I MICROCONTROLLORI

CARATTERISTICHE BASE DI UN MICROCONTROLLORE E DIFFERENZE CON UN MICROPROCESSORE

FONTI

- https://slideplayer.it/slide/13714625/
- Pietro Di Lena «Cenni sull'architettura del calcolatore» presentazione nel sito dell'Università di Cesena
- http://studentidoc.altervista.org/guida/il-microcontrollore/

INTRODUZIONE

- Microcontrollori e microprocessori sono dispositivi programmabili per memorizzare ed elaborare dati
- Per funzionare si basano su due componenti:
  - l'hardware che consiste delle componenti fisiche
  - il software che consiste di istruzioni che possono essere immagazzinate ed eseguite dalla parte hardware
- Microcontrollori e microprocessori sono tuttavia due cose diverse
  - I sistemi embedded basano il loro funzionamento sui microcontrollori
    - Un sistema embedded identifica genericamente tutti quei sistemi elettronici di elaborazione a microprocessore progettati appositamente per un determinato utilizzo (special purpose), ovvero non riprogrammabili dall'utente per altri scopi, spesso con una piattaforma hardware ad hoc, integrati nel sistema che controllano e in grado di gestirne tutte o parte delle funzionalità richieste
  - I calcolatori in genere (tra cui i PC) funzionano grazie ai microprocessori
  - Arduino si basa invece sul microcontrollore Atmega328 prodotto dalla Atmel

DIFFERENZE TRA MICROCONTROLLORE E MICROPROCESSORE

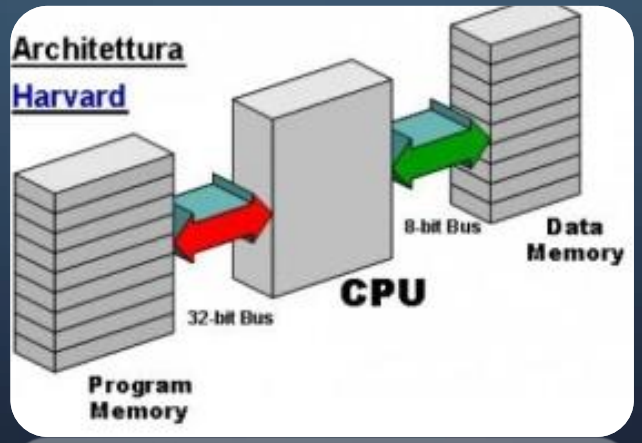
Microcontrollori (MCU)	Microprocessori
Si basano su un'architettura Harvard con due memorie e due bus distinti, uno per i programmi e uno per i dati	Si basano su un'architettura Von Neumann con una memoria unica in cui risiedono sia i programmi, sia i dati
Il programma risiede al suo interno in un'apposita «memoria programma» di tipo FLASH	Esegue un programma che risiede in una RAM esterna
È presente una RAM interna per i dati e una EEPROM per memorizzare dati che non devono essere persi in assenza di alimentazione (per esempio valori di taratura)	Ha una memoria interna limitata a pochi byte (i registri)
Sono implementate delle periferiche che consentono alla CPU di interfacciarsi con segnali esterni di tipo digitale e analogico	Si interfacciano tramite un bus parallelo con la memoria e le periferiche (le quali a loro volta si occupano di comunicare con l'esterno)
Sono integrati al suo interno amplificatori operazionali, convertitori ADC e DAC, comparatori, generatori di segnali PWM	Non sono integrate altre funzioni. Il suo compito è solo quello di svolgere operazioni matematiche molto velocemente
È sempre presente almeno un timer	È presente solo il clock della CPU
Hanno un set di istruzioni RISC (Reduced Instruction Set Computing)	Hanno un set di istruzioni CISC (Complex Instruction Set Computing)

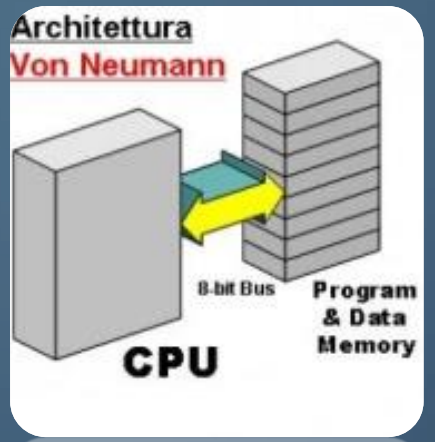
ARCHITETTURA DI VON NEUMANN VS. HARVARD

HARVARD

Sono presenti bus separati e memorie separate, per i dati e il programma

- il prelievo dell'istruzione (fetch) e dei dati, che servono all'esecuzione della stessa, non devono transitare nello stesso bus
- mentre si preleva già l'istruzione successiva (fetch) è possibile eseguire quella caricata precedentemente (execute) e insieme prelevare dati o scriverli, perchè i bus, essendo separati, non saranno congestionati
- aumenta la velocità di esecuzione riuscendo ad eseguire in media un'istruzione completa per ogni ciclo macchina





VON NEUMANN

I dati e il programma, risiedono nella stessa memoria

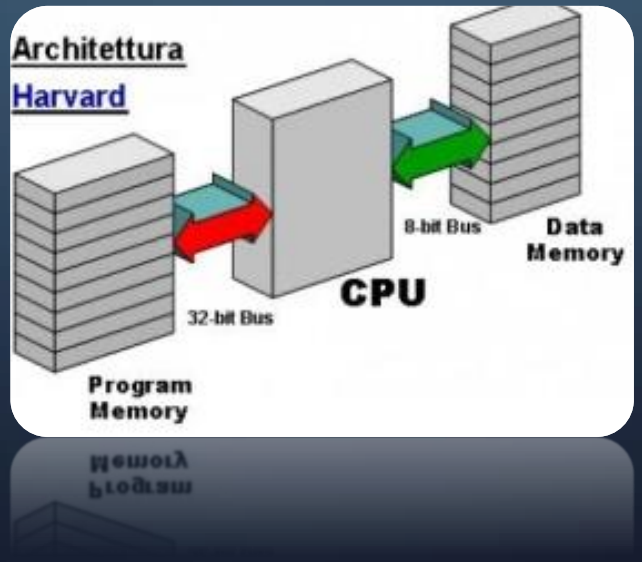
- il prelievo dell'istruzione (fetch) e dei dati, che servono all'esecuzione della stessa, devono transitare nello stesso bus
- ci sono quindi multipli accessi nella memoria per prelevare i dati o scriverli o per fare calcoli da parte della CPU
- su quel bus ci sarà un traffico molto sostenuto e questo inciderà sulla velocità di esecuzione delle istruzioni, che richiederanno vari cicli macchina per essere eseguite

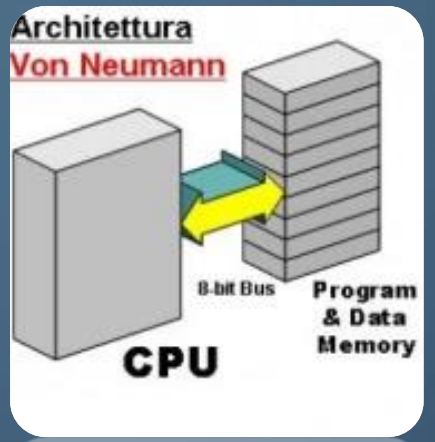
ARCHITETTURA DI VON NEUMANN VS. HARVARD

HARVARD

La progettazione dell'architettura è più complessa

- difficilmente si raggiunge la velocità di clock di 1 GHz
- è ottimizzato per acquisire e elaborare efficacemente sequenze di bit dalle periferiche
- non è importante la velocità di esecuzione del codice, quanto piuttosto la velocità di accesso alle periferiche



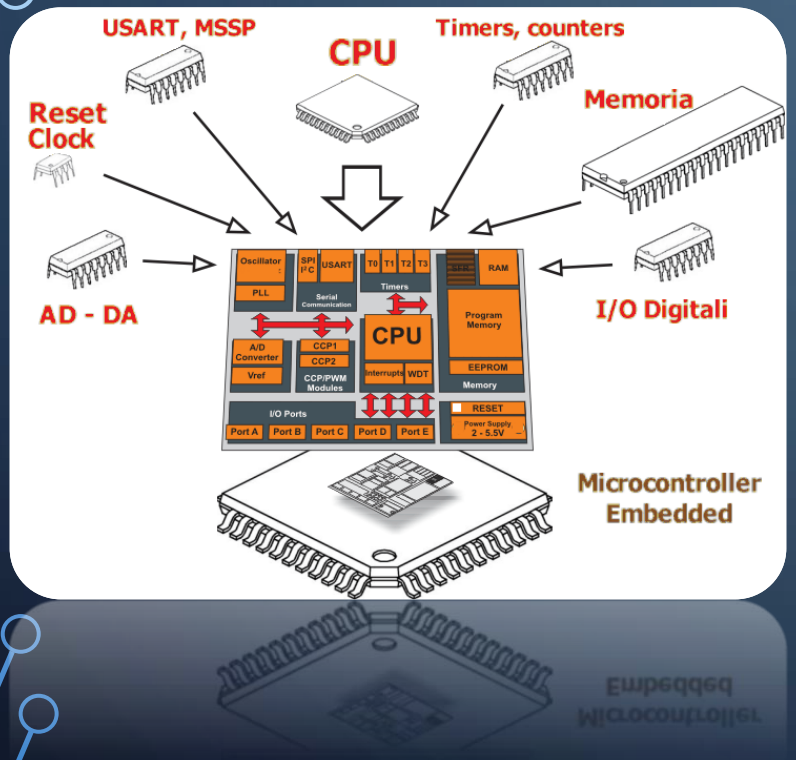


VON NEUMANN

La progettazione dell'architettura è più semplice

- è possibile raggiungere velocità di clock di vari GHz
- è ottimizzato per effettuare velocemente calcoli in virgola mobile
- è possibile gestire un numero elevato di istruzioni che coinvolgono il microprocessore (CPU) e le periferiche e svolgono operazioni complesse (CISC Complex Instruction Set Computing)

STRUTTURA DI UN MICROCONTROLLORE (MCU)



Un microcontrollore (chiamato anche MCU) è costituito da:

- una CPU (Central Process Unit) che controlla ed elabora i dati di una costellazione di periferiche. La CPU è costituita a sua volta da:
  - CU (Control Unit), che coordina l'esecuzione delle istruzioni
  - ALU (Arithmetic-Logic Unit), che esegue i calcoli aritmetici e logici
- delle unità di memoria utilizzate per memorizzare
  - istruzioni (memoria programma di tipo FLASH)
  - dati (memoria dati di tipo RAM e EEPROM)
- delle unità utilizzate per:
  - ricevere e trasmettere dati (I/O, UART, I2C, …)
  - convertire dati da analogico a digitale e viceversa (ADC, DC)
  - eseguire temporizzazioni (Timer)
  - gestire interruzioni
- un bus di sistema, che permette il trasferimento di dati tra le varie componenti

MEMORIA FLASH

- È una memoria non volatile (se si toglie l'alimentazione i dati non vengono persi)
- Offre la programmazione in-circuit usando un solo transistor per cella raggiungendo alte densità di integrazione, equivalenti alla densità di una DRAM (ram dinamica)
- Permette di essere letta un numero praticamente infinito di volte, ma riscritta solo fino circa 10.000 volte
  - le operazioni di riscrittura comportano la cancellazione di blocchi di memoria di grandezza prefissata, quindi in fase di riscrittura è probabile che dei dati, inutilmente, vengano cancellati e riscritti uguali degradando la durata della memoria
- Viene utilizzata per contenere il codice del programma dell'MCU che in genere non subisce molte riscritture

MEMORIA EEPROM

- EEPROM sta per Electrically Erasable Programmable Read Only Memory
- È una memoria non volatile come la FLASH
- Gli MCU in genere non ne hanno una grossa quantità, infatti è meno integrabile della FLASH
- Viene usata per memorizzare in modo non volatile delle variabili perché in genere supporta più di cicli di lettura e scrittura rispetto alla FLASH
  - è possibile cancellare e riscrivere ogni singolo byte
  - il numero totale delle riscritture è comunque limitato per cui è consigliabile valutare l'ordine di grandezza del numero dei cicli di lettura e scrittura a cui sarà sottoposta nel tempo

MEMORIA RAM

- RAM sta per Random Access Memory
- È una memoria volatile (se si toglie l'alimentazione i dati vengono cancellati)
- È tipicamente di tipo STATICO (SRAM) e si differenzia dalla RAM DINAMICA per essere più veloce
- La limitazione è che la SRAM è poco integrabile per cui occupa molto spazio
- Non ha alcun limite sul numero totale di letture e scritture

PERIFERICHE

Una MCU viene progettata per poter essere impiegato in più campi possibili; di qui la necessità di equipaggiarli con diversi tipi di dispositivi chiamati periferiche

Alcuni di questi sono:

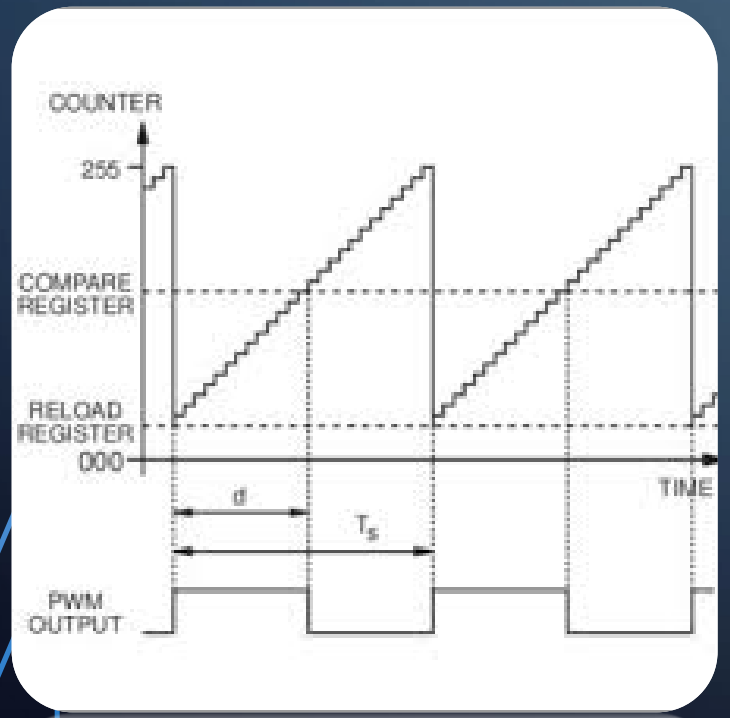
- **UART** (Universal Asynchronous Receiver Transmitter) è un adattatore di porte seriali per trasmissioni seriali asincrone
- **USART** (Universal Synchronous/Asynchronous Receiver Transmitter) è un adattatore di porte seriali per trasmissioni seriali sincrone ed asincrone selezionabili via software caratterizzate da una velocità superiore rispetto alla UART
- **I^2C bus** (Inter-Integrated Circuit bus, introdotto da Philips) è una semplice interfaccia seriale a 2 fili sviluppata per applicazioni ad 8 bit e largamente usata per interfacciare dispositivi esterni agli MCU
- **CAN** (Controller Area Network) è un protocollo di comunicazione seriale sviluppato dalla Bosch e dalla Intel e nato per interconnettere i vari dispositivi nelle autovetture
- **PWM** (Pulse Width Modulator) è un generatore di impulsi a frequenza costante con duty-cycle variabile
- **ADC** (Analog to Digital Converter) che hanno caratteristiche diversi nei vari MCU, distinguendosi soprattutto per l'accuratezza della conversione, ossia per il massimo errore di quantizzazione possibile, che è diretta conseguenza del numero di bit usati nella rappresentazione della conversione eseguita
- **DAC** (Digital to Analog Converter) che fanno l'operazione opposta degli ADC
- **I/O** veri e propri, la CPU legge i livelli logici sui piedini e li trasforma in parole binarie (input digitale) o viceversa (output digitale)

TIMER

La temporizzazione in una MCU è fondamentale poiché le varie operazioni devono essere fatte in momenti ben precisi

- I timer sono dispositivi che servono a fornire riferimenti temporali
- Possono essere utilizzati anche per contare degli impulsi in ingresso (utilizzo come COUNTER)
- Possono misurare frequenze esterne, o intervalli di tempo
  - l'operazione avviene copiando il valore di un timer interno autonomo in un registro ogni volta che occorre un evento esterno
- Possono essere utilizzati in coppia per generare segnali PWM

Pulse Width Modulation (PWM)



PROGRAMMAZIONE ASSEBLY

- Un programma è costituito da una sequenza di istruzioni, ognuna delle quali identifica univocamente una funzione che l'MCU deve svolgere
- Ogni istruzione è rappresentata da un codice operativo composto da un certo numero di bit, che varia in base allo specifico MCU, e memorizzata in una locazione di memoria dell'area programma
- Tali codici operativi sono gli unici che l'MCU è in grado di interpretare, ma essendo privi di significato per l'essere umano si è associata una sigla ad ogni istruzione in modo da ricordarne l'azione svolta
- Non solo alle istruzioni, ma anche alla dichiarazione di variabili, costanti ed etichette (label) sono associate delle sigle
- L'insieme di tali sigle nonché delle relative regole di utilizzo viene definito **Linguaggio Assemby** dell'MCU

PROGRAMMAZIONE AD ALTO LIVELLO

- L'assembly ha la caratteristica di essere basato sull'hardware della MCU e quindi ne sfrutta al massimo le potenzialità
- Lo svantaggio è che risulta complicato da utilizzare nella programmazione
- Le MCU moderne sono sufficientemente potenti da poter eseguire velocemente anche codice non ottimizzato, come quello che si ottiene a partire da programmi scritti in un linguaggio di programmazione al alto livello come C, Python, Java, ecc …
  - La differenza di velocità, sempre a favore dell'Assembly, va comunque dalle 100 alle 1000 volte più veloce
- Il vantaggio dell'utilizzo dei linguaggi ad alto livello sta nella velocità con cui si può scrivere del codice funzionante
  - L'operazione che richiede più tempo nella scrittura di un codice è l'eliminazione gli errori di semantica (cioè quando il codice fornisce un output incoerente o si comporta in un modo inaspettato, non previsto dal programmatore o in generale non desiderabile)