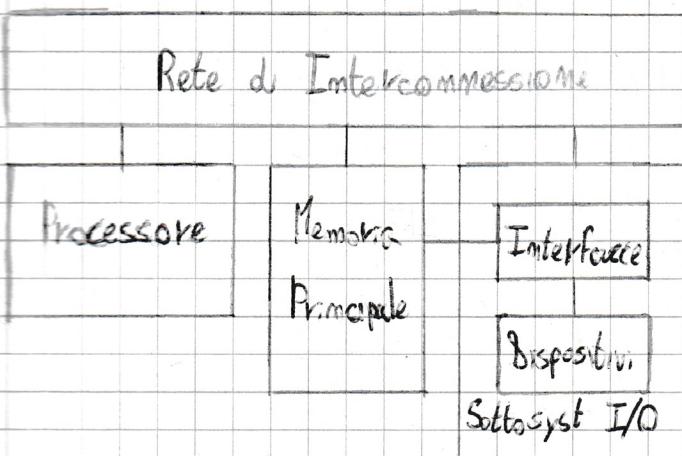


Architettura del "semplice" calcolatore visto a lezione SEP8

Imisiamo a mettere insieme tutte le fonti dei mostri incubi.



Calcolatore "semplice" visto a lezione.

In breve il funzionamento dei blocchi

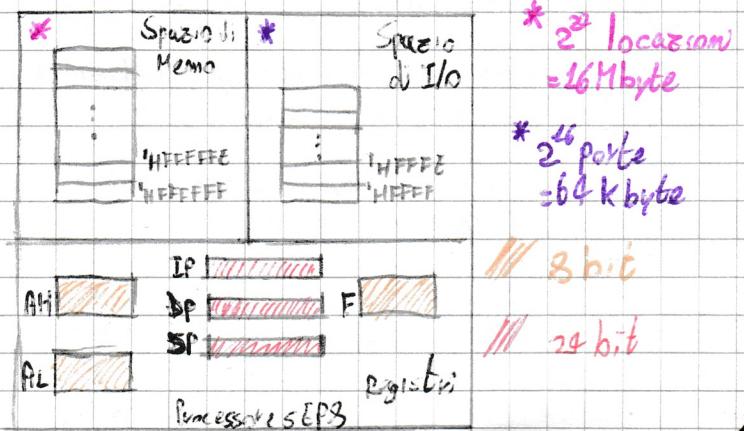
Sottosyst I/O: riceve / trasmette col mondo esterno delle informazioni sotto varie forme
Al suo interno si trovano interfacce e dispositivi. Le prime gestiscono i secondi (per permettere un collegamento standard tra processore e disp.)
i dispositivi si occupano della vera e propria codifica.

Mem principale: ad ogni istante contiene istruzioni e dati che il processore elabora.
Parte di essa è dedicata a memoria video

Processore: ciclicamente preleva ed esegue un'istruzione della memoria. Queste istruzioni solitamente sono salvate in modo sequenziale, ma possono capitare dei salti. Al reset è dotato di un indirizzo ben preciso dove è presente in modo indelebile (sv EEPROM) il programma di bootstrap

Rete di interconnessione: è il bus che collega vari moduli tra loro

Con il processore che abbiamo scelto di studiare possiamo a vedere cosa vede il programmatore (vogliere fa assembly)



Vediamo i registri presenti:

AH, AL sono registri accumulatori, usati per memorizzare gli operandi; sono a 8 bit

F registro dei flag, degli 8 bit ce ne interessano i 4 meno significativi perché contengono i 4 flag (CF, ZF, SF, OF)

IP, DP, SP sono registri puntatori (a 24 bit), in particolare:

IP → Contiene l'indirizzo della prossima istruzione da eseguire

SP → Contiene l'indirizzo del top della pila (modalità LIFO)

DP → Contiene l'indirizzo di memoria che contiene un operando

Focus Reset iniziale

Di questi registri appena visti, al reset vengono settati solo **IP** a **HFF000** e **F** a **H00** si trova il bootstrapping

Istruzioni assembler che il processore ~~non~~ comprende

La struttura delle istruzioni è uguale a quella dell'assembler che abbiamo già studiato (anche le operazioni possibili)

Tutte queste istruzioni possono essere divise in 8 formati (il formato viene indicato dai 3 bit più significativi dell'opcode).

Ogni formato ha la propria fase di fetch

Formato Cosa è

Fase Fetch

F0 Operazione senza operanti o dove tutti gli operandi sono registri 8bit

Fetch F2_0: $A_{23..AO} \leftarrow DP$; $IP \leftarrow$ FetchEnd
 $STAC \leftarrow READ_B$
... : SOURCE $\leftarrow APPO$ STAN \leftarrow FetchEnd

Sorgente puntato da %DP, destinatario è un registro 8bit

F2 Sorgente è un registro, destinatario è puntato da %S 8bit

Fetch F3_0: DEST_APPO $\leftarrow DP$ STAN \leftarrow FetchB

Sorgente è un registro, destinatario è puntato da %S 8bit

Fetch F4_0: $A_{23..AO} \leftarrow IP$; $IP \leftarrow IP+1$; $STAC \leftarrow READ_B$
... : SOURCE $\leftarrow APPO$

Sorgente è un immediato, destinatario è un registro 16bit

Fetch F4_0: $A_{23..AO} \leftarrow IP$; $IP \leftarrow IP+3$; $STAC \leftarrow READ_M$
 $STAB \leftarrow READ_B$ → SOURCE $\leftarrow APPO$
 $A_{23..AO} \leftarrow EAPPO_2, EAPPO_1, EAPPO_0$, ...;
 $A_{23..AO} \leftarrow IP$; $IP \leftarrow IP+3$; $READ_M$
DEST_APPO $\leftarrow APPO_2, APPO_1, APPO_0$

F5 Sorgente è un registro, destinatario è un indirizzo diretto 32bit

F6

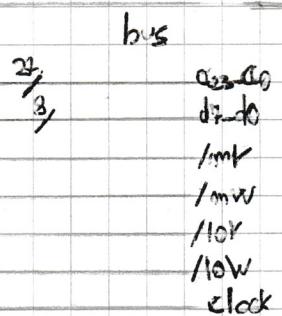
F.7 Tutte le istruzioni di salto con rebbit, va: indirizzo 32 bit
 sono le costanti operazioni, le grandezze sono varie

A23-A0<IP IPC<IP+3 \$DATA<RD/DM
 DEST/ADDR< S APP2, APP1, AD103,

Nulle

Collegamenti del bus

Dato che tutti i moduli passano dal bus, guardiamone i fili che ci passano



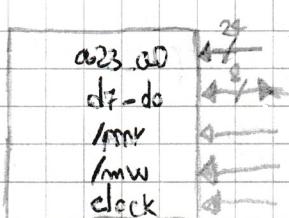
Fili A cosa serve

	Processore	Memoria	I/O
a23-a0	Trasmette l'indirizzo richiesto dal processore	✓ (24)	✓ (24) ✓ (16-sign)
d7-d0	Trasporta i dati presenti a quell'indirizzo	✓ (8)	✓ (8) ✓ (8)
imr	Indica se sta avvenendo una lettura in mem	✓	✓ X
imw	Indica se sta avvenendo una scrittura in mem	✓	✓ X
ior	Indica se sta avvenendo una lettura in I/O	✓	X ✓
iow	Indica se sta avvenendo una scrittura in I/O	✓	X ✓
clock	serve per sincronizzare tutti i moduli di clock legato ad un gen	✓	✓ ✓

Adesso che abbiamo visto quali fili sono vanno dove, andiamo a vedere i singoli moduli partendo dalla memoria

Sposto di memoria

Dall'esterno la struttura è:



Per come lo abbiamo studiato, tutto lo spazio è porzionato in 3 parti:

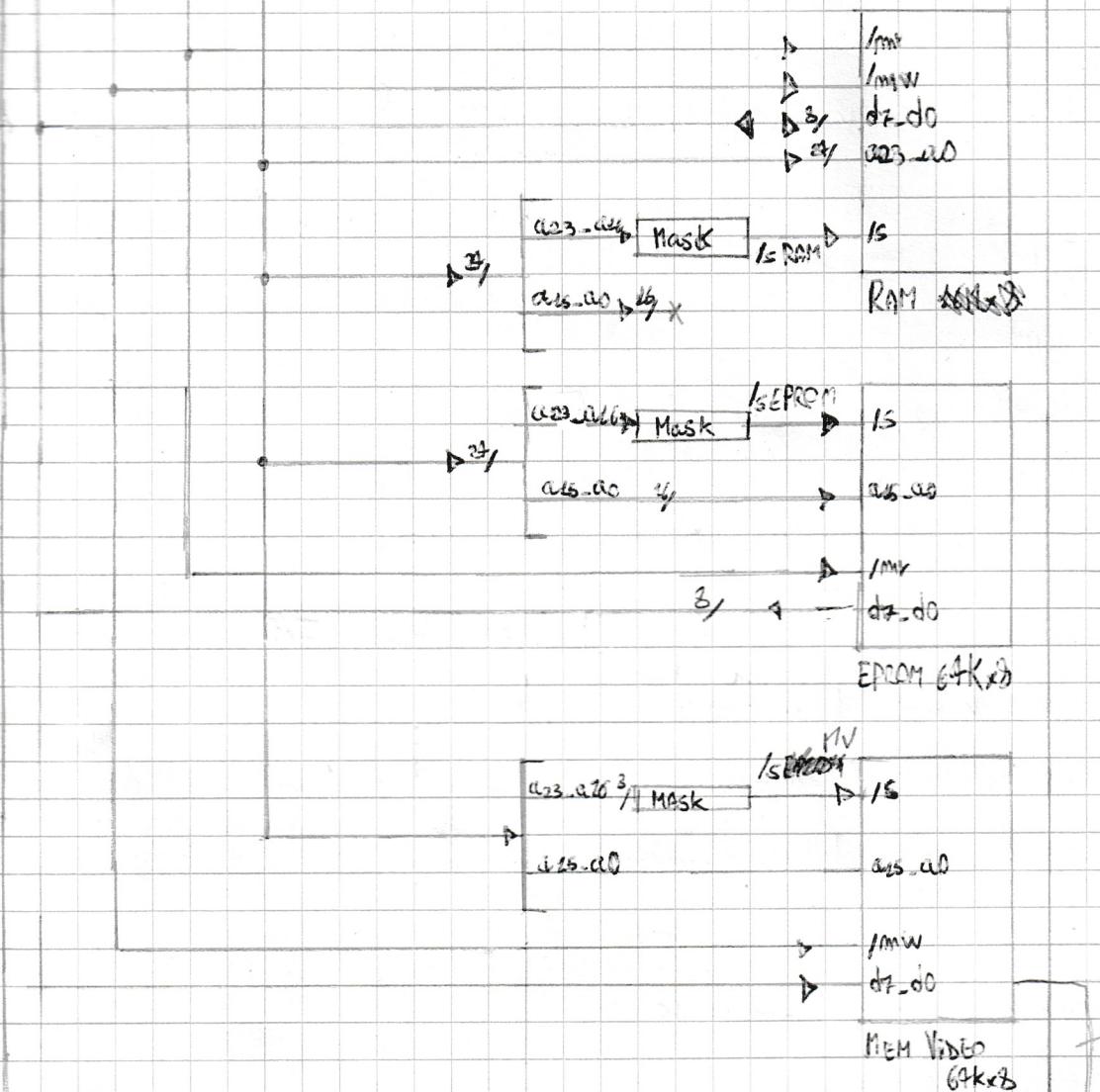
- porzione volatile (RAM) 'H000000 a 'H03FFFF e 'H0B0000 a 'HFFEFF

- porzione video 'H0A0000 a 'H0AFFFF

- porzione non volatile (EPROM) 'HFF0000 a 'HFFFFFF

Guadagnare i disegni completi

22.00
3/ 1MW 1K 023.00
△
▽ ▽ ▽ ▽



La mask mi serve per capire chi è selezionato

a23 a22 a21 a20 a19 a18 a17 a16 RAM 1Mbit 1EPROM

0 0 0 0 1 0 1 0 1 0 1

1 1 1 1 1 1 1 1 1 0

altro

0 1 1

Allo stesso modo andiamo a vedere lo spazio di I/O, in particolare noi vedremo le interfacce che possono essere di ingresso, di uscita o entrambe

Come prima iniziamo da fuori: le scatole (spazio I/O)

4) $\begin{array}{l} \text{a}_{25-20} \\ \text{l}_{10V} \\ \text{l}_{10V} \end{array}$

All'interno dello spazio di I/O possono esserci una o più interfacce che hanno una struttura simile a quelle già viste

3) $\begin{array}{l} \text{d}_{7-do} \\ \text{d}_{7-do} \end{array}$

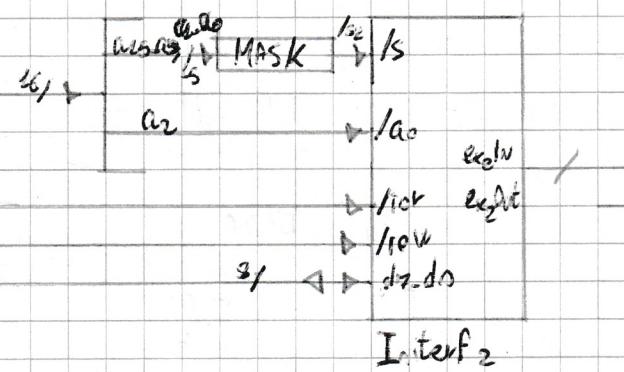
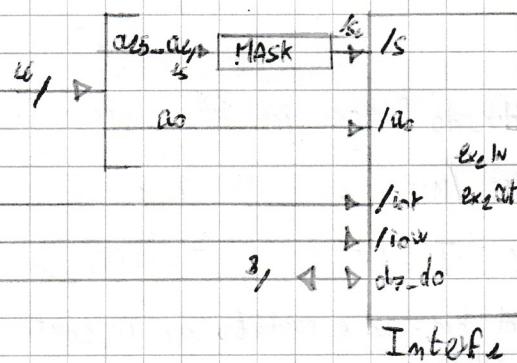
Le interfacce possono avere vari registri, questo filo serve per designarne molti (se +du uno)

Supponiamo che nel nostro spazio di IO ci siano 2 interfacce

in lettura e scrittura agli indirizzi H'05CB e H'03C9, H'0060 e H'0064

d_{7-do} /low h₁ a₂₅₋₂₀

$\begin{array}{c} \text{a}_1 \\ \downarrow \\ \text{a}_2 \\ \downarrow \\ \text{a}_3 \\ \downarrow \\ \text{a}_4 \end{array}$



Mask è come quella di prima per la RAM, con i giusti valori.

Il funzionamento dei vari valori d. ls, l_{or} (l_{mr}), l_{ow} (l_{mw})

è il seguente.

ls l_{or} (l_{mr}) l_{ow} (l_{mw}) Che succede

1 - - Non selezionata

Anche se il significato dei fili

è praticamente uguale

0 1 1 Nessuna azione

i cicli di lettura e scrittura

0 0 1 Ciclo di lettura

e le loro relative temporizzazioni

0 1 0 Ciclo di scrittura

sono differenti per più motivi

0 0 0 Non definito

che vediamo ora

Sulla ram si può leggere e scrivere ovunque, mentre nelle interfacce
è fatto avere entrambe in una (magari ci sono entrambi i fili,
ma se ne usa uno solo).

Guardando dalla parte dei dispositivi, essi hanno velocità diverse (e sono più lente del processore) e modalità di trasferimento diverse (seriali e parallele).

Per questo avere le interface in mezzo ai tuoi

Per capire i vari cicli è necessario vedere le fasi di esecuzione del processore ed il modulo processore stesso

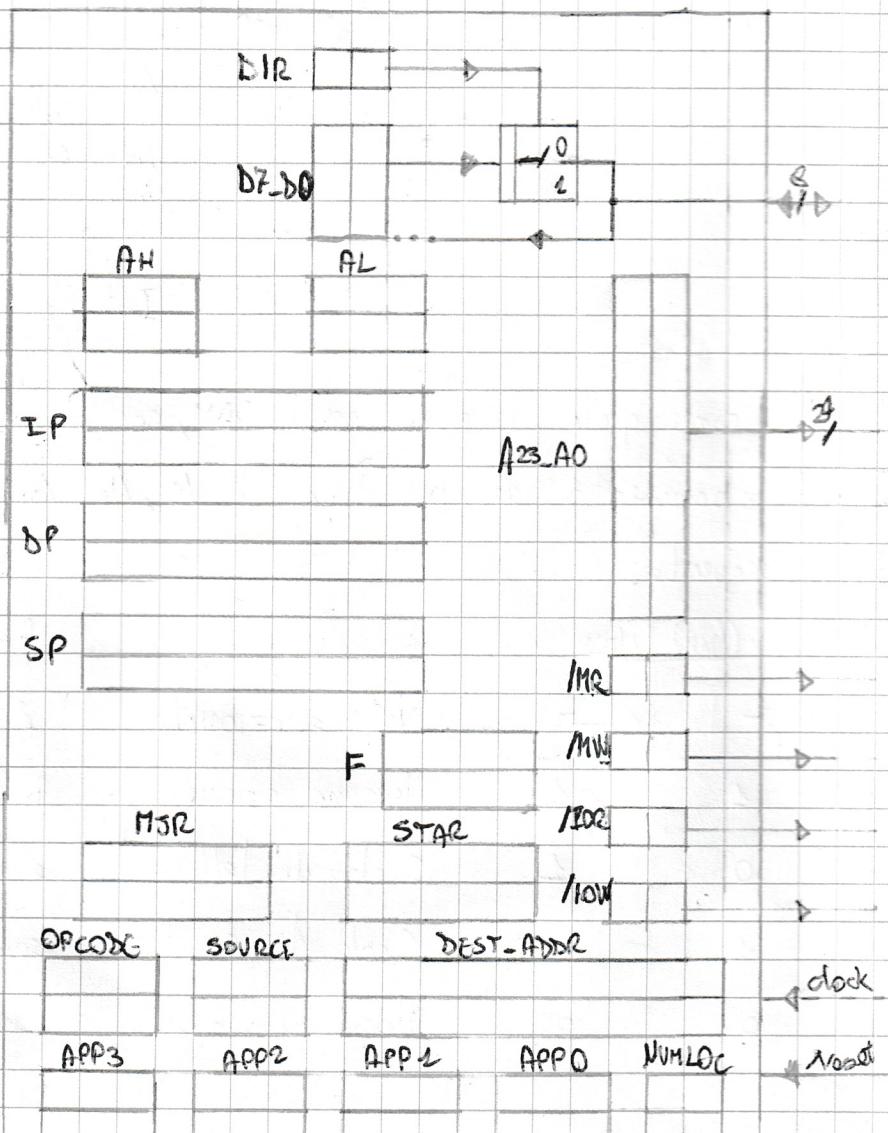
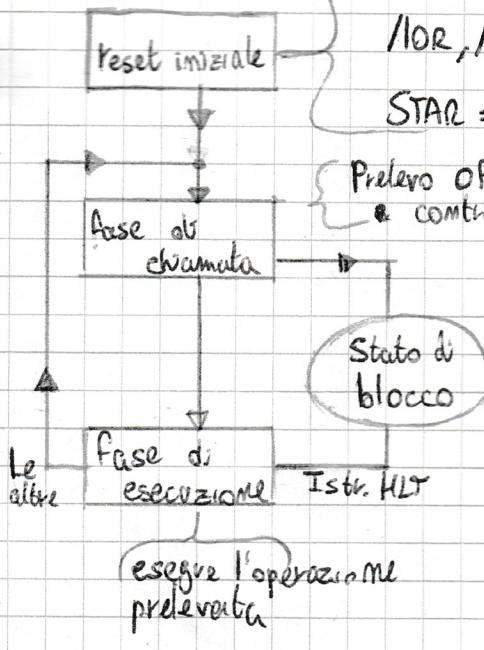
Evoluzione del processore

$DIR = 0 \rightarrow d_7$ - do finisce in alta impedenza

$\text{HOR}, \text{HOM}, \text{MO}, \text{MW}$ som buttar av

STAR = Fetch0, IP = 'HFF0000, F = 'H00

Preferisco l'OPCODE di ciò che è puntato da IP, controllo il formato ed eseguo il fatto corrispondente e controllo se valido



Nella memoria le ~~le~~ letture possono essere continue (con MR=0 dopo tutte le letture).

La scrittura in memoria, come qualsiasi operazione in I/O, è obiettiva quando se ne fanno una per volta. Ad ogni scrittura (ovunque sia) DR va posto a 1.

/m w → dati devono essere stabili sul fronte di salita di /m w

/lat → indirizzo pronto ^{un} clock prima di quando si abbassa

/low → i dati sono memorizzati alla sua discesa [mem in salita]

Nel nostro processore i n-sottoprogrammi di lettura e scrittura sono modulari ovvero usiamo readB per leggere 1 byte, readW per 2 byte, readM per 3 e readL per 4 (e la stessa cosa per la WriteB/W/M/L) i valori lett. verranno salvati su APPD a salire ed i valori scritti verranno presi da APPD a salire e messi in D7-D0

Fase fetch

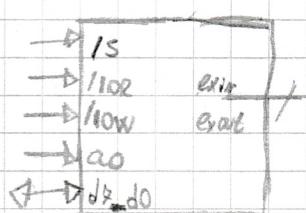
La prima parte è uguale per tutti: leggo l'indirizzo, lo metto in OP code scorro IP di 1 e controllo che formato sia e salto al Fetch preciso - scritto nello specifico qualche pag indietro-

Successivamente c'è l'esecuzione che dipende dall'istruzione stessa.

Da guardare sul libro o dispense!!

Interfaccia

Se manda qui:



Se c'è presente un filo di indirizzo vuol dire che all'interno di tale interfaccia ci sono 2 registri (discernibili dal valore di d_7)

Un'interfaccia molto semplice contiene 2 registri

RBR \rightarrow per i valori in ingresso, TBR \rightarrow per i valori in uscita

Se lo volessi un po' più sicuro, ovvero assicurarmi che il dato presente sia nuovo (o che quello precedente sia stato letto) devo aggiungere 2 registri di stato (uno per ricevere e uno per trasmettere) i quali contengono un Flag (FI) (FO) per capire se TBR è vuoto e RBR pieno

$F_I = 1$ nuovo dato presente $F_O = 1$, trasmettitore vuoto

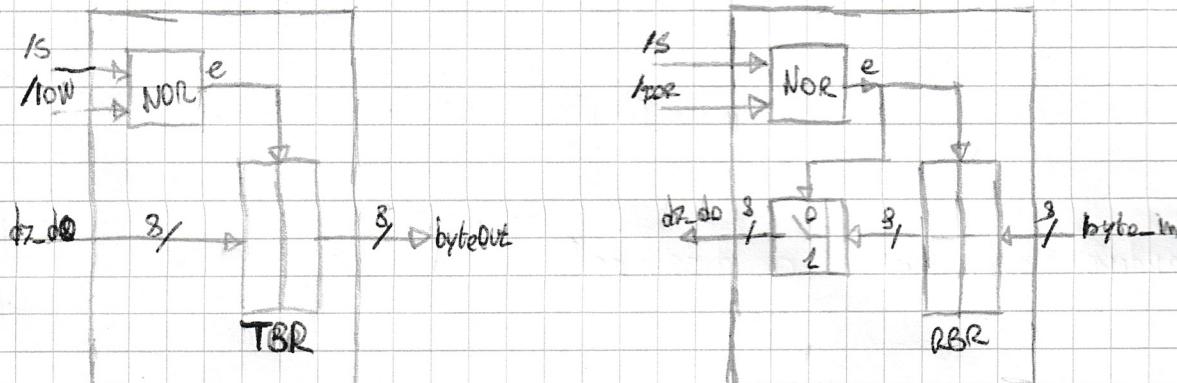
Parallela

Ovvio viene trasmesso un byte per volta

Vediamo quelle semplici di ingresso ed uscita

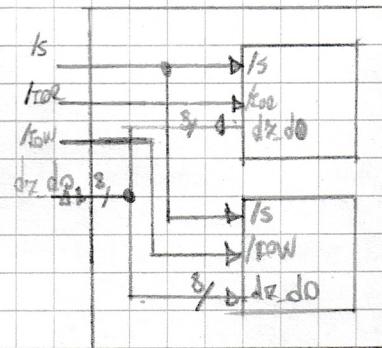
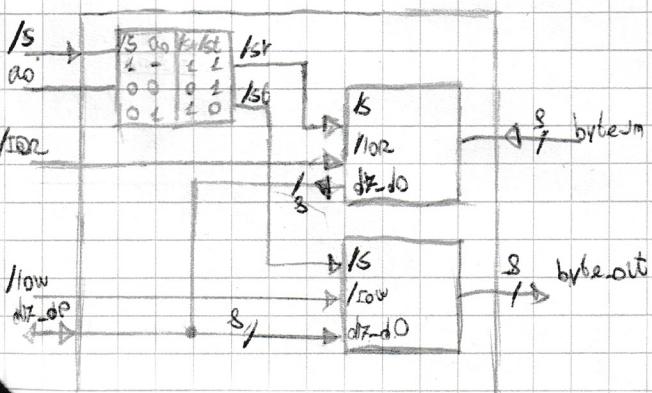
uscita

Ingresso



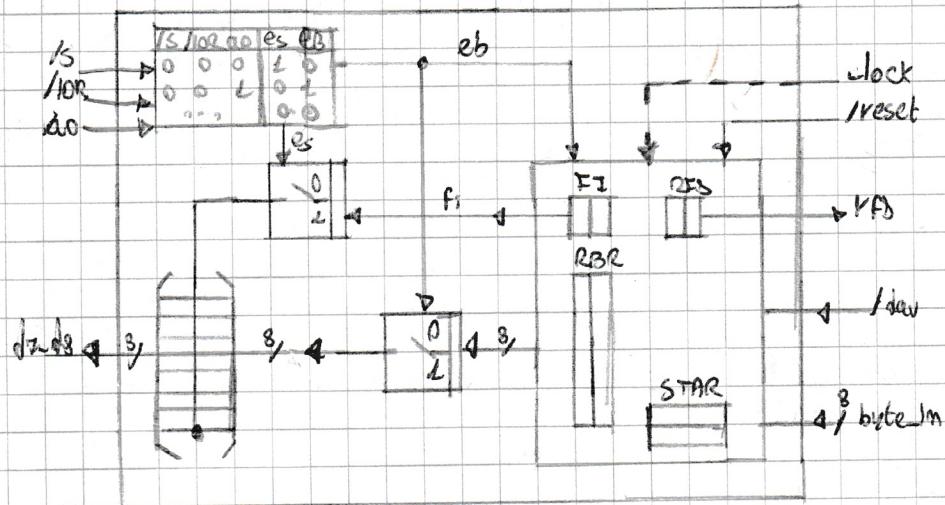
Possiamo anche unire entrambe in un'interfaccia parallela unica componendo 2 registri:

O com uno solo?

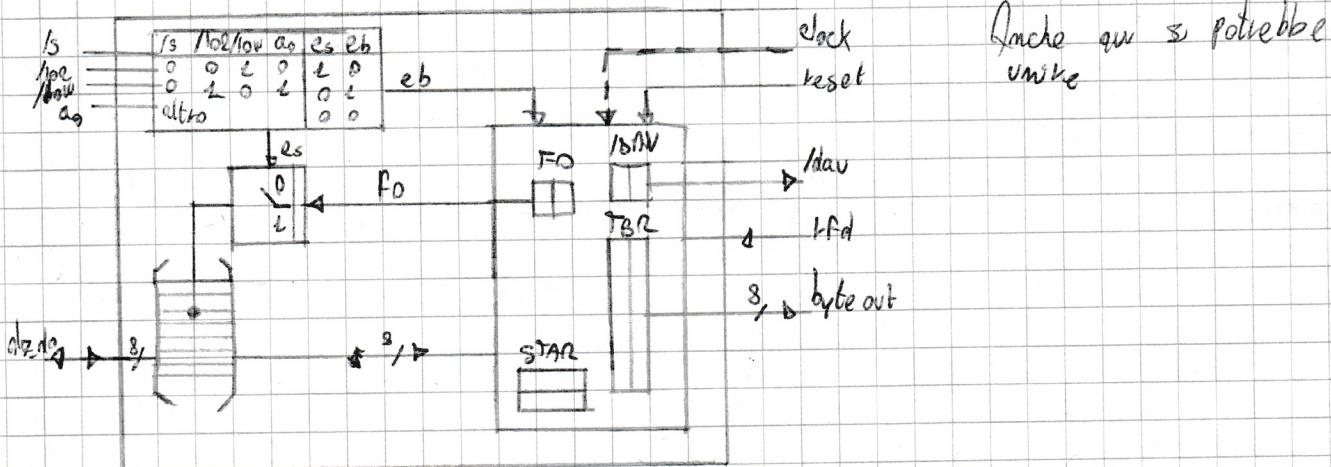


Vediamo adesso quelle con handshake

Ingresso



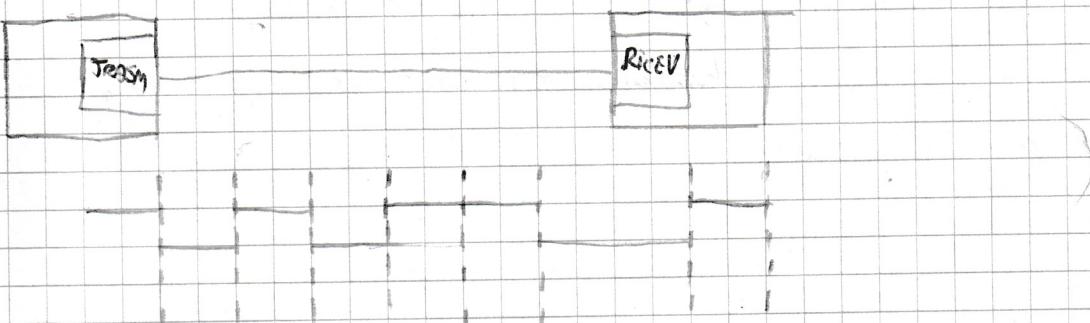
Uscita



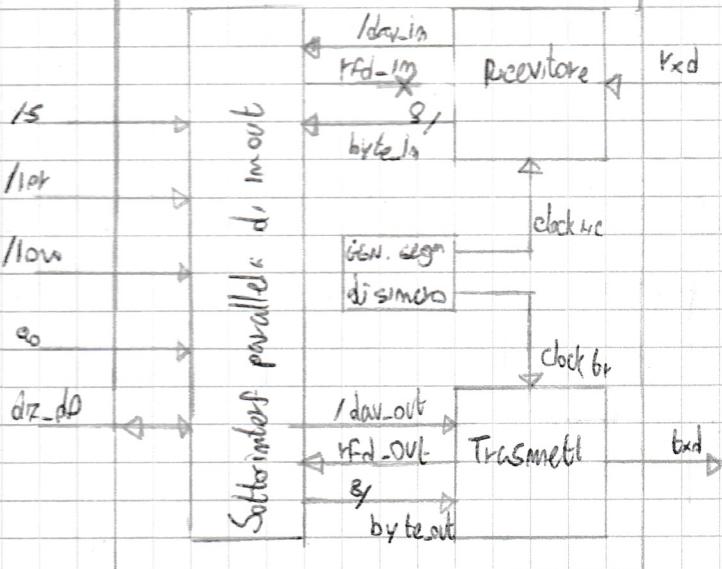
Serial

Questo tipo di interfaccia trasmette al dispositivo un bit per volta

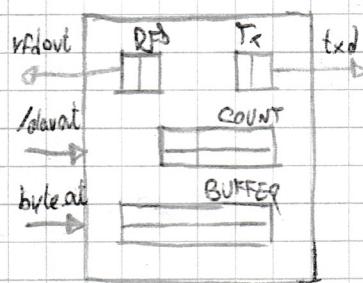
Come avviene la comunicazione seriale



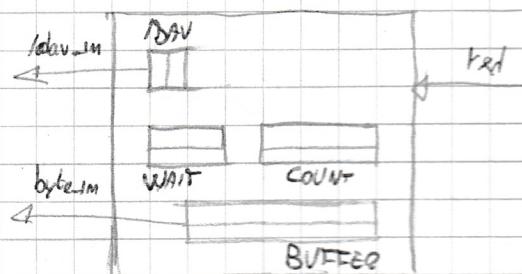
Vediamo come si realizza nella pratica



Vediamo adesso trasmettitori e ricevitori nei particolari:



Trasmettitore



Ricev (clock + veloce)

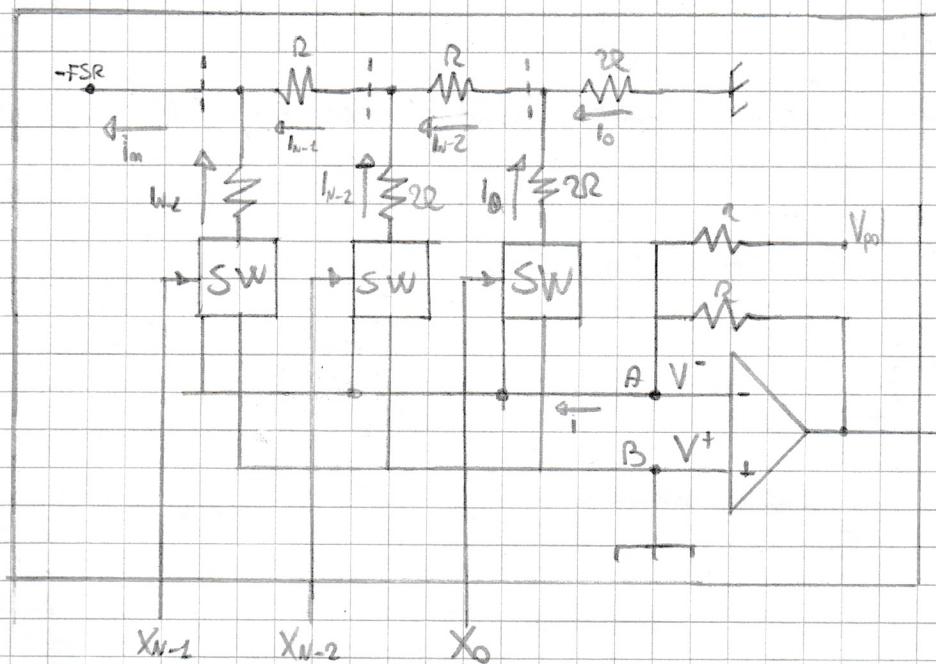
Conversione analogico/digitale e viceversa

Nelle trasmissioni dei dati (in ambo i versi) dato che i circuiti non sono ideali si dice che è presente un errore di non linearità.

Altro problema presente solo da analogica a digitale è detto di quantizzazione ed è dovuto dall'arrotondamento applicato per passare da grandezza continua a discreta.

Vediamo le strutture di questi convertitori.

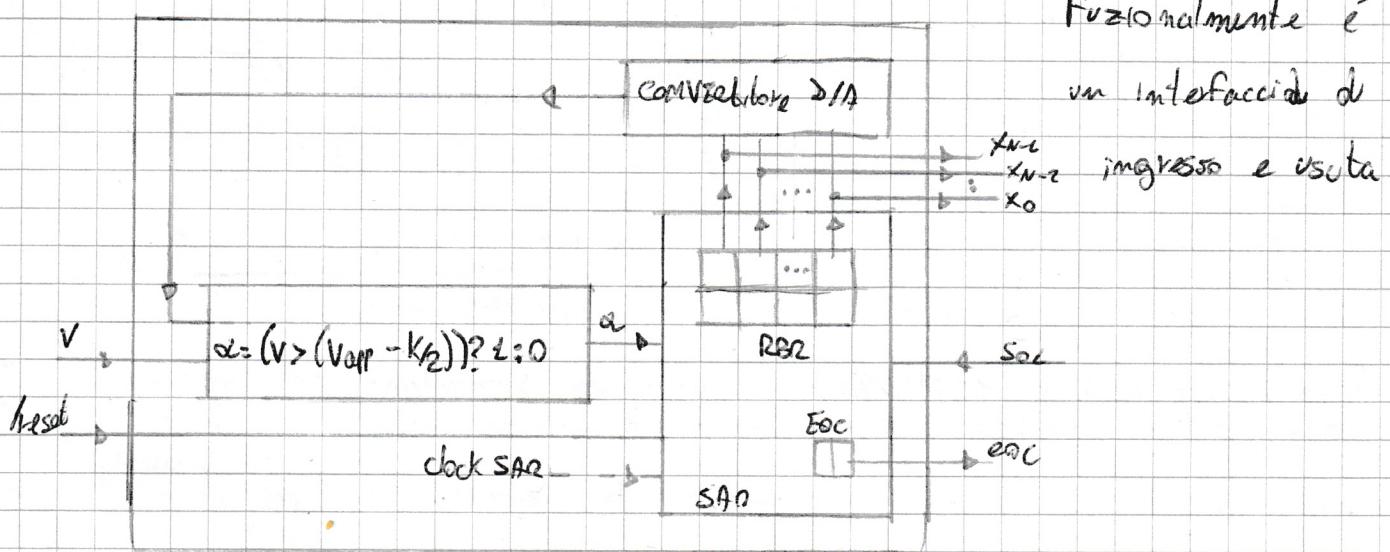
Convertitore D/A



È una rete esclusivamente combinatoria

Equivale ad un qualcosa che possiamo mettere in un interf parallela di uscita senza hardware

Convertitore A/D (approssimazione successiva)



Funzionalmente è un interfaccia di ingresso e uscita