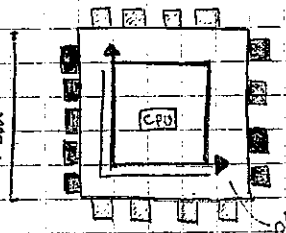
Il microcontrollore è un'architettura di computer = insieme di circuiti intorno a un CPU

La maggior parte degli elettromedicali contiene uno o più microcontrollori per la gestione dell'interfaccia e delle altre funzioni base.

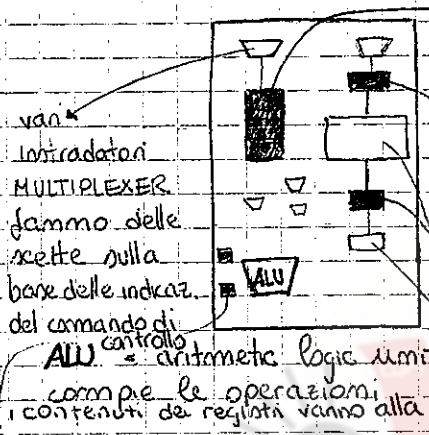


- **funzioni analogiche** (le grandezze analogiche sono quelle in cui l'informazione è contenuta in grandezze continue)
- **funzioni digitali** (le grandezze digitali sono quelle in cui l'informazione contenuta è discretizzata in n° finito di numeri)

ha bisogno di un orologio e di una sorgente di energia viene consumata energia al momento in cui cancelliamo dei dati, a livello teorico non al momento del calcolo.

nota: l'alfabeto è una grandezza digitale, il quadro di un pittore è una grandezza analogica

CPU = Central Processing Unit
è un processore, cioè un'intelligenza



memoria di lavoro: 32 contenitori da 8 caselle ognuno non molto estesa, 256 bit di memoria
ogni casella può memorizzare un valore logico (Vero e Falso)
i contenitori di memoria sono

- **PROGRAM COUNTER**
  contatore di programma: contiene l'indirizzo dell'istruzione da eseguire e la mette nell'instruction register. Sa incrementarsi, cioè sa contare.
- **INSTRUCTION REGISTER**
  instruction decode decodifica l'istruzione
  program memory: è la memoria da cui viene estratta l'istruzione per numero

ALU = arithmetic logic unit
compie le operazioni
i contenuti dei registri vanno alla ALU

STATUS REGISTER contiene informazioni che permettono di interpretare il risultato ottenuto che è dato in forma sintetica

La sequenza delle operazione non è rigida; il program counter possiede qualora in grado di alterare il normale flusso delle operazioni che è quello lineare
Il program counter possiede un multiplexer sopra.

ARCHITETTURA HARVARD (il CPU segue questo modello)
è stata la prima architettura proposta.
memoria dati → fornisce i dati alla ALU

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

memoria istruzioni

N.B. caratteristica è che le due memorie sono separate, cioè i dati sono separati dal programma.
il programma viene eseguito ma non può modificare se stesso → FIRMWARE
(esiste un'altra architettura in cui si usa la stessa scatola di memoria sia per i dati che per i programmi, ARCHITETTURA VON NEUHANN)
"von norman"

- doppio numero di fili per la comunicazione fra le parti
- doppia capacità di elaborazione perché dati e istruzioni viaggiano in parallelo

L'architettura Harvard è diventata vantaggiosa quando siamo riusciti a mettere memoria e processore nello stesso chip.

Nell'architettura von neuman il programma può modificare se stesso (svantaggio) ma maggiore semplicità

per prelevare i dati vi bisogno di sapere quale è istruzione.
pipeline: prelevo l'istruzione e i dati dell'istruzione precedente. mentre eseguo un'operazione, prelevo l'istruzione seguente, e quale può essere previsto perché in genere l'istruzione seguente è quella successiva.

TIPI DI ARCHITETTURE

- **RISC** macchina con set di istruzioni ridotto → tutte le istruzioni vengono eseguite dal processore con un unico ciclo macchina
- **CISC** " " " " " → complesso

La tendenza fu quella di costruire macchine che eseguissero istruzioni sempre più complesse. Per far ciò gli è stato detto di spezzarle in più cicli. Ci si accorse che il software sviluppato era di alto livello ma che i compilatori preferivano tradurre i programmi in sequenze di istruzioni semplici condivisi da più processori possibile. La presenza di istruzioni complesse mai utilizzate rallentava tutto.

microcontrollore
- CPU
- contenitore dati RAM
- contenitore programmi ROM
- periferiche I/O

periferiche = sono circuiti specializzati nella risoluzione di particolari problemi. comunicano con il processore principale, cioè il microprocessore. Esso trasmette dati alle periferiche e li riceve.

TIPI DI PERIFERICHE

- **Pin di I/O** → è il principale tipo di periferica. la periferica del circuito ha dei pin sui quali è presente un segnale digitale. I pin di I/O permettono di mediare tra il microcontrollore e i conduttori esterni.
- **timer e contatori** → conteggio di eventi, generazione di segnali periodici. il microcontrollore, grazie a queste periferiche di conteggio, sa generare onde rettangolari note.
- **convertitori A/D** → traduce una grandezza analogica in numero. cioè è un'oggetto che esegue una misura.
- **convertitori D/A** → traduce un numero in una grandezza analogica che è proporzionale al numero stesso.
- **interfaccia di comunicazione standard** (come la USB) → necessarie per la comunicazione tra oggetti digitali

COME DIALOGANO PROCESSORE e PERIFERICHE

Le periferiche vengono viste dal processore come contenitori di memoria e dialogano con il processore con scambi di messaggi su registri, cioè in memoria.

Oltre alle periferiche ci sono anche altri **circuiti ausiliari** (inizialmente erano esterni, poi sono stati inseriti nel cip):

- **generatore del clock** → da un tempo alla macchina, ad ogni segnale viene eseguita una operazione
- **supervisore dell'alimentazione** → l'alimentazione deve essere compresa in un certo range e per questo è necessario un supervisore.

Esempio del pacemaker;

- **circuito di reset** → riporta il microcontrollore allo stato iniziale. è collegato al supervisore.
- **supervisore della modalità operativa** → sceglie quali periferiche accendere per consumare meno energia possibile. (normale, sleep)
- **watchdog timer** → è un circuito ausiliario indipendente dal processore che conta con un meccanismo indipendente dal processore principale. Watchdog timer e processore possono resetarsi a vicenda. Arrivato a un massimo, a tal punto resetta il contatore principale, se sono stati visti dei problemi. Evita che il programma resti bloccato in una situazione non prevista.

INTRODUZIONE AL LINGUAGGIO ASSEMBLY

creato con C++

L'assembly del programma è in linguaggio binario.
L'assembly è subito eseguibile, non ha bisogno di essere compilato.
Il linguaggio assembly ci permette di decifrare il linguaggio binario.
Ogni combinazione binaria ha un corrispondente in linguaggio assembly.
Il tempo di esecuzione del programma è strettamente legato alla velocità della macchina scelta, perché nasce per quella macchina e verrà sempre usato su essa.
Questo è un punto debole perché non è versatile.
Ogni riga dell'assembly è un comando per il microcontrollore.
Le righe che iniziano con un punto si chiamano direttive e sono dirette all'assemblatore.
Esempi
- .org 100 avvisa l'assemblatore che deve partire dall'indirizzo 100

Le etichette sono seguite dai due punti e indicano un indirizzo di memoria

M1 ha indirizzo 106 e M2 ha indirizzo 108

I verbi inglesi designano delle istruzioni; ancora a destra c'è l'oggetto su cui l'istruzione opera

L'assemblatore converte il linguaggio assembly in linguaggio macchina.
Ogni CPU o famiglia di CPU ha un suo proprio assembly.
Im genere l'assembly si usa in combinazione con altri linguaggi di alto livello.

nota ROM sta per read only memory. Contiene i programmi che possono essere estratti.
RAM sta per random access memory, possiamo estrarre dati o salvare nuovi dati.

Il programma scritto con l'Assembly viene scritto da un programmatore e quello che vediamo nella slide è come appare a noi.

L'assembly → è un linguaggio imperativo, cioè da dei comandi (non descrittivo)

→ l'oggetto sul quale l'istruzione opera si chiama operando; un'istruzione può avere un numero variabile di operandi anche nessuno

→ label = etichetta, è un nome simbolico che fa riferimento a un indirizzo si ottiene iniziando una riga, estraendo dopo un nome corrisponde all'indirizzo che l'istruzione seguente avrà nella memoria di programma.

È importante quando non vogliamo eseguire una operazione dopo l'altra, alterazione dell'ordine.

I commenti non hanno impatto sull'assemblatore, servono solo all'uomo per leggere e studiare e sono essenziali sia se si lavora da soli sia se si lavora in team.
È necessario commentare bene il programma
sono introdotti da

;
//
/* ... */ (come in C)

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

C'é anche una cosa da poter fare:
posso scrivere un programma e inserire i commenti,
poi do questo in pasto all'assembly e ottengo il programma,
se invece do questo in pasto a un'altro programma mi esce la documentazione.
Tra i commenti avevo inserito delle keywords che servono per fa questo.

le righe che iniziano con un . danno delle indicazioni all'assemblatore, gli dicono come interpretare. Si chiamano DIRETTIVE.

Si possono usare dei flag che servono per fare un assemblaggio condizionato (ma non lo faremo).
Si può dire su quali microcontrollori le nostro programma può girare.

microcontrollore
- sezione di controllo → program counter, instruction register [...] status register.
  - raccoglie le istruzioni da una memoria di programma.
- sezione operativa → ALU
  - esegue le istruzioni

TIPOLOGIE DI MEMORIA → Nell'elaboratore elementare sono presenti diversi tipi di memoria, a cui sono associati gli elementi del linguaggio.

- memoria di PROGRAMMA
  le label sono un'indirizzo della memoria di programma.
  Nelle architetture di tipo Harvard la memoria di programma é un'oggetto a se stante.

- memoria dei DATI INTERNA
  - registri interni
    - registri di I/O
    - variabili logiche
  Ci lavora la sezione operativa
  fa parte del processore
  l'accesso é rapido
  é costituita da pochi registri di lavoro.
  la memoria dati interna contiene una parte dei registri input/output
  sono condivisi dal processore e dalle periferiche

  variabili logiche → sono particolari bit di registro I/O
  → servono a condizionare
  → contengono informazioni sull'esito di ogni istruzione

- memoria DATI ESTESA
  Contengono gran parte delle informazioni cui il microcontrollore ha accesso.
  Possono essere aggiunte anche memorie esterne.

  Ogni periferica ha un posto in questa memoria;
  da quando le periferiche sono diventate molte non aveva più senso porle vicino al processore.

RAPPRESENTAZIONE DELL'INFORMAZIONE → ad ogni tipo di memoria (dimensione n bit) possono essere attribuiti 2^n valori binari con diversi significati.

Le memorie sono oggetti costituiti da bit,
le informazioni sono insiemi di bit che possono essere interpretati se introduciamo delle regole di rappresentazione.

MASTER COPY
Tel. 050-8312126
Cell. 388-9837745

(bit) 1 0

Il programmatore decide la rappresentazione, cioè decide a che cosa corrisponde una determinata serie di bit
GPU = processore specializzato a ~~int~~ elaborare immagine

ISTRUZIONI → occupano 16 bit
↘ informazione complessa formata da più campi

BYTE → 8 bit
WORD → 16 bit
} → con segno → la macchina utilizza il complemento a due
→ senza segno

CARATTERE → esistono tabelle predefinite, la più comune è l'ASCII (American standard character 2!) (∧ carattere = 8 bit)
permettono di interpretare una determinata sequenza di bit come un carattere.
la macchina non presenta istruzioni per elaborare caratteri, ma solo per cercarli.

VALORI DI INDIRIZZO → 16 bit in genere
hanno valori di indirizzo della memoria di programma o della memoria dati esterna.

2^{20} = mega
2^{32} = giga
2^{10} = kilo

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

VALORI DI TIPO SALTO → è un pezzo di istruzione
è un numero relativo espresso da un certo numero di bit

FLAG → è un valore di tipo vero o falso al quale possiamo far riferimento

MEMORIA DI PROGRAMMA

① → non volatile → non si cancella quando si spegne il microprocessore
→ contiene in generale **valori di tipo istruzione**
→ contiene **in parole da n bit**
Ogni valore istruzione è rappresentato da un numero in di parole
? [in fa riferimento al numero massimo di byte, non al numero di istruzioni → per ottenerlo bisogna dividere per due
è giusto?
→ la struttura delle istruzioni può essere più complicata di 1 riga, 1 istruzione;
ci sono istruzioni che occupano più di una riga → ISTRUZIONI MULTIPAROLA
nella prima riga di ogni istruzione ci deve essere scritto da quante parole è formata tale istruzione.

deprogettazione: siamo in possesso della serie di numeri e vogliamo capire come è fatto il programma.
Abbiamo un pezzo di codice, il frammento parte dall'indirizzo 6 ma non è detto che quello sia l'inizio di un'istruzione, potrebbe essere qualcosa di già iniziato, come un'istruzione da