Elettronica Digitale

Roberto Roncella — Dip. Ing. Informazione
Via Caruso 16 – primo piano (ufficio C1.2)
roberto.roncella@iet.unipi.it — 050 2217669
roncella.iet.unipi.it

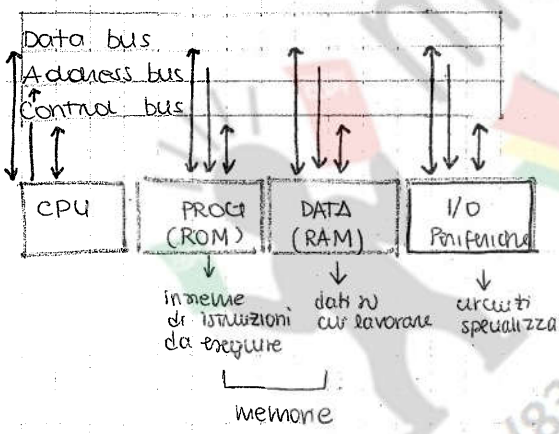
RICEVIMENTO → Giovedì 15-18

ESAME
- SISTEMI DIGITALI → scritto + orale . 5 es. in 1 ora, minimo 17,5 per orale. Stessa sessione anche appelli diversi per orale
- ELETTRONICA ANALOGICA → scritto + orale

voto modulo = (2·voto orale + voto scritto)/3
voto finale = media voti moduli

MATERIALE : slide sito
reti ← D.D. Givone, Digital Principles and Design, Mc Graw Hill, 1 ed 2003 (cap 3-8)
Elettr. dig ← stesso libro, appendice A
J Millmann, A. Gabriel, P. Terreni, Elettronica di Millmann, Mc Graw Hill, Italia, 4 ed 2005

MICROCONTROLLORE



È un insieme di vari oggetti collegati nello stesso dispositivo fisico tra cui la CPU (microprocessore o central processing unit). Il microcontrollore racchiude tutto in un unico chip.

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

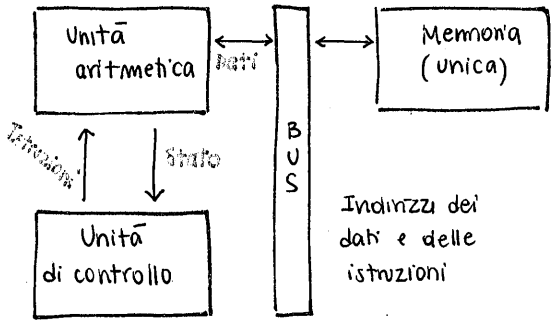
COS'È UNA CPU? È il "cervello" del microcontrollore.

vedi figura pag seguente

- **Gestisce la ALU** (unità logico-aritmetica) con l'uso di piccole memorie veloci, i REGISTRI, che contengono gli operandi usati dalla ALU.
- Contiene un **interfaccia con le memorie**, più importanti
- si preleva dalla memoria un'istruzione alla volta, e viene decodificata; si recuperano gli operandi e poi la si esegue a ritmo di clock scegliendo la collocazione del risultato

Si alternano la fase di **FETCH** (prelievo dell'istruzione) e di **EXECUTE** (esecuzione): ci sono vari tipi di istruzioni come quelle di controllo che permettono di indirizzare il prelievo dell'istruzione successiva.

ARCHITETTURA VON NEUMANN

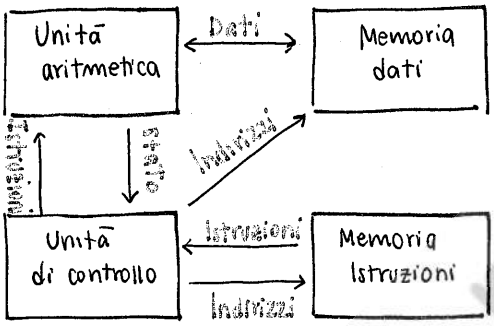


è una macchina di ridotta complessità in cui nell'unica memoria vi sono sia dati sia istruzioni separati

Punti di forza → • è presente **solo un bus** (dati + indirizzi) che collega il processore verso l'esterno
• viene usato sia per le istruzioni che per i dati da elaborare → si **risparmiano connessioni**

Punti di debolezza → • unica memoria con rischio di sovrapposizione tra dati e istruzioni (virus, bug durante un)
• unico bus e dunque rischio di creare un **collo di bottiglia** dell'architettura perché non si possono leggere contemporaneamente dati e istruzioni.

ARCHITETTURA HARVARD

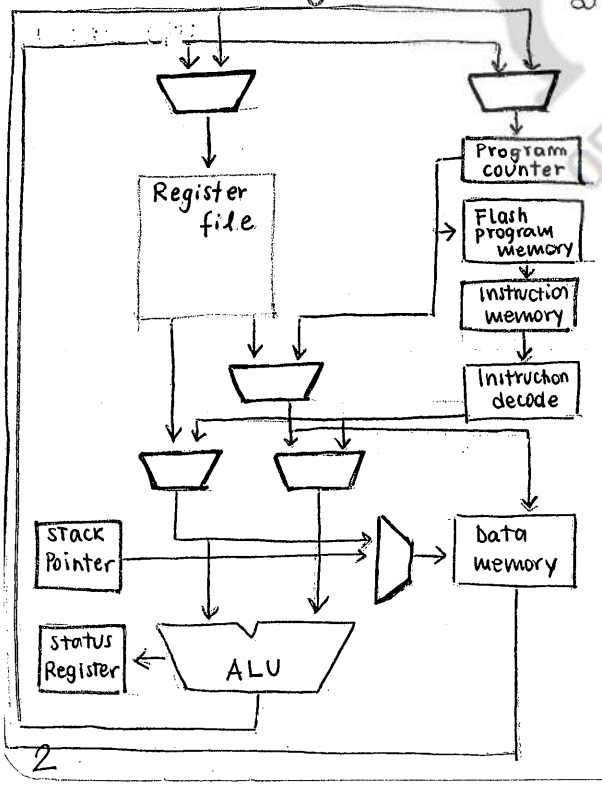


Inizialmente scartata rispetto alla macchina di von Neumann fu poi ripresa negli anni '80 per elaborare flussi di informazione in tempo reale (creare processori che elaborano e forniscono risultati mentre in contemporanea arrivano i dati).
È l'architettura alla quale si ispirano i microprocessori.

MEMORIA DATI
- accesso diretto alla ALU
- realizzazione con la memoria RAM

MEMORIA ISTRUZIONI
- il contenuto rimane invariato
- può essere realizzata con memoria **non volatile** quindi la memorizzazione del programma è permanente
- la parte di programma contenuta in ROM è definita "**firmware**" ovvero programmi intrinsechi nella macchina

Caratteristiche
- non presenta colli di bottiglia → vi sono infatti **memorie separate**
- **maggiori prestazioni**: numero di operazioni al secondo delle MIPS (i moderni digital signal processors arrivano a centinaia di GIPS)
- migliorabile aggiungendo altri bus dati o frazionando le fasi di fetch e di esecuzione



→ REGISTER FILE = registri (32)
→ ALU = unità logico aritmetica. Il suo risultato viene scritto nei registri
→ TRAPEZI NERI = multiplexer, instradano l'ingresso scelto sull'unica uscita
→ INSTRUCTION REGISTER = dove si appoggia l'istruzione dopo il prelievo

PROGRAM COUNTER = "contatore di programma" = a ogni clock conta 0, 1, 2, 3... e il suo contenuto viene usato per scegliere l'istruzione. Contiene l'indirizzo dell'istruzione da eseguire e ci garantisce lo svolgimento ordinato della sequenza di istruzioni. Parte da 0 all'accensione e forse è la componente più importante della CPU.
STATUS REGISTER = alcune istruzioni sono condizionate dalle istruzioni precedenti e serve qualcuno nella CPU che conservi una traccia di ciò che è successo prima → tramanda alle istruzioni successive lo stato corrente
STACK POINTER = contenitore che punta a un indirizzo nella memoria dati

MASTER COPY Tel. 050 831216 Cell. 388 9837745

PERIFERICHE DI UN µC

- Sono sottocircuiti dedicati ad **applicazioni specifiche**, hanno un hardware specifico e programmabile per risolvere ogni problema di quella categoria. Si parla di flessibilità nella programmazione e ciò permette di sollevare il processore da compiti specifici.
- Sono studiate in modo da **dialogare facilmente con la CPU**: all'inizio si pensava di creare istruzioni specifiche per la CPU, ora invece i circuiti periferici sono visti dal µC come pezzi di memoria, sono configurate dal µP che li controlla direttamente ricevendo e trasmettendo info, scambiando dati. La configurazione delle periferiche avviene in opportuni indirizzi e infatti si dice che le periferiche sono "mappate in memoria".
- Le famiglie di µC sono chip con lo stesso µP e diverse periferiche.

MASTER COPY
Tel. 050 831 21 26
Cell. 366 ...

TIPOLOGIE DI PERIFERICHE

PIN DI I/O → lo scambio di informazione digitale avviene su un singolo PIN, l'informazione può essere bidirezionale. Intenzione sul livello o sulla variazione.

TIMER e CONTATORI →
- conteggio di eventi: misurazioni di tempo e frequenza
- generazione di segnali periodici: onde rettangolari note o segnali con ciclo di lavoro variabile (PWM). → PULSE WIDTH MODULATION

(Es. Contare quante volte passa la tensione bassa al secondo: 100 in Italia, 120 in Usa.)



$\delta = T_1/T$ = "CICLO di Lavoro" = rapporto tra tempo stato alto e periodo

Modulando la lunghezza dell'impulso si varia la natura del segnale (Es luci di natale)

CONVERTITORE A/D → leggono la tensione presente su un PIN e la traducono in un valore digitale in un registro. Nel caso più semplice fornisce un bit (comparatore).

CONVERTITORE D/A → generano una tensione analogica su un PIN di uscita corrispondente al valore di un registro.

INTERFACCE DI COMUNICAZIONE STANDARD → RS232, CAN, USB = protocolli = fisicali' ma standard
- CAN → nelle centraline veicolari
- USB → procedure per dialogare

CIRCUITI AUSILIARI = presupposti per garantire un buon funzionamento della CPU.

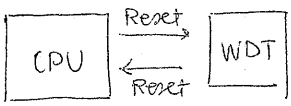
GENERATORE DEL CLOCK = la CPU necessita di un segnale elettronico che dia il sincronismo alla macchina, e sarà un oscillatore fatto apposta per produrre il clock (max 32 Mega Hz).
C'è un compromesso tra velocità di elaborazione (frequenza del clock) e potenza dissipata.
La progettazione del clock è una strategia per minimizzare il dispendio di energia.

SUPERVISORE DELL'ALIMENTAZIONE = qualcuno che fornisce l'energia alla CPU per lavorare e che impedisca il funzionamento della macchina quando la tensione scende sotto un certo livello.

CIRCUITO DI RESET = riporta la macchina alle condizioni iniziali, fa ripartire il clock, e azzera il contatore di programma.

SUPERVISORE DELLA MODALITÀ OPERATIVA = NORMALE / SLEEP = per ottimizzare il consumo di potenza (2 livelli di stand-by)

WATCH DOG TIMER = circuito autonomo rispetto alla CPU con un proprio clock, e alimentazione a comune. WDT e CPU sono collegati l'uno col reset dell'altro in modo che, se il meccanismo è attivato, quando si arriva alla massima capacità di conteggio, il WDT azzera e fa ripartire la CPU e viceversa la CPU può azzerare il conteggio impedendogli di arrivare in fondo e quindi causare a sua volta un azzeramento = deve essere previsto dal programmatore.



Se il programma dimentica di azzerare il WDT in tempo, questo quando arriva in fondo al conteggio azzera la CPU
È un meccanismo che si trova nei PACEMAKER (se il battito è normale allora il PM non è attivo altrimenti entra in funzione).

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

LINGUAGGIO ASSEMBLY

Strumento di programmazione più vicino alla realtà fisica-elettronica di un elaboratore.
→ gli elementi del linguaggio corrispondono agli elementi dell'architettura del processore.
→ i programmi assembly sono codificati in modo diretto (tradotti in binari) e quindi **non vanno compilati** e sono inseriti nella memoria di programma dell'elaboratore per essere eseguiti in modo univoco
→ è possibile conoscere la relazione tra programma assembly (contando le istruzioni) e tempo di esecuzione della macchina (non ci è nel linguaggio ad alto livello come C++).
→ ogni macchina o famiglia di macchine ha il suo linguaggio assembly.

Nel linguaggio macchina l'elemento base che definisce l'ISTRUZIONE è il **CODICE MNEMONICO** che dirà alla macchina cosa fare e gli **OPERANDI** (contenitori sui quali deve essere eseguita l'istruzione). che sono variabili / costanti di tipo opportuno

Il linguaggio macchina verrà tradotto in binario e memorizzato nella memoria di programma Di alcune istruzioni voglio conoscere l'indirizzo: quindi inserisco un segnaposto che consiste in una etichetta o **LABEL** che infatti indica l'indirizzo dell'istruzione a fianco (in modo da non tenerne di conto tutti) nella memoria di programma e sono concluse dai due punti.

La prima riga del programma assembler è diversa dalle altre, appare un punto e una parola chiave Questa è una **DIRETTIVA**, ovvero un pezzo del documento del programma non destinato a essere tradotto in codice ed è un comando rivolto all'assemblatore. in modo da tradurre l'assembly in codice. (sono introdotte dal punto).
Es. ". org 100" → org = origine → significa che le istruzioni vanno messe dall'indirizzo 100 in poi.
". equ pippo = 8" → permette di definire una costante (pippo) che varrà sempre 8.
". define" → definisce una variabile

Nel linguaggio Assembly, a differenza del C++, si richiede una maggiore rigidità nella scrittura del codice. Si usa una riga per istruzione e i **commenti** sono elementi accessori fondamentali per comprendere il funzionamento del programma e la sua manutenibilità. sono preceduti dal punto e virgola e durano fino al terminatore di riga. Nelle ultime versioni si usa anche // o /*... */ come in C++.

TIPOLOGIE DI MEMORIA

Nel processore sono presenti diversi tipi di memoria a cui sono associati gli elementi del linguaggio

- **Memoria di Programma** = non volatile, contenente le istruzioni del programma. in m parole da n bit. Ogni locazione è individuata da un indirizzo di $\log_2(m)$ bit

fascia media bassa ↑

Il microcontrollore XMEGA256A3BU definisce la dimensione in KBi della memoria di programma (256).

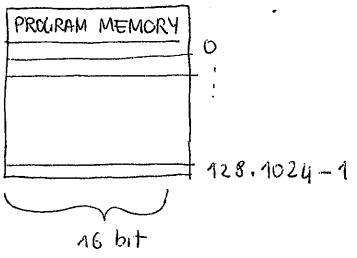
↓ "Kilo-byte" con la "i" finale perché $K = 2^{10} \neq 10^3$ a distinzione di 10^3.

Esistono istruzioni con più operandi, ma il 1° individua il genere il risultato in senso di destinazione.

Il microcontrollore nasce per gestire problemi reali, come la lettura di sensori, in genere operazioni di controllo che si ripetono. ed è l'applicazione a stabilire quale deve essere la memoria di programma.

La mem. di programma è organizzata in un contenitore di 16 bit

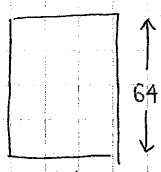
$128 = 2^7$
$1024 = 2^{10}$
] servono 17 cifre binarie per creare 2^{17} diverse combinazioni.



Il numero 17 ci dice quanto deve essere grande il CONTATORE di programma che in tal caso deve contenere 17 bit da 0 a 16
4 (la Atmel lo ha costruito con 22 bit)

- **Memoria dati interna**
  - Registri interni = si tratta di 32 registri da 8 bit ciascuno, vicino alla ALU. Vanno quindi da R0 a R31. Sono i più importanti dal punto di vista delle istruzioni.
  - Registri di I/O
  - Variabili logiche : contengono info sull esito di ogni istruzione

- **Memoria dei dati estesa** = contenitore che presenta 64 kb celle da 8 bit ciascuna.



→ $64 = 2^6$, $k = 2^{10}$] 2^{16} indirizzi => gli indirizzi contenuti hanno ciascuno dimensione pari a 16 bit.

Nella macchina non ci sono fisicamente tutti i 64 kb:
→ I primi 4 kb sono dedicati alla gestione delle periferiche → all interno vi sono i **registri di stato** che contengono le info prodotte dalla ALU.
→ Altri 4 kb sono di memoria non volatile
→ Dall indirizzo 8000 al 26000 c'è memoria fisica vera leggibile e scrivibile.
→ Oltre non c'è niente

RAPPRESENTAZIONE DELL'INFORMAZIONE

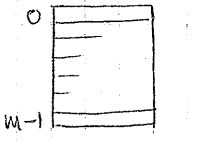
A ogni tipo di memoria (dimensione n bit) possono essere attribuiti valori binari (2^n) con diversi significati.
Ci sono tanti modi di usare i bit per rappresentare l'informazione

- Valori istruzione
- Valori byte (8 b) o word (16 bit) senza segno
- Valori byte o word con segno
- Valori carattere (ASCII)
- Valori indirizzo (22 bit nella memoria di programma) < nella memoria di programma / nella memoria estesa
- Valori di salto (displacement)
- Valori flag (0 o 1)

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

MEMORIA DI PROGRAMMA

Contiene in generale valori di tipo istruzione codificate su **16 bit** in binario (possono anche arrivare su 16, 32, 48 bit).
L'istruzione ha dimensione di m parole da n bit => ogni valore istruzione sarà rappresentato da 1, 2 o più parole da n bit con $n = \log_2(m)$. che è l'indirizzo



Quando un programma in linguaggio macchina, il codice ci dà informazioni sulle prestazioni dell istruzione.
Tipicamente i bit che definiscono l'istruzione ci dicono:
→ quale operazione si esegue = si chiama infatti **CODICE OPERATIVO**
→ gli **operandi** usati nell operazione (gli operandi nelle vengono introdotti tramite il multiplexer).