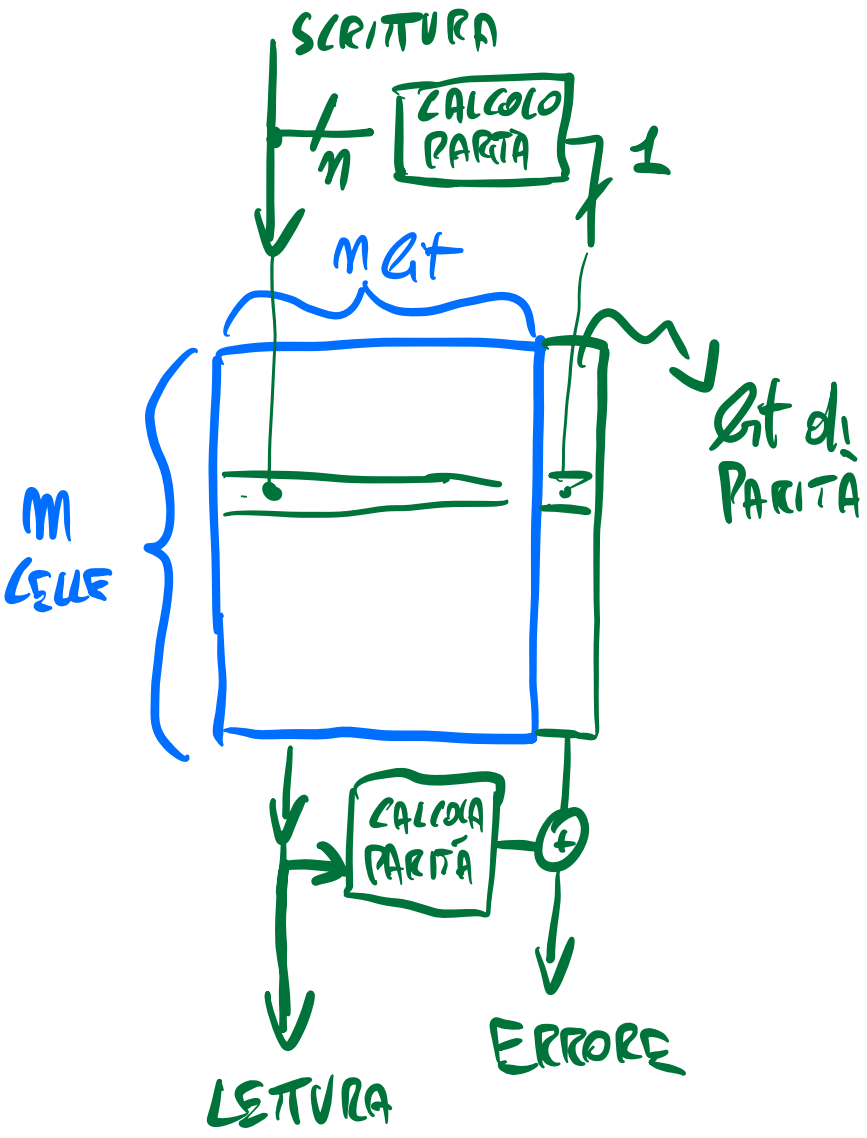
CODICI DI CORREZIONE

- BIT DI PARITÀ



0 1 1 1 : 1

1 0 1 0 : 0

- COSTA POCO
  - 1 bit PER CELLA
- RILEVA # DISPARI DI ERRORI
- NON LI CORREGGE

- CODICE DI HAMMING
  - CODICE LINEARI: SEMPLICE
  - COSTA DI PIÙ DI bit DI PARITÀ

⇒ $\log_2(\text{bit})$; 8 bit di word ⇒ 3 bit di cod.

⇒ RILEVA E CORREGGE 1 ERRORE

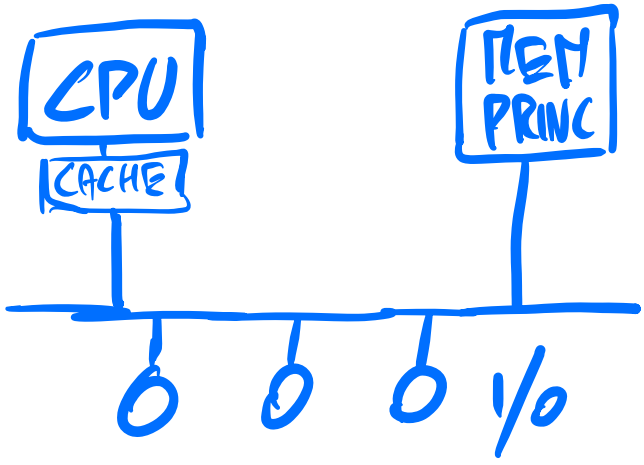
⇒ RILEVA 2 ERRORI

"SEC DED"

SINGLE DOUBLE

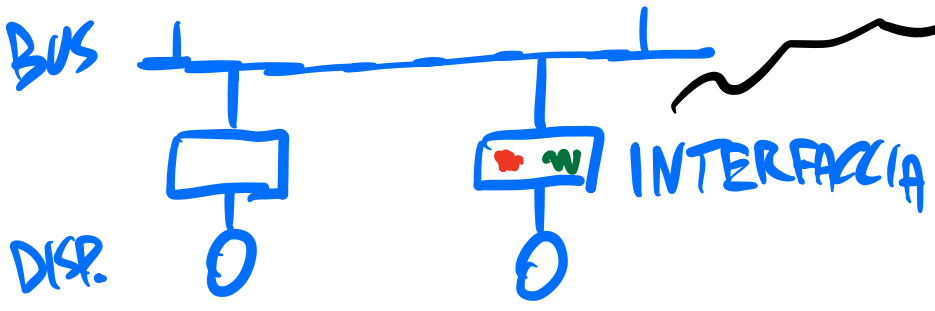
- DRAM HA CODICI DI HAMMING

INPUT/OUTPUT



- PER SCAMBIARE INFO CON ESTERNO
  - SOTTO FORMA DI BIT
  - SENSORI, TASTIERE, SCHERMI, DISCHI
- ETEROGENEI PER Q.TITÀ DI DATI, MECCANISMO

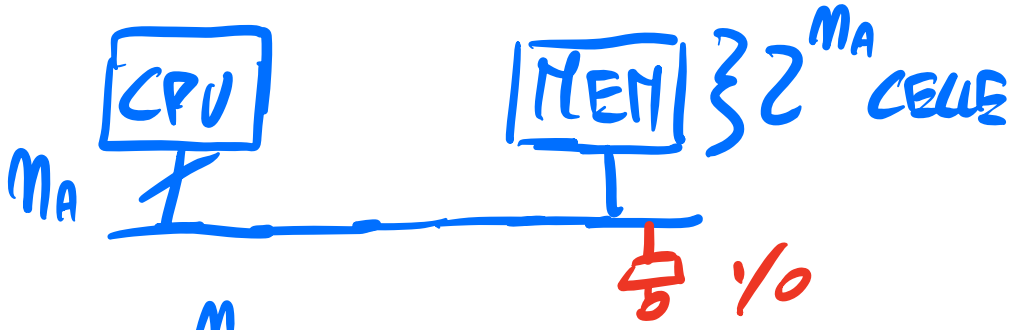
- FUNZIONAMENTO



→ UNO O PIÙ REGISTRI
- LA CPU CI PUÒ LEGGERE E SCRIVERE, USANDO IL SUO INDIRIZZO
- IL DISPOSITIVO GESTISCE

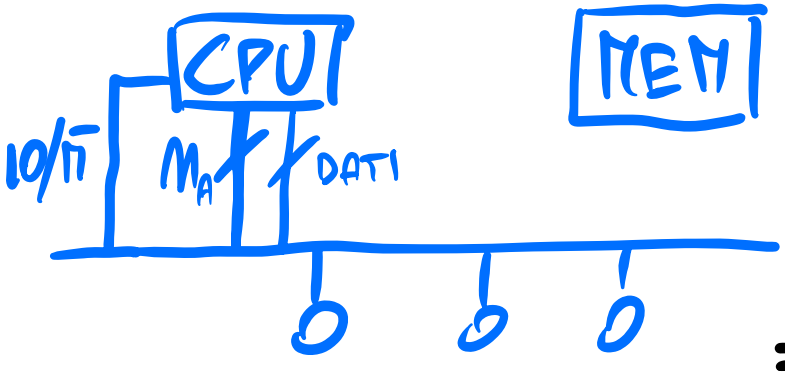
LA COMUNICAZ. CON INTERFACCIA
- È N MEMORIA
⇒ SEMPLICE

INDIRIZZAMENTO



2^{M_A} CELLE DI MEM.
E I DISP DI I/O?
QUALI INDIRIZZI?

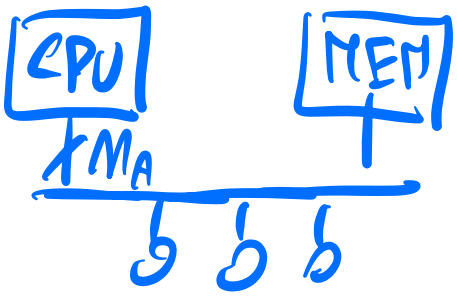
ISOLATED I/O



IO/$\overline{M}$ È UN bit CHE AFFIANCA ADDRESS BUS
↳ IO/$\overline{M}$ = 1 ⇒ INDIRIZZA I/O
↳ IO/$\overline{M}$ = 0 ⇒ INDIRIZZA MEM

⇒ RADDOPPIO SPAZIO DI INDIRIZ.
2^{M_A} CELLE DI MEM
2^{M_A} REGISTRI DI I/O

MEMORY MAPPED I/O



NO I/O FT

UN INDIRIZZO { MEM / I/O

TUTTI I COMPONENTI DEVONO CONOSCERE LA STRATEG.

TUTTE E 2 STRATEGIE USATE

IMPLICAZ. SU ASSEMBLY ISOLATED I/O IN/OUT

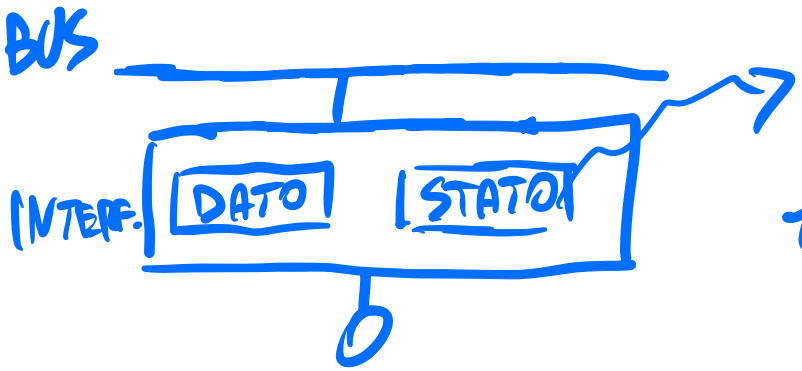
M. MAP I/O MV LD, STR

- MECCANISMO

TUTTO SEMPLICE SE SENSORE.

↳ e.g. SENS DI TEMP. => CPU LEGGE REG.

1. COME CI SI SINCRONIZZA CON <u>FLUSSO</u> DI DATI?
2. COME FA LA CPU A SAPERE SE C'È NUOVO DATO?

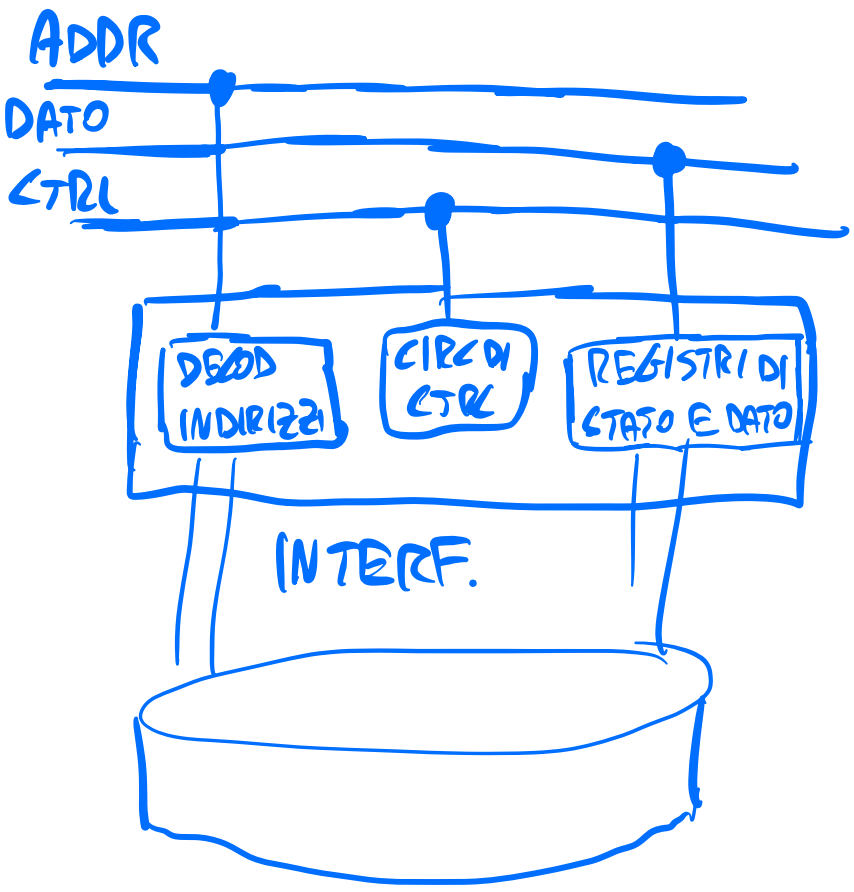


INFORMAZ. DI STATO, NO DATI

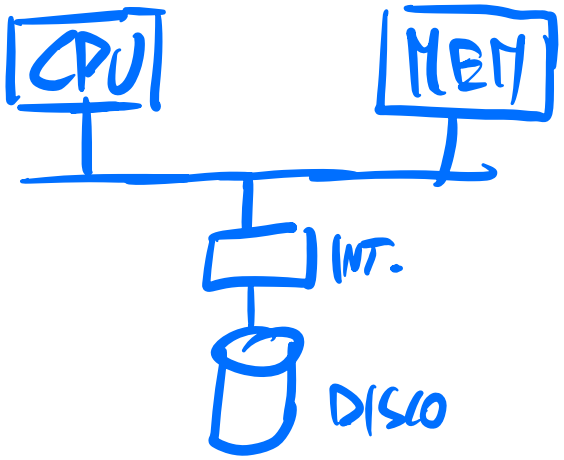
1. I/O → CPU { "UN NUOVO DATO" / "DISP. È SPENTO"

2. CPU → I/O { "HO LETTO DATO" / "SPEGNITI!"

INTERFACCIA



GESTIONE DI I/O



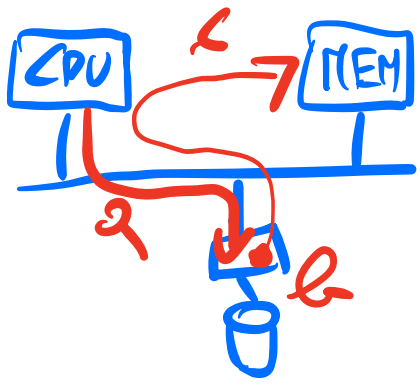
ESEMPIO:

LA CPU VUOLE CARICARE IN MEM. BLOCCO i DA DISCO

DIM (BLOCCO) = 1024 B

3 MECCANISMI: DAL MENO AL PIÙ EFFICIENTE

1. POLLING



a. CPU DICE A DISCO CHE VUOLE LEGGERE BLOCCO i

b. QUANDO DISCO PRONTO, SETTA UN FLAG IN REG. DI STATO, E SCRIVE DATO IN REG. DI DATO

c. CPU PERIODICAMENTE CONTROLLA SE C'È NUOVO DATO (LEGGENDO REG. DI STATO)
↳ SE C'È, LO COPIA IN MEM.

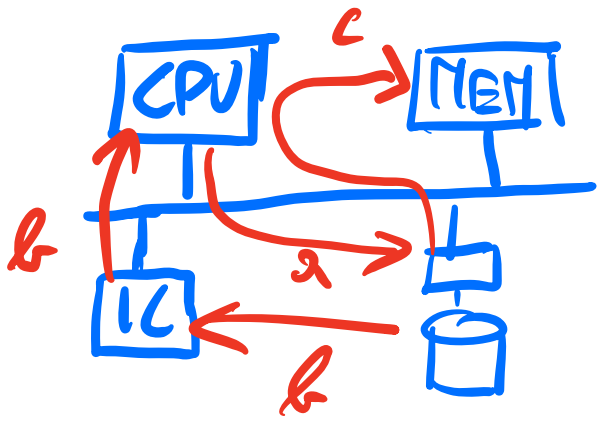
b, c: RIPETUTI FINO A FINE DEL TRASF.

e.g. BLOCCO DA 1024 B; BUS DATI 32 b
⇒ 256 ISTRUZ PER MUOVERE DATI
+ ISTRUZ. POLLING A VUOTO

PRO: SEMPLICE

CONS: INEFFICIENTE

2. INTERRUPT



a. CPU DICE A DISCO DI LEGGERE BLOCCO i

b. QUANDO IL DISCO HA UN NUOVO DATO
   => LO DICE A I.C.
   => L'I.C. LO DICE A CPU

c. LA CPU ESEGUE LA I.S.R. CON CUI COPIA IL DATO DA I/O A MEM.

=> SI RIPETE FINO A FINE BLOCCO

PRO: NON CI SONO CICLI DI POLLING A VUOTO

CONS: COSTA
LA CPU FA OPERAZ. STUPIDA: IL TRASFERIM.
INT. HANNO UN OVERHEAD
↳ INEFFICIENTE SE FLUSSO VELOCE

3. DMA ...