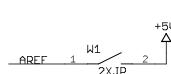
# Arduino Nano Schematic

Copyright 2008 under the Creative Commons Attribution Share-Alike 2.5 License  
<http://creativecommons.org/licenses/by-sa/2.5/>

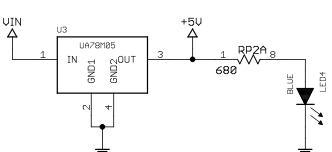


**16MHz**

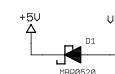
**+5V AREF OPTION**



**+5V REG**



**+5V AUTO SELECTOR**



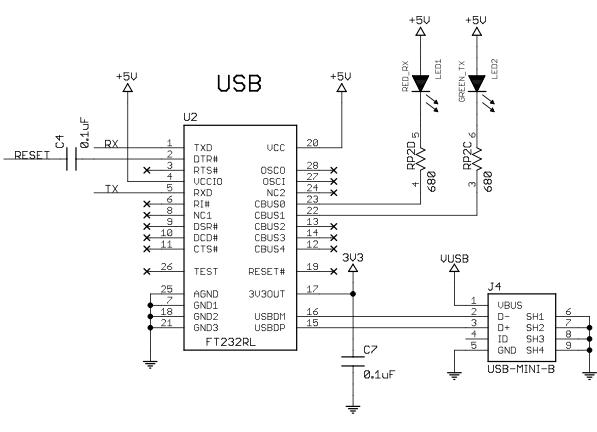
**NOT USED**



x 4.7k

RP1D\_5

1k



v2.3 - Modify FT232RL to use +5V

TITLE: Arduino Nano

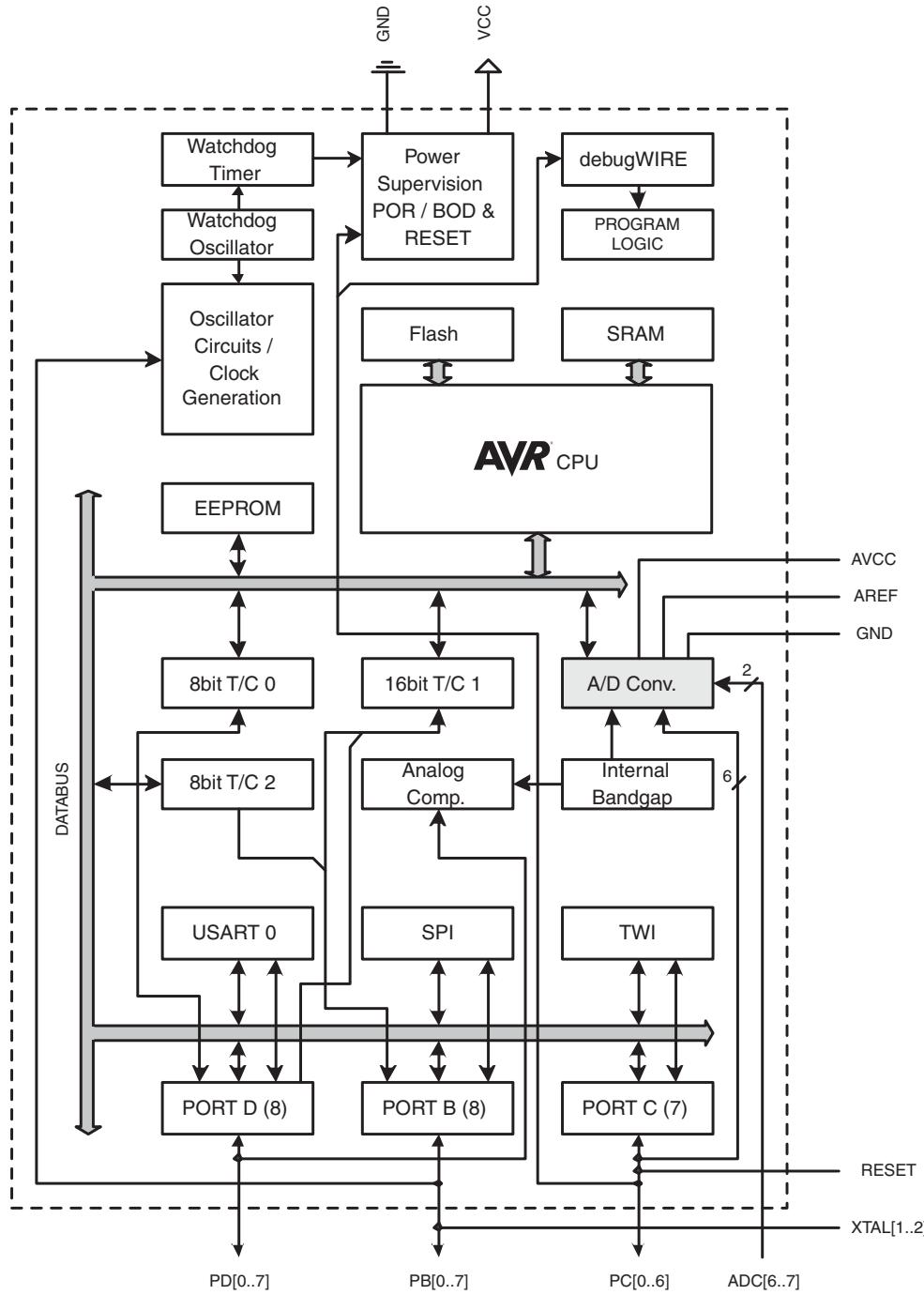
Document Number:

REV: 2.3

Date: 6/26/2008 8:35:54 PM

Sheet: 1/1

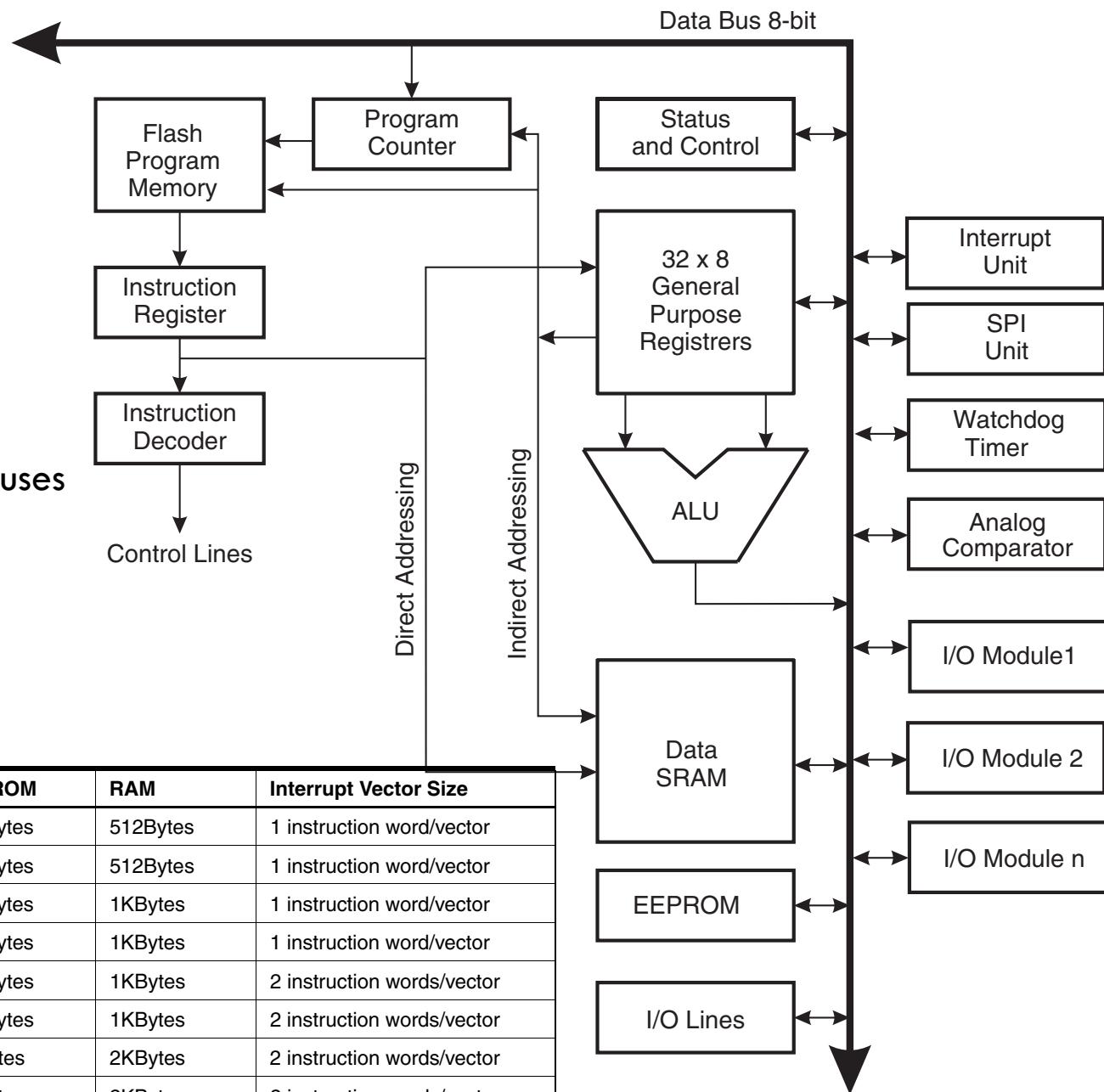
# ATMEL 368 BLOCK DIAGRAM



# ATMEL 368

## CPU Detail

**Harvard architecture**  
**Separate data and program memories/buses**



Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48A	4KBytes	256Bytes	512Bytes	1 instruction word/vector
ATmega48PA	4KBytes	256Bytes	512Bytes	1 instruction word/vector
ATmega88A	8KBytes	512Bytes	1KBytes	1 instruction word/vector
ATmega88PA	8KBytes	512Bytes	1KBytes	1 instruction word/vector
ATmega168A	16KBytes	512Bytes	1KBytes	2 instruction words/vector
ATmega168PA	16KBytes	512Bytes	1KBytes	2 instruction words/vector
ATmega328	32KBytes	1KBytes	2KBytes	2 instruction words/vector
ATmega328P	32KBytes	1KBytes	2KBytes	2 instruction words/vector

# Memorie

23

- Tutti i processori fanno uso di memorie per immagazzinare dati ed istruzioni.
- La memoria e` organizzata in modo gerarchico:
  - La memoria piu` veloce e` realizzata nel processore sotto forma di registri
  - Immediatamente a livello piu` alto e` la RAM interna al chip
  - Infine spesso si possono gestire memorie esterne al chip
- In Arduino abbiamo diverse memorie interne (pochi kbytes):
  - SRAM (static RAM), FLASH, EEPROM
- Nei computer le memorie sono esterne al processore
  - Molti Gbytes – piu` lente delle memorie interne

# Memorie

24

Vari tipi di RAM (random access memory):

- SRAM (Static RAM): l'elemento di memoria e` il FF ed i dati vengono mantenuti finche` la memoria e` alimentata. Si possono ottenere memorie molto veloci, tempo di accesso ca. 1 ns. L'utilizzo e` limitato da due fattori:
  - Alto Consumo
  - Complessita` realizzativa (vari transistor per cella).
- DRAM (Dynamic RAM): la memoria consiste in pratica in un condensatore (in effetti condensatori di dimensioni micrometriche integrati in un chip). Possono essere di dimensioni elevate ma hanno bisogno di continui cicli di "refresh", in pratica i condensatori tendono a scaricarsi, per mantenere la memorizzazione.
  - Tempi di accesso di decine di ns
  - Necessitano di tutta la circuiteria per i cicli di refresh
  - Economiche e basso consumo
- NVRAM (Non Volatile RAM): mantiene l'informazione anche a alimentazione spenta. Si realizza integrando in uno stesso chip una RAM Statica con una EEPROM.
- FLASH RAM: e` un particolare tipo di EEPROM piu` veloce e flessibile.
  - Ormai disponibili per fare mass storage a stato solido (TBytes)

# Le caratteristiche di base di una CPU

25

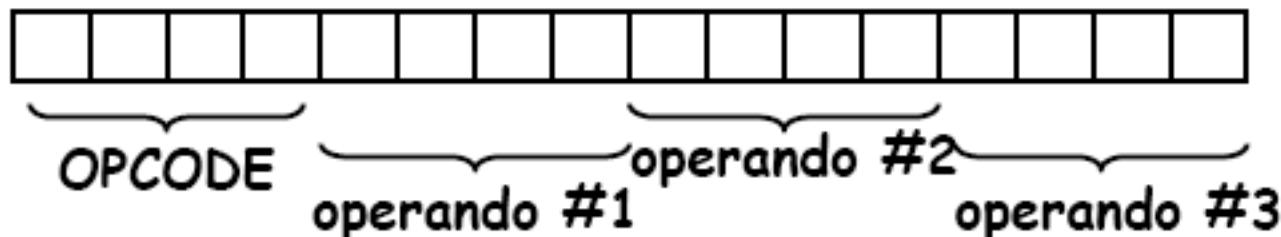
Gli elementi base che caratterizzano una CPU sono:

- l'insieme di istruzioni disponibili
  - Architettura RISC: Reduced Instruction Set Computer, istruzioni semplici con esecuzione veloce
  - Architettura CISC: Complex Instruction Set Computer, istruzioni più complesse ma con più esecuzione lenta
- frequenza di clock, velocità con cui le istruzioni vengono eseguite
  - Normalmente i mC non vengono realizzati con frequenze di clock molto elevate, tipicamente lavorano a qualche decina di MHz
  - Le CPU dei computer lavorano a GHz
- organizzazione hardware (numero di bit dei bus, risorse disponibili)
  - in generale si hanno una o più ALU (Arithmetic Logic Unit), vari registri, circuiti e sistemi di controllo

# Le istruzioni tipiche di un mC

26

- I mC sono nati per implementare operazioni di controllo (dispositivi simili pensati per il calcolo sono denominati DSP: Digital Signal Processor)
- Molto spesso operazioni come la moltiplicazione esistevano solo come istruzioni di base. Nel mC noi non esiste neanche come istruzione base. L'esecuzione della moltiplicazioni richiedera` quindi un certo nro di istruzioni di base.
- In generale ogni singola istruzione dovrà contenere le seguenti informazioni:
  - Operazione da eseguire: questa parte si indica con codice operativo (OPCODE)
  - Se necessario: indirizzo dove trovare il/i dati con cui eseguire l'operazione (operandi)
  - Se necessario: La destinazione del dato risultato



# Some ATmega328 instructions

Mnemonics	Operands	Description	Operation	Flags	#Clocks
<b>ARITHMETIC AND LOGIC INSTRUCTIONS</b>					
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	$Rdh:Rdl \leftarrow Rdh:Rdl + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \leftarrow Rd - K - C$	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	$Rdh:Rdl \leftarrow Rdh:Rdl - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \bullet Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \bullet K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \vee K$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	$Rd \leftarrow 0xFF - Rd$	Z,C,N,V	1
NEG	Rd	Two's Complement	$Rd \leftarrow 0x00 - Rd$	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \bullet (0xFF - K)$	Z,N,V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
SER	Rd	Set Register	$Rd \leftarrow 0xFF$	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \ll 1$	Z,C	2

**BRANCH INSTRUCTIONS**

RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
IJMP		Indirect Jump to (Z)	$PC \leftarrow Z$	None	2
JMP <sup>(1)</sup>	k	Direct Jump	$PC \leftarrow k$	None	3
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
ICALL		Indirect Call to (Z)	$PC \leftarrow Z$	None	3
CALL <sup>(1)</sup>	k	Direct Subroutine Call	$PC \leftarrow k$	None	4
RET		Subroutine Return	$PC \leftarrow STACK$	None	4
RETI		Interrupt Return	$PC \leftarrow STACK$	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) $PC \leftarrow PC + 2$ or 3	None	1/2/3
CP	Rd,Rr	Compare	$Rd - Rr$	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	$Rd - Rr - C$	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	$Rd - K$	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) $PC \leftarrow PC + 2$ or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) $PC \leftarrow PC + 2$ or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then $PC \leftarrow PC + k + 1$	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then $PC \leftarrow PC + k + 1$	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then $PC \leftarrow PC + k + 1$	None	1/2

**DATA TRANSFER INSTRUCTIONS**

MOV	Rd, Rr	Move Between Registers	$Rd \leftarrow Rr$	None	1
MOVW	Rd, Rr	Copy Register Word	$Rd+1:Rd \leftarrow Rr+1:Rr$	None	1
LDI	Rd, K	Load Immediate	$Rd \leftarrow K$	None	1
LD	Rd, X	Load Indirect	$Rd \leftarrow (X)$	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	$Rd \leftarrow (X), X \leftarrow X + 1$	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	$X \leftarrow X - 1, Rd \leftarrow (X)$	None	2
LD	Rd, Y	Load Indirect	$Rd \leftarrow (Y)$	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	$Rd \leftarrow (Y), Y \leftarrow Y + 1$	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	$Y \leftarrow Y - 1, Rd \leftarrow (Y)$	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	$Rd \leftarrow (Y + q)$	None	2
LD	Rd, Z	Load Indirect	$Rd \leftarrow (Z)$	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	$Rd \leftarrow (Z), Z \leftarrow Z+1$	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	$Z \leftarrow Z - 1, Rd \leftarrow (Z)$	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	$Rd \leftarrow (Z + q)$	None	2
LDS	Rd, k	Load Direct from SRAM	$Rd \leftarrow (k)$	None	2
ST	X, Rr	Store Indirect	$(X) \leftarrow Rr$	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	$(X) \leftarrow Rr, X \leftarrow X + 1$	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	$X \leftarrow X - 1, (X) \leftarrow Rr$	None	2
ST	Y, Rr	Store Indirect	$(Y) \leftarrow Rr$	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	$(Y) \leftarrow Rr, Y \leftarrow Y + 1$	None	2

# ALU: Unita` Logica Aritmetica

29

- E` il dispositivo logico che implementa le operazioni aritmetiche e/o logiche (AND, OR, ...) con i dati
- La ALU e` progettata per eseguire operazioni con dati a 4, 8, 16, 32 bit a seconda della architettura del microprocessore
- I dati elaborati dalla ALU vengono ospitati in opportuni registri. Esistono diverse filosofie di implementazione della ALU che possono richiedere l'utilizzazione di un nro diverso di registri.
- Nel caso piu` semplice si ha un registro detto accumulatore e le istruzioni agiscono su questo registro. Ad esempio:

ADD	Rd, Rr	Add two Registers	Rd $\leftarrow$ Rd + Rr
-----	--------	-------------------	-------------------------

- ADD Rd,Rr il contenuto del registro Rr viene aggiunto al contenuto del registro accumulatore Rd e immagazzinato in Rd



# Unita` Logica Aritmetica

30

- Lo svolgimento delle funzioni tipiche della ALU richiede circuiti del tipo:
  - Reti combinatorie per funzioni logiche
  - Circuiti sommatori/sottrattori
  - Registri a scorrimento per le operazioni di shift
  - Multiplexer
  - Eventualmente circuiti per moltiplicazioni/divisioni
- Il ATmega328 ha una ALU a 8 bit capace di svolgere addizioni, sottrazioni (utilizzando il complemento a 2), shift e operazioni logiche
  - Ha 32 registri a 8 bit di uso generale

7	0	Addr.	
	R0	0x00	
	R1	0x01	
	R2	0x02	
	...		
	R13	0x0D	
	R14	0x0E	
	R15	0x0F	
	R16	0x10	X-register Low Byte
	R17	0x11	X-register High Byte
	...		
	R26	0x1A	Y-register Low Byte
	R27	0x1B	Y-register High Byte
	R28	0x1C	Z-register Low Byte
	R29	0x1D	Z-register High Byte
	R30	0x1E	
	R31	0x1F	

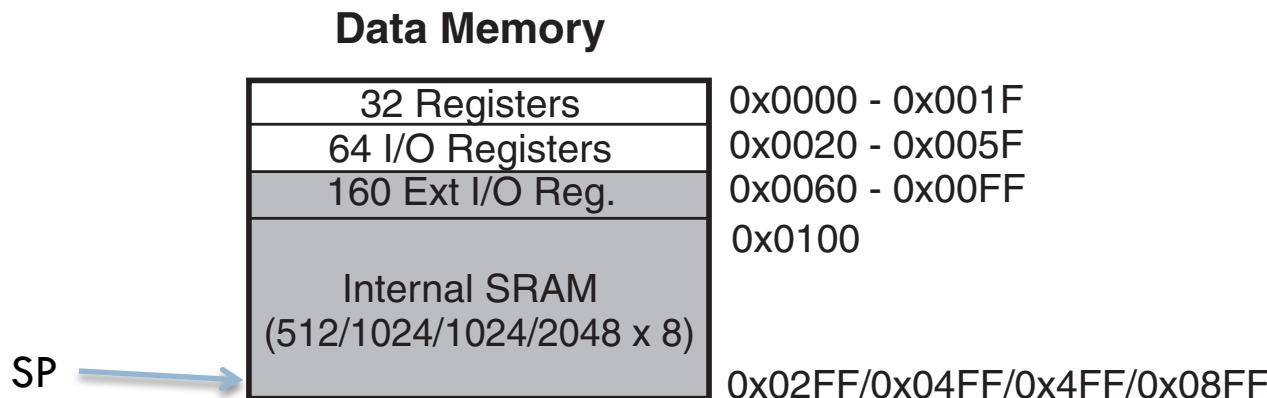
Memory-mapped registers

# Altri registri della CPU

31

Oltre ai registri per i dati della ALU esistono altri tipi di registri della CPU:

- program counter (PC): contiene l'indirizzo nella memoria programmi della prossima istruzione da eseguire (anche detto INSTRUCTION POINTER IP).
- status register (SREG): registro che contiene informazioni riguardo allo stato dell'esecuzione delle operazioni (es.: overflow), reset.
- Stack pointer (SP): serve per salvare lo stato e ricordare dove tornare dopo interrupt e subroutines: di solito parte dal fondo
- Registri per gestione/configurazione periferiche e I/O



# Comunicare con il mondo: dispositivi di I/O

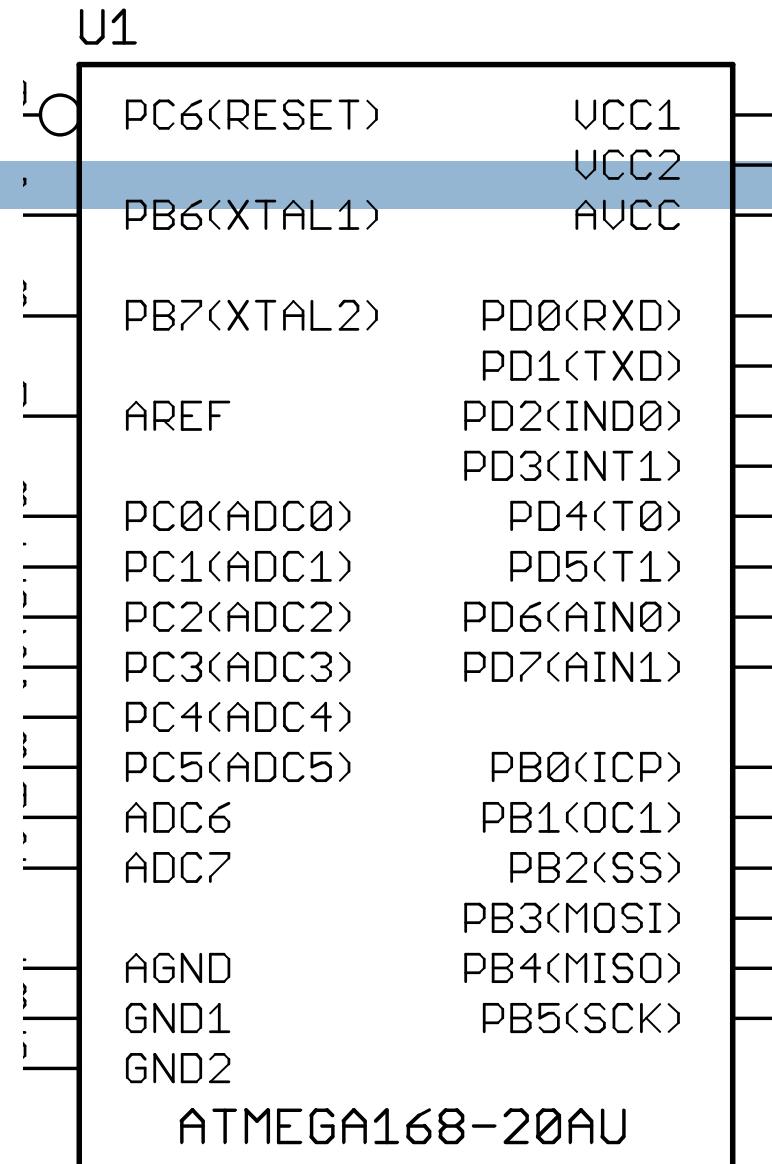
32

- Uno dei principali utilizzi dei mC e` la gestione di sensori o macchine e` quindi importante la possibilita` di comunicare con l'esterno
- Per fare un'esempio vedrete in laboratorio che il mC controlla l'accensione/spegnimento di LED, un sensore di temperatura e degli interruttori ...
- Tutto cio` che permette di comunicare con il mondo si dice indica con il nome generico di dispositivo di Input/Output
- Quasi tutti i pin del chip che utilizziamo possono essere utilizzati come dispositivi di I/O, alcuni fungono da interfaccie con particolari dispositivi interni
- Per esempio la comunicazione tra arduino e computer avviene attraserso la porta USB che comunica con il mC attraverso due pin particolari che sono interfacciati ad un delle periferiche interne detto USART (Universal Synchronous/Asynchronous Receiver/Transmitter) che gestisce la comunicazione seriale.

# Pin diagram

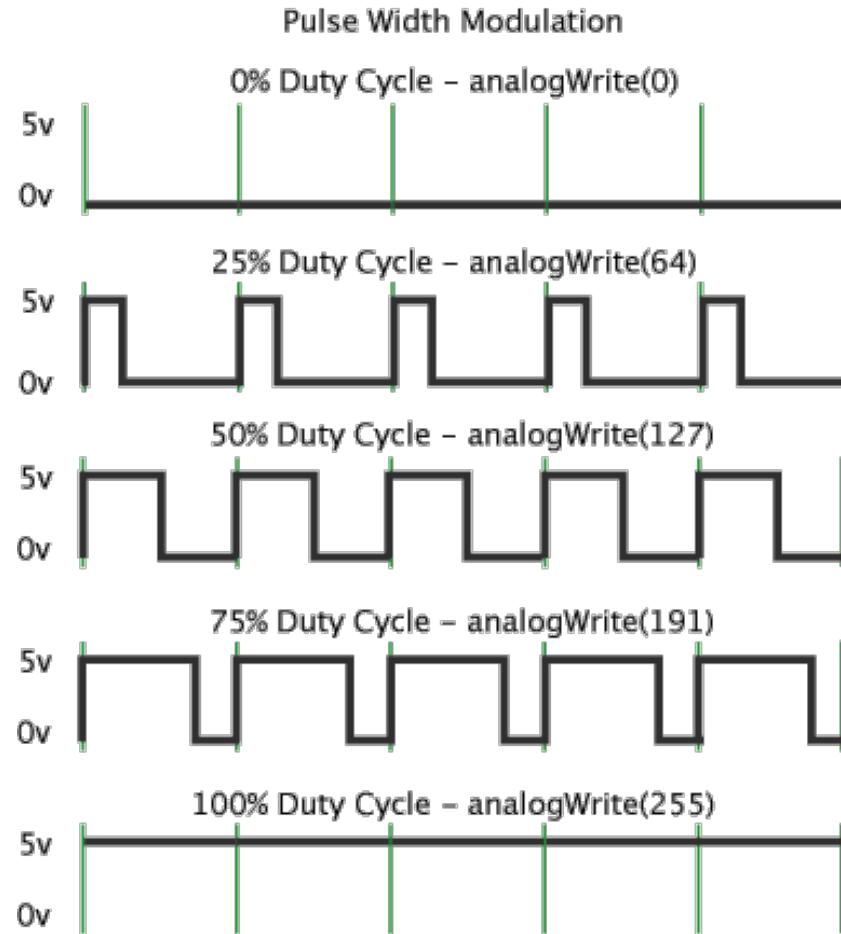
33

- I pin di I/O dell'ATMEGA328 sono organizzati in 3 porte (B, C, D)
- I pin sono in generale configurabili per ingresso/uscita digitale
- Altri pin possono essere collegati all'ADC (conversione analogico-digitale)
- Per generare una segnale analogico si usa la Pulse Width Modulation
  - Segnale digitale di cui si varia il duty cycle: se filtrato a bassa frequenza da un segnale analogico proporzionale al duty cycle
- Esistono dei timer/counter usabili per generare segnali di durata determinata



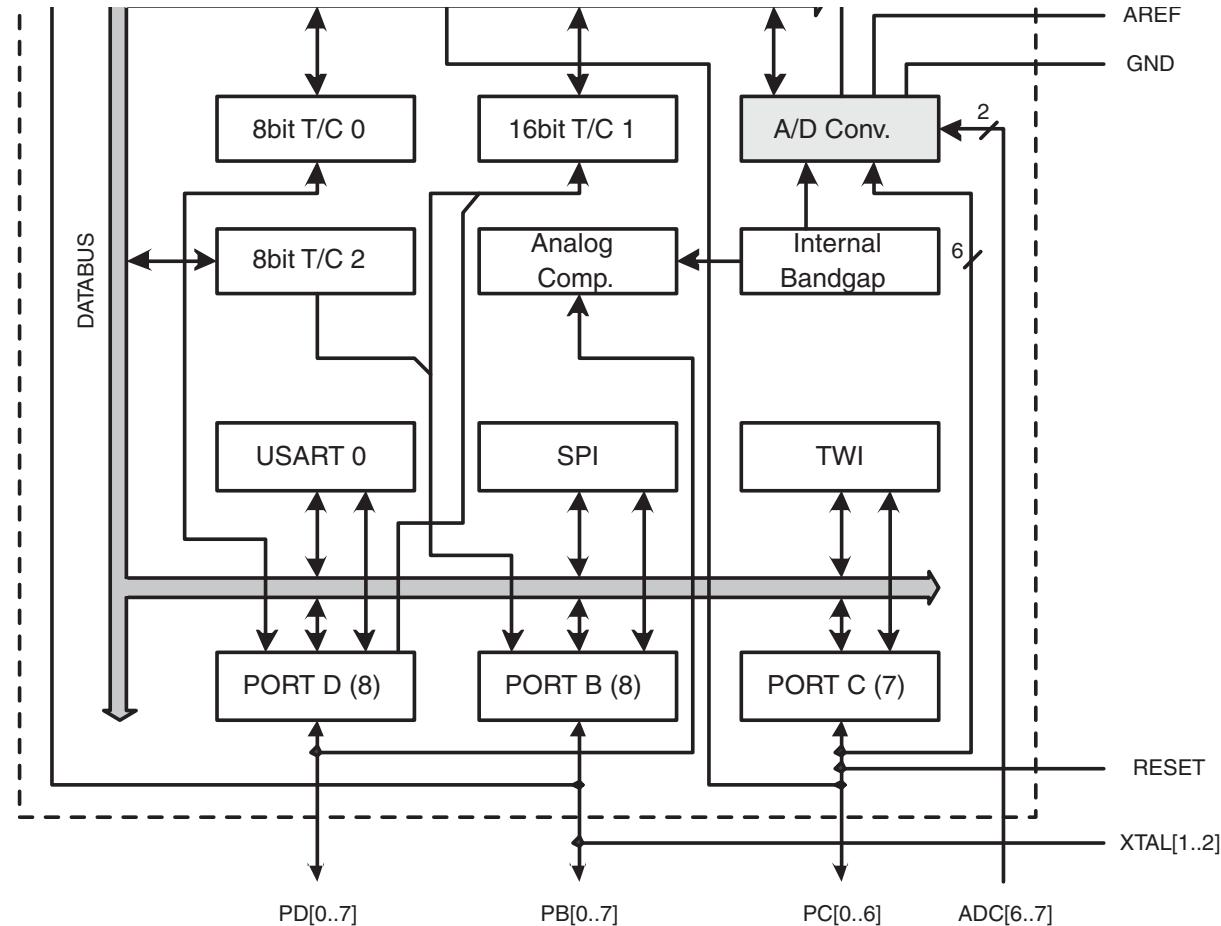
# Pulse width modulation

34



# Arduino I/O subsystem

35



# Circuito di I/O digitale

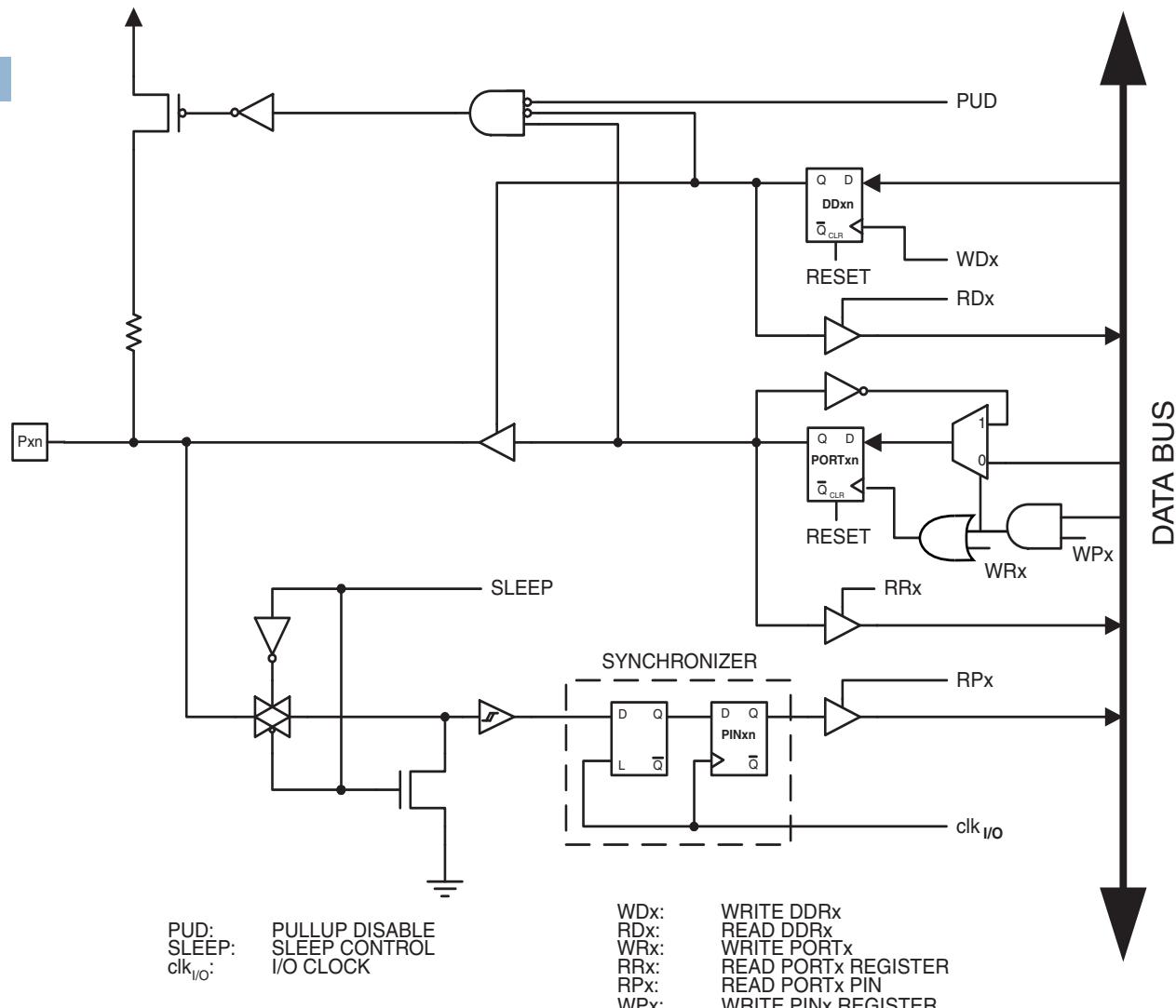
36

Programmabile se input o output

Uscita tri-state disabilitata quando si usa come ingresso

Leggibile e scrivibile dal data bas.

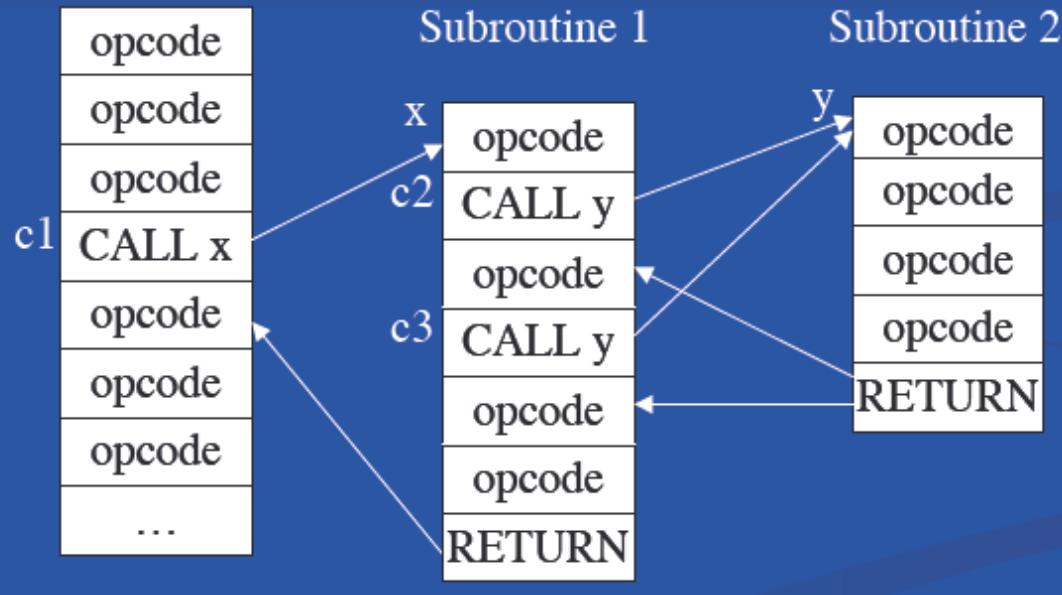
Molte caratteristiche programmabili (ad esempio la resistenza di pull-up)



# Esecuzione di una Subroutine

37

Main program



Il programma principale viene “sospeso” all’indirizzo  $c_1$  da cui si inizia l’esecuzione della Subroutine 1.

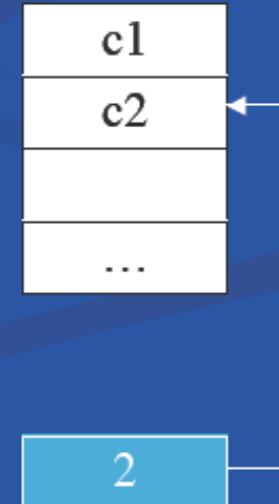
Un registro apposito detto **Stack Pointer (SP)** tiene in memoria il puntatore alla prima locazione libera nel registro detto STACK. L’indirizzo  $c_1$  viene salvato nella prima locazione libera dello STACK e SP viene incrementato.

Il primo indirizzo della Subroutine 1 viene copiato nel PC e la subroutine viene eseguita ...

**CALL x:**

```
[SP] ← PC  
SP ← SP+1  
PC ← x  
RETURN:  
SP ← SP-1  
PC ← [SP]
```

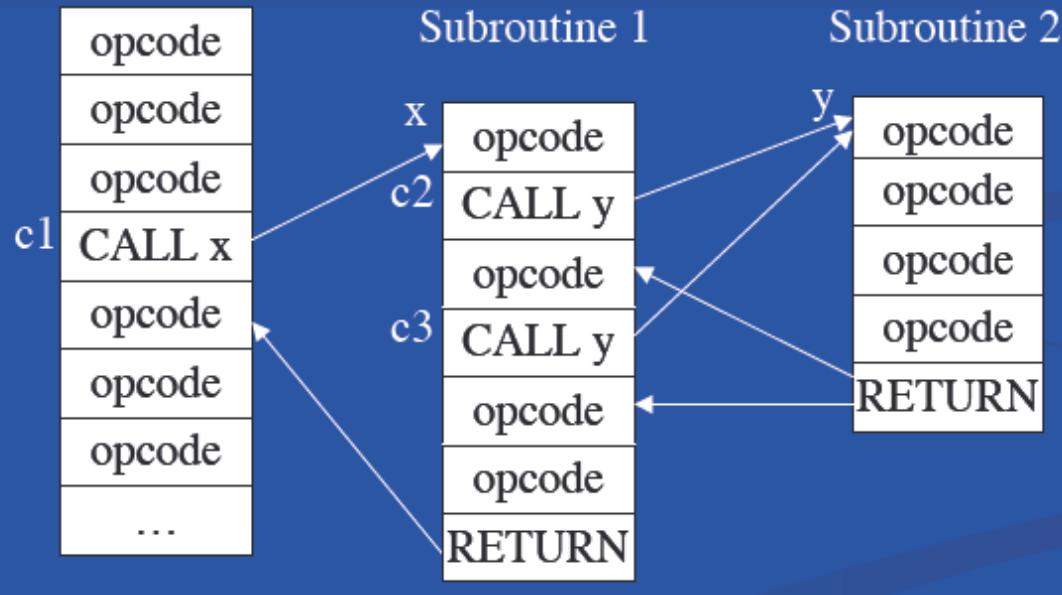
**STACK**



# Esecuzione di una Subroutine

38

Main program



Se una seconda Subroutine viene chiamata in Subroutine 1

(CALL y) la procedura si ripete:

C2 viene copiato nello STACK

SP viene incrementato

L'indirizzo y viene copiato nel PC e la esecuzione di  
Subroutine 2 inizia

Alla fine c2 viene copiato nel PC e SP viene decrementato

CALL x:

$[SP] \leftarrow PC$

$SP \leftarrow SP+1$

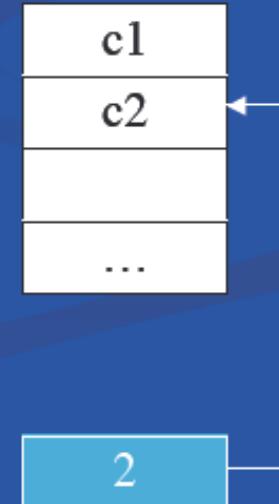
$PC \leftarrow x$

RETURN:

$SP \leftarrow SP-1$

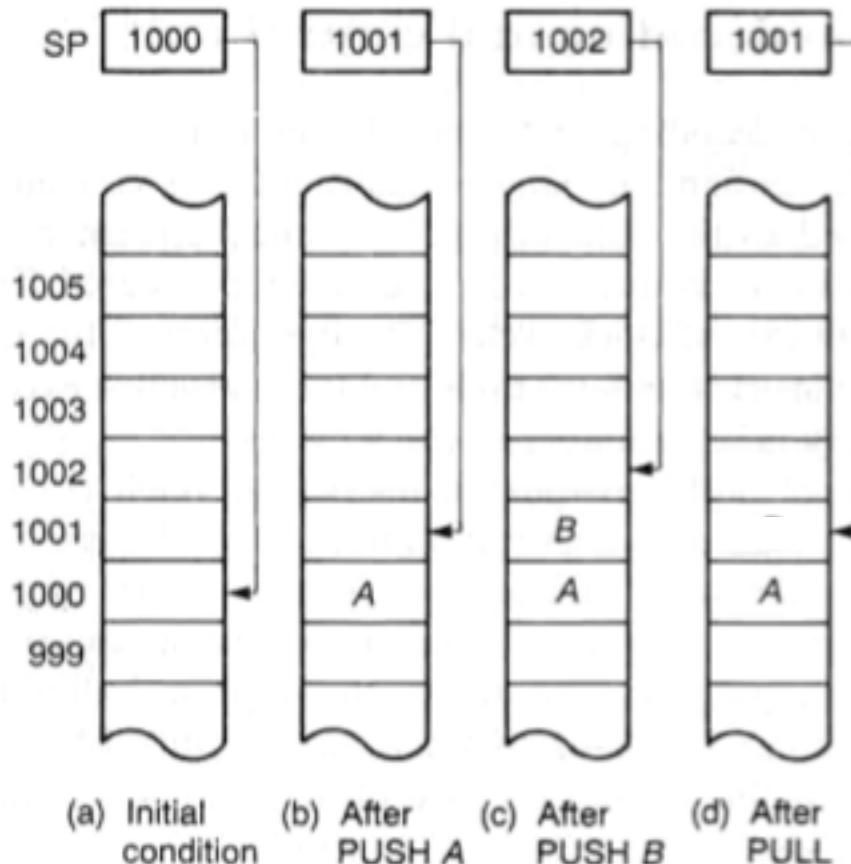
$PC \leftarrow [SP]$

STACK



# Funzionamento dello Stack (pila)

39



E` una componente di memoria che si dice LIFO: **Last In First Out**

Ogni volta che un dato viene inserito il puntatore in memoria viene automaticamente incrementato

Viceversa ogni volta che un dato viene estratto solo quello che e` "piu` in alto" puo` essere estratto ed il puntatore viene decrementato.

SLANG o istruzioni ...

PUSH == Scrittura Dato

PULL == Lettura Dato

Esempio di introduzione ed estrazione dati dallo  
STACK

# Esecuzione di una subroutine e pipeline

40

- Poiche` il nostro mC funziona con la pipeline delle istruzioni nel momento in cui incontra la chiamata alla subroutine l'istruzione successiva che viene caricata deve essere scaricata per eseguire la subroutine
- Si vede quindi perche` una istruzione che richiede un “programa branch” ha bisogno di 2 Tcy anziche` uno solo

	Tcy0	Tcy1	Tcy2	Tcy3	Tcy4	Tcy5
1. MOVLW 55h	Fetch 1	Execute 1				
2. MOVWF PORTB		Fetch 2	Execute 2			
3. CALL SUB_1			Fetch 3	Execute 3		
4. BSF PORTA, BIT3 (Forced NOP)				Fetch 4	Flush	
5. Instruction @ address SUB_1					Fetch SUB_1	Execute SUB_1
						Fetch SUB_1 + 1

All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is “flushed” from the pipeline while the new instruction is being fetched and then executed.

# Il concetto di INTERRUPT

41

- La funzionalita` del microControllore che abbiamo visto appare piuttosto vasta per la piccola dimensione e costo del chip tuttavia manca una ulteriore funzione molto importante: la possibilita` di “attirare l’attenzione” della CPU su un evento che succede in un momento non prevedibile.
- Abbiamo visto che abbiamo la possibilita` di far accendere un LED in base alla pressione su un tasto. In questo caso la CPU controlla continuamente se il tasto e` stato premuto ed in caso positivo accende il LED
- Nel caso in cui avessi molti tasti, o piu` in generale molte sorgenti di input, dovrei continuare a controllare ciclicamente tutte le sorgenti da cui mi posso aspettare un input. Questa operazione si chiama *polling* e costa molto tempo
- Questa funzione viene implementata in modo molto piu` efficace con metodo che si chiama *interrupt request query (IRQ)*

# Il concetto di INTERRUPT

42

- In pratica esiste un semplice circuito indipendente dalla CPU che genera un segnale (interrupt) che indica alla CPU che uno degli “eventi attesi” si e` verificato
- La CPU interrompe le operazione che stava eseguendo e si occupa di eseguire le operazioni richieste per gestire il tipo di interrupt che si e` verificato. In pratica la CPU sospende il programma in esecuzione, esegue la routine apposita (interrupt handler) e poi riprende l'esecuzione del programma
- Quando le sorgenti di interrupt sono piu` di una la CPU deve avere un modo per verificare quale e` la sorgente dell'interrupt
- A volte la CPU esegue delle operazioni che non devono essere interrotte. In questo caso esiste la possibilita` di disabilitare temporaneamente i segnale di interrupt.
- Il numero, tipo e livelli di interrupt dipende fortemente dal tipo di processore o microcontrollore
- In generale: uso degli interrupt essenziale per velocità ed efficienza, ma... molto delicato e difficile da capire e debuggare perche' non dipende da eventi esterni

# Gestione del segnale di interrupt

43

Quando il segnale di interrupt viene ricevuto la CPU:

- Clear il bit di Global Interrupt Enable (GIE) per fare in modo di disabilitare qualsiasi altro interrupt
- Fa il PUSH del del registro Program Counter sullo stack e nel PC viene copiato il valore del vettore di interrupt corrispondente all'evento che si e' verificato
- In generale nessun valore dei registri viene salvato a parte PC
  - Necessario salvare a mano i registri che si usano
- Una volta che l'interrupt e` stato servito il bit GIE viene rimesso a 1 e nuovi interrupt possono essere serviti
- Si deve fare un restore di tutti i registri modificati.
- Si fa un PULL del PC dallo stack e si riprende da dove si era interrotto.
  - In linea di principio non si dovrebbe accorgere che c'e' stato un interrupt.

Vector = indirizzo di memoria da caricare nel PC

VectorNo.	Program Address <sup>(2)</sup>	Source	Interrupt Definition
1	0x0000 <sup>(1)</sup>	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface
26	0x0032	SPM READY	Store Program Memory Ready

## L'inizio della memoria di programma - RESET

The most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega328/328P is:

Address	Labels	Code	Comments
0x0000		jmp RESET	; Reset Handler
0x0002		jmp EXT_INT0	; IRQ0 Handler
0x0004		jmp EXT_INT1	; IRQ1 Handler
0x0006		jmp PCINT0	; PCINT0 Handler
0x0008		jmp PCINT1	; PCINT1 Handler
0x000A		jmp PCINT2	; PCINT2 Handler
0x000C		jmp WDT	; Watchdog Timer Handler
0x000E		jmp TIM2_COMPA	; Timer2 Compare A Handler
0x0010		jmp TIM2_COMPB	; Timer2 Compare B Handler
0x0012		jmp TIM2_OVF	; Timer2 Overflow Handler
0x0014		jmp TIM1_CAPT	; Timer1 Capture Handler
;			
0x0033	RESET:	ldi r16, high(RAMEND); Main program start	
0x0034		out SPH, r16 ; Set Stack Pointer to top of RAM	
0x0035		ldi r16, low(RAMEND)	
0x0036		out SPL, r16	
0x0037		sei ; Enable interrupts	
0x0038		<instr> xxx	
... ... ... ...			

# Aiuto dal compilatore

46

- Il compilatore che si prende cura di generare la routine che controlla quale e` la sorgente dell'interrupt
- Nessun registro viene salvato automaticamente a parte il valore del PC e` quindi necessario che all'inizio dell'esecuzione della Subroutine tutti i registri di cui non devo perdere il valore siano salvati.
- Il compilatore genera il codice per salvare molti dei registri utili (es.: W, Status Register...)

# Programmazione di Arduino

47

- Arduino viene con un sistema di sviluppo Arduino IDE (= Integrated Development Environment)
  - <http://arduino.cc/en/Main/Software>
- Permette di:
  - Collegarsi ad arduino via USB
  - Scrivere programmi (“sketch”) per arduino in c
    - Organizzabili in librerie
  - Compilare e caricare i programmi su arduino
  - Comunicare con arduino via USB durante il funzionamento

# Da definire (almeno) due funzioni

48

```
void setup() {  
    // put your setup code here, to run once:  
  
}
```

```
void loop() {  
    // put your main code here, to run repeatedly:  
  
}
```

Non si definisce la main

```
/*
```

## Blink

Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.

```
*/
```

```
// Pin 13 has an LED connected on most Arduino boards.
```

```
// give it a name:
```

```
int led = 13;
```

```
// the setup routine runs once when you press reset:
```

```
void setup() {
```

```
    // initialize the digital pin as an output.
```

```
    pinMode(led, OUTPUT);
```

```
}
```

```
// the loop routine runs over and over again forever:
```

```
void loop() {
```

```
    digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
```

```
    delay(1000);           // wait for a second
```

```
    digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
```

```
    delay(1000);           // wait for a second
```

```
}
```

## Esempio: blink



Compila e  
carica su  
arduino

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
    // initialize the digital pin as an output.
    pinMode(led, OUTPUT);
}

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

Porta seriale a cui e' connesso

Auto Format ⌘T  
Archive Sketch  
Fix Encoding & Reload  
Serial Monitor ⌘M

Board ▶  
Serial Port ▶  
Programmer ▶  
Burn Bootloader ▶

five seconds of the sketch  
maximum of expected values

Final values may seem backwards.

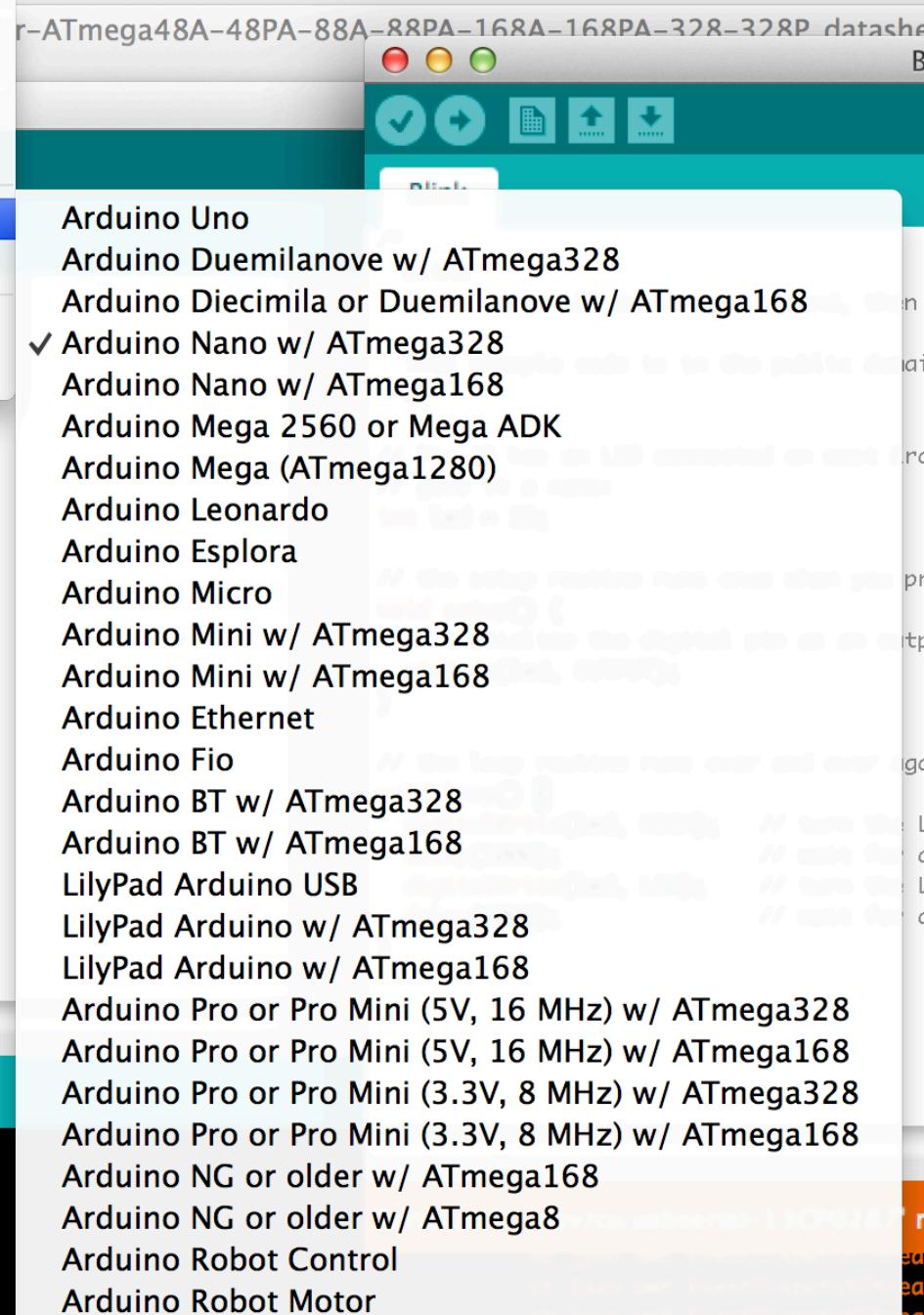
Bisogna specificare il modello

Spesso ci sono problemi con i driver

Cercate sul web per le istruzioni

to ground

operation



Molti sketch di esempio da dove partire

The screenshot shows the Arduino IDE interface. On the left, a sketch titled "toneMelody" is displayed. The code uses the tone() function to play a melody. On the right, the "File" menu is open, showing various examples categorized by topic. The visible categories include:

- Calibration
- 01.Basics
- 02.Digital
- 03.Analog
- 04.Communication
- 05.Control
- 06.Sensors
- 07.Display
- 08.Strings
- 09.USB
- 10.StarterKit
- ArduinoISP
- EEPROM
- Esplora
- Ethernet
- Firmata
- GSM
- LiquidCrystal
- Robot\_Control
- Robot\_Motor
- SD
- Servo
- SoftwareSerial
- SPI
- Stepper
- TFT
- WiFi
- Wire

The "toneMelody" sketch is visible in the code editor area.

# Dove trovare gli argomenti di oggi

53

- cap. 3 Introduzione all'elettronica – parte I – E.Falchini et al.
- Un libro sulla struttura dei calcolatori: D.Patterson & J.Hennessy – Struttura e Progetto dei calcolatori – Zanichelli
- Un po` di storia:<http://cronologia.leonardo.it/storia/tabello/tabe1559.htm>
- Un corso (dettagliato) sulle computer architecture
  - <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-823-computer-system-architecture-fall-2005/lecture-notes/>
- Arduino: [www.arduino.cc](http://www.arduino.cc)