La gestione dell'I/O

Introduction, programmed I/O

Sandro Savino (sandro.savino@dei.unipd.it)
Department of Information Engineering, University of Padova

Argomenti:

- Il problema della gestione dell'I/O
- Come funzionano alcuni device (hard disk, monitor lcd, USB)
- Come comunicano i device (handshaking, seriale vs. parallelo, comunicazioni a pacchetti)
- La gestione dell'I/O programmato

Materiale:

- Capitolo 11

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Dispositivi di I/O

- Ogni elaboratore deve possedere dei dispositivi (periferiche) di I/O
- Sono necessari:
  - Per programmare l'elaboratore
  - Per vedere i risultati dell'elaborazione

- Un elaboratore senza dispositivi di I/O sarebbe poco utile!

La gestione dell' I/O

È un argomento:

- **Rilevante**

  l'interazione che abbiamo quotidianamente con gli elaboratori avviene proprio tramite I/O

- **Interessante**

  la gestione dell'I/O richiede di far lavorare insieme due mondi «diversi»:
  - Da un lato il mondo di funzionamento regolare e cadenzato dell'elaboratore
  - Dall'altro l'innata «asincronia» che caratterizza tutte le richieste di I/O

..la gestione degli eventi asincroni è un problema trasversale in tutta l'informatica (es. SO, UI)

Dispositivi di I/O

- Esistono moltissimi tipi diversi di dispositivi di I/O
  - Dispositivi di input / output utente come mouse, tastiere, touchpad e il monitor o l'impianto audio
  - Dispositivi di storage
  - Dispositivi di comunicazione
  - Dispositivi di input / output «macchina» come sensori ambientali (temperatura, luce, …) e controllori o attuatori (apri porta)
- Nota: con dispositivo di I/O si intende un dispositivo che è controllato dalla CPU: la scheda di rete è un dispositivo di I/O, il router no






La tastiera

- La tastiera ha un controller interno che N (centinaia) volte al secondo controlla se ogni tasto è premuto o no
- Il meccanismo è simile a quanto visto per la RAM: i tasti sono collegati ad una «scan matrix» e il controller «accendendo» la linea x e y può vedere se il tasto al suo incrocio è premuto o no
- Il controller è in grado di rilevare due stati per il pulsante:
  - Premuto (depressed) [make code]
  - Non premuto (released) [scan code]
- Lo stato del tasto viene comunicato all'elaboratore e il codice del tasto premuto è tradotto in un codice ASCII

Nota: premere più forte il tasto non farà andare più veloce la macchinina!

Hard disk

- L’hard disk è una memoria di massa; i dati sono memorizzati in file, organizzati in cartelle
- Un hard disk è composto da più dischi che ruotano (detti piatti), ognuno dotato di due testine a lettura magnetica (una per lato)
- Ogni piatto è diviso in tracce concentriche, numerate (i cilindri sono l’insieme di tracce con lo stesso numero sui diversi piatti)
- Ogni traccia è divisa in settori, ognuno contenente N byte
- Da un punto di vista logico, i settori sono raggruppati in cluster (also: block o allocation unit); un file occupa uno o più cluster (se il file è più piccolo del cluster, avremo dello spazio vuoto, detto slack space)
- Tempo di accesso: Seek time + Rotational delay + Controller time
- Transfer rate: numero byte per settore / tempo per leggere un settore

Monitor

- L’immagine è composta da pixel
- Ogni pixel è composto da 3 sub-pixel, uno per ogni colore primario della sintesi additiva (rosso, verde, blu)

LCD: Liquid Cristal Display (display a cristalli liquidi)

- I cristalli liquidi sono dei cristalli che si possono muovere; muovendosi cambiano la polarizzazione della luce che li attraversa; inoltre sono suscettibili ai campi elettrici: è possibile controllare la rotazione dei cristalli tramite una corrente
- Nei pannelli LCD (generalizzando):
  - una fonte di luce è messa dietro i pixel e fornisce il colore bianco; la luce viene polarizzata
  - ogni sub-pixel ha un filtro colorato
  - ogni sub-pixel ha il suo cristallo liquido: modificando la tensione si muovono i cristalli (che agiscono da filtro polarizzatore): se sono a 90° rispetto alla luce, il pixel diventa nero (la luce non passa)
  - ogni sub-pixel ha un transistor (realizzato su un film sottile Thin Film Transistor) che controlla un condensatore e permette di pilotare il sub-pixel
- Il controllore dello schermo «accende» i pixel in modo simile a quanto accade su una DRAM

Maggiori info:
https://youtu.be/jiejNAUwcQ8

Dispositivi di I/O

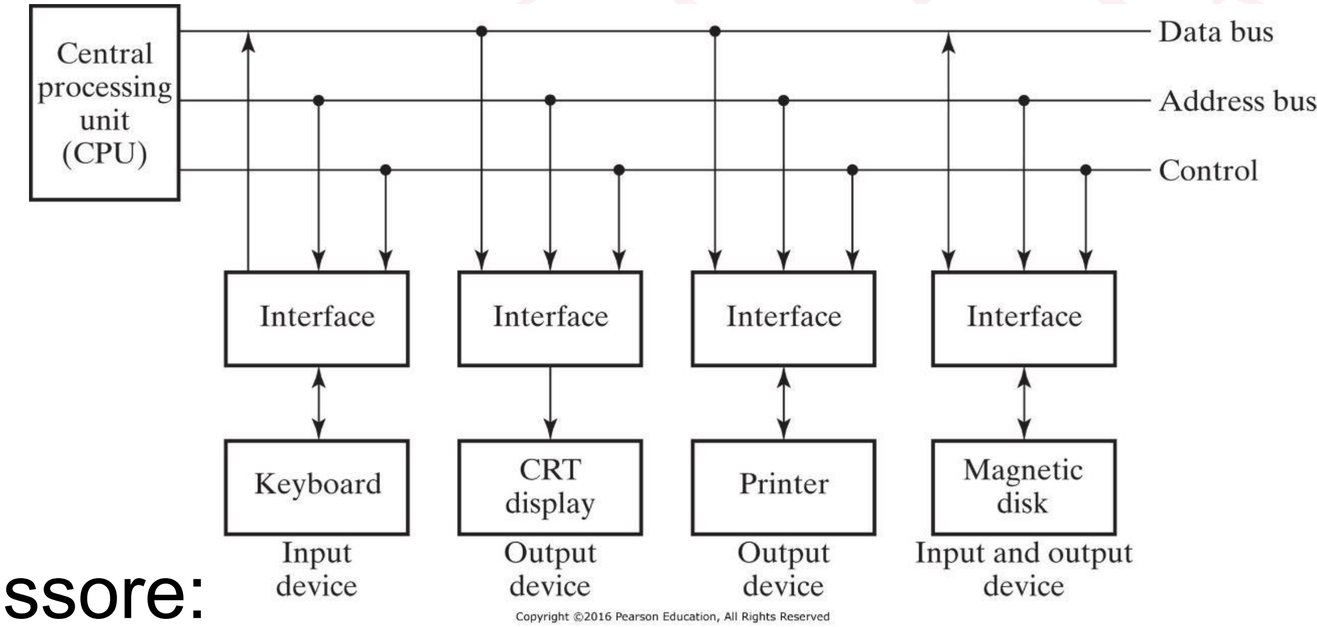
- Hanno tutti qualcosa in comune:
  - Devono comunicare con il processore e la RAM
  - Trasferiscono dati (in ingresso e/o uscita)

- In generale però sono molto diversi tra loro:
  - la velocità di trasmissione dei dati (data rate) può variare molto:
    - Tastiera: 100 scan al secondo, 128 tasti (7bit) → max 700b/s → <100 Byte/s
    - Hard-disk: HDD 140 Mbyte/Sec, SDD 1.6Gbyte/s / 3.2Gbyte/s
    - Monitor: schermo full-hd: 1920x1080 pixel, codifica RGB a 24bit, refresh-rate di 144hz: circa 900 Mbyte/s
  - Il modo di funzionamento è diverso, e infatti ognuno è governato da un suo controller interno

- Come possiamo farli comunicare con l'elaboratore?

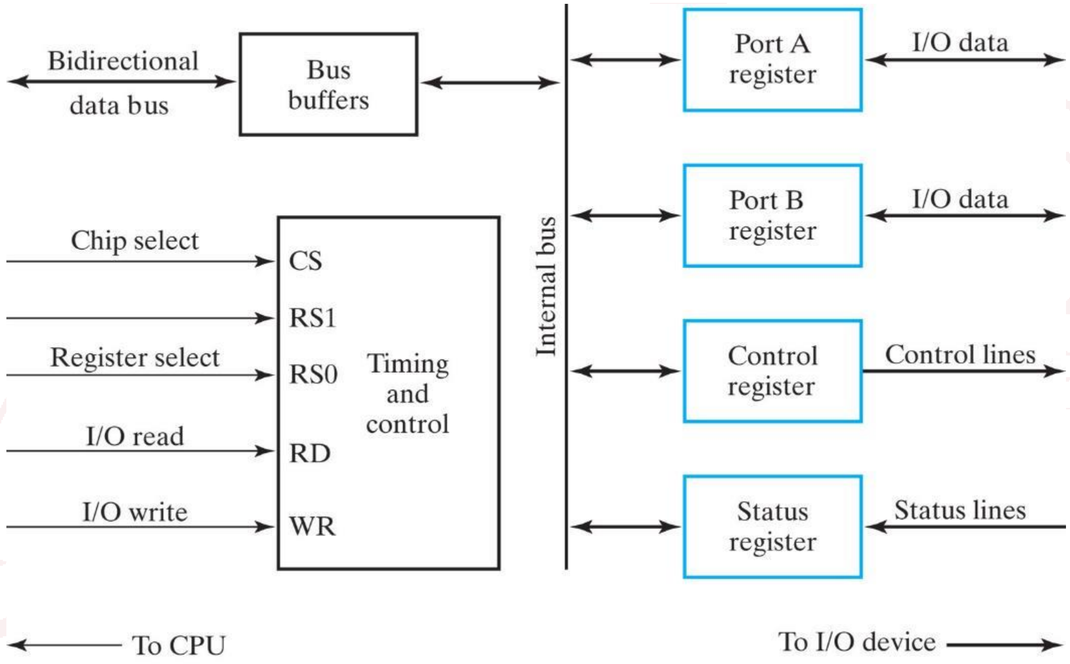
Interfaccia di I/O

- L’elaboratore è dotato di interfacce di I/O
  - Sono collegate al bus dell’elaboratore
  - Sono collegate al controllore del dispositivo di I/O
- Funzione:
  - Controllo
  - Temporizzazione
  - Data buffer
- Per accedere all’interfaccia il processore:
  - Può usare lo stesso bus della memoria (I/O Memory Mapped)
  - Può usare delle linee dedicate (isolated I/O)



Interfaccia di I/O

- L'interfaccia di I/O permette di dialogare in modo «standard» con i più diversi dispositivi di I/O
- Il produttore del dispositivo dovrà realizzare il controller in modo tale da essere compatibile con l'interfaccia di I/O del nostro elaboratore
- Generalizzando, un controller mostrerà all'interfaccia 4 registri:
  - 2 per passare dati in ingresso e uscita
  - 1 per passare codici di comando
  - 1 per passare codici di status
- Il processore
  - userà un segnale di chip select per attivare una precisa interfaccia di I/O
  - due bit RS0 e RS1 per selezionare il registro su cu operare,
  - 2 bit RD e WR per scrivere o leggere tale registro



CS	RS1	RS0	Register selected
0	X	X	None: data bus in high-impedance state
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

Comunicazione tra device

- I dispositivi di I/O sono per loro stessa natura asincroni ed inoltre vanno ad una velocità diversa del processore (ad es. quindi la comunicazione può impiegare più cicli di clock): è necessario trovare un modo per farli comunicare

STROBING:

- Si usa una linea dati singola, dove passare un segnale «strobe»
- Trasmissione da source: S mette il dato da trasferire sul bus e imposta strobe da 0 a 1. D quando vede strobe=1 si prepara a trasferire il dato. Dopo un tempo X, S imposta strobe da 1 a 0. D quando vede strobe tornare a 0 copia il dato nel registro
- Ricezione da destination: D imposta strobe da 0 a 1. S quando vede strobe=1 mette il dato da trasferire sul bus. Dopo un tempo X, D prende il dato dal bus e imposta strobe da 1 a 0

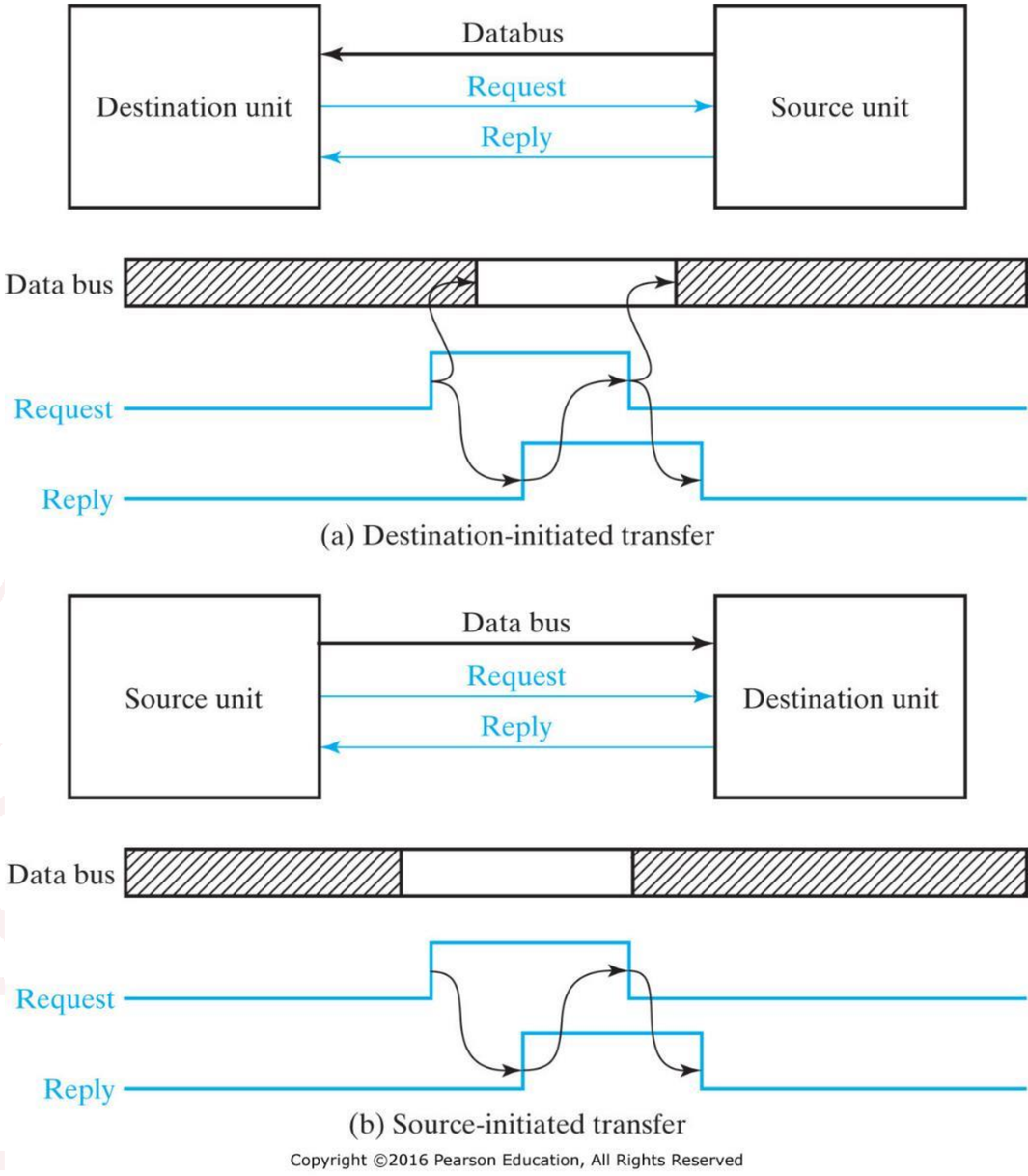
- Problema il «tempo x» potrebbe non essere stato sufficiente, inoltre non ho conferma dell'avvenuto trasferimento

Comunicazione tra device

HANDSHAKING

- Si usano 2 linee: request e reply
- Il concetto è che su una linea un device richiede il dato, con la seconda linea l'altro device risponde che il dato è pronto
- Ricezione da D:

REQ	REP	
0	0	Stato iniziale
1	0	D richiede un dato
1	1	S indica che il dato è sul bus
0	1	D indica che ha letto il dato
0	0	S indica che è pronta a mandare altri dati



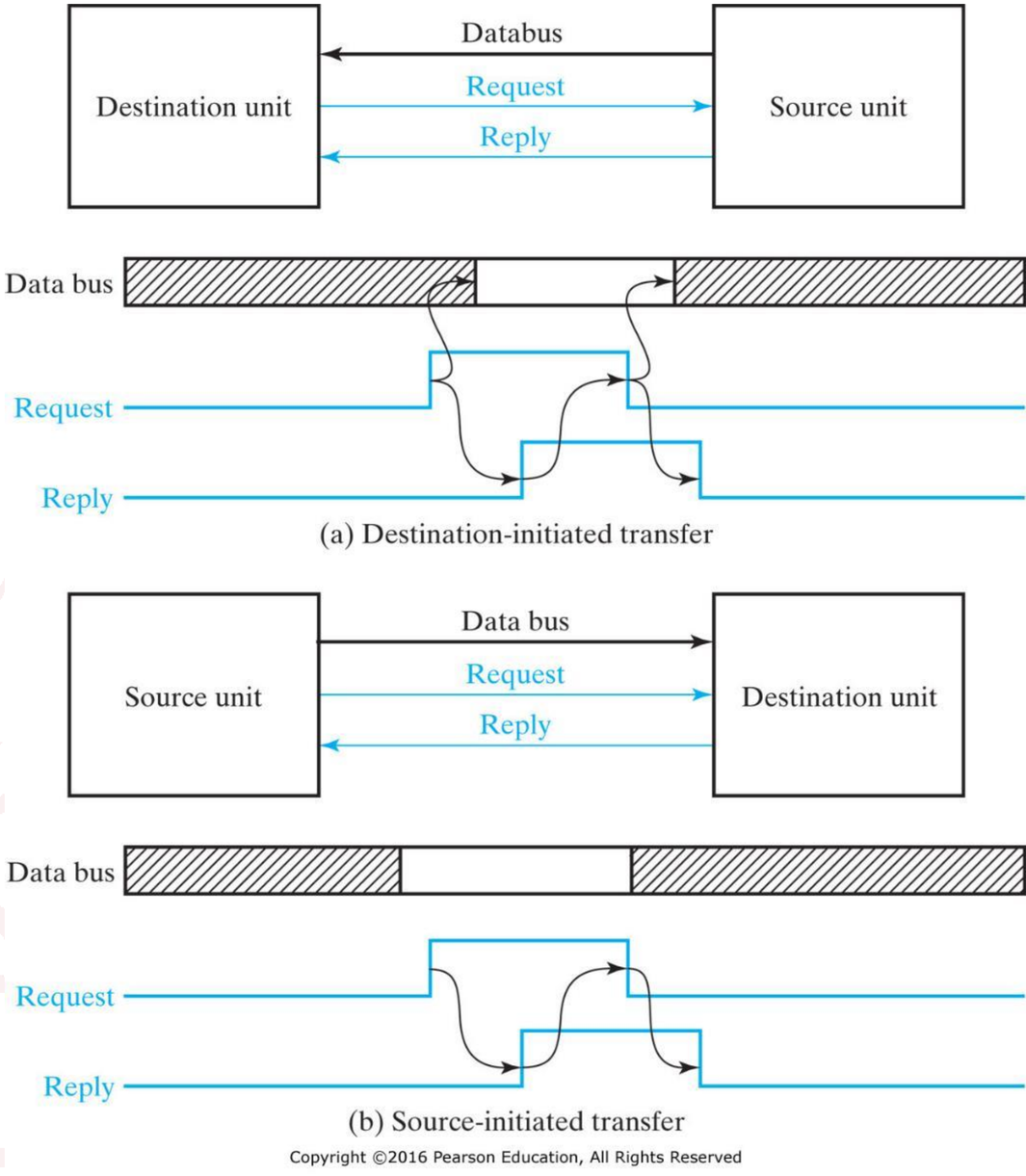
Comunicazione tra device

HANDSHAKING

- Invio da S:

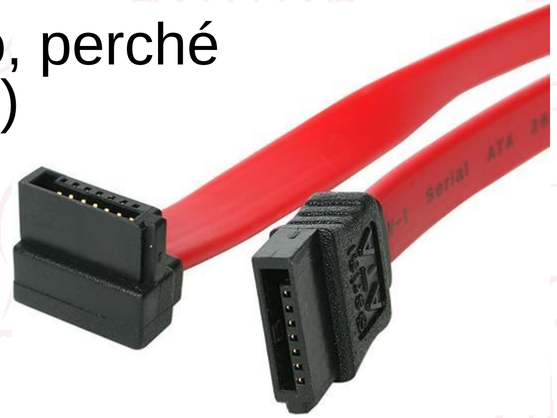
REQ	REP	
0	0	Stato iniziale
1	0	S avverte che sul bus c'è un dato pronto
1	1	D indica che ha letto il dato
0	1	S indica che ha rimosso il dato dal bus
0	0	D indica che è pronta a ricevere altri dati

- Protocollo più sicuro: entrambi i dispositivi hanno conferma del trasferimento dei dati
- Un time-out può essere usato per rilevare errori nei dispositivi
- Usato tra interfaccia di I/O e device

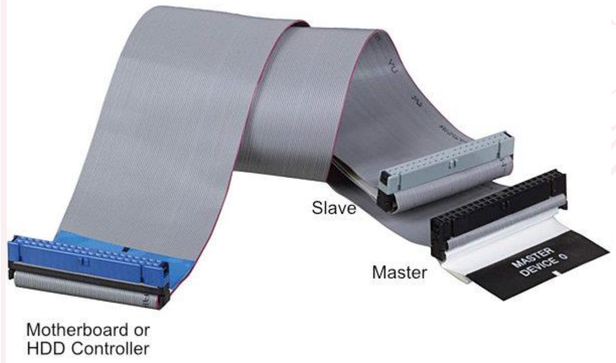


Comunicazione tra device

- Una volta che due device sono pronti a comunicare, possiamo effettuare il trasferimento
- Questo può avvenire:
  - più bit alla volta, con un collegamento parallelo (più costoso, ma più veloce e problema induzione)
  - un singolo bit alla volta, con un collegamento seriale (meno costoso, perché ho meno cavi, no problema induzione, ma più lento a parità di clock)

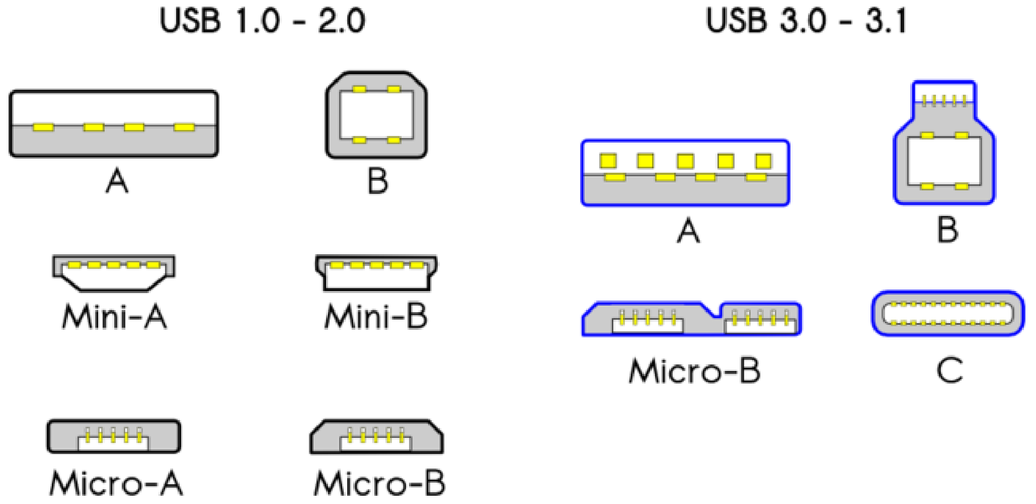


- La trasmissione del dato può essere:
  - Simplex (monodirezionale)
  - Half-duplex (bidirezionale, ma 1 sola direzione per volta)
  - Full-duplex (tx e rx contemporanei)
  - Sincrona (D e S hanno lo stesso segnale di clock)
  - Asincrona (D e S si devono sincronizzare)

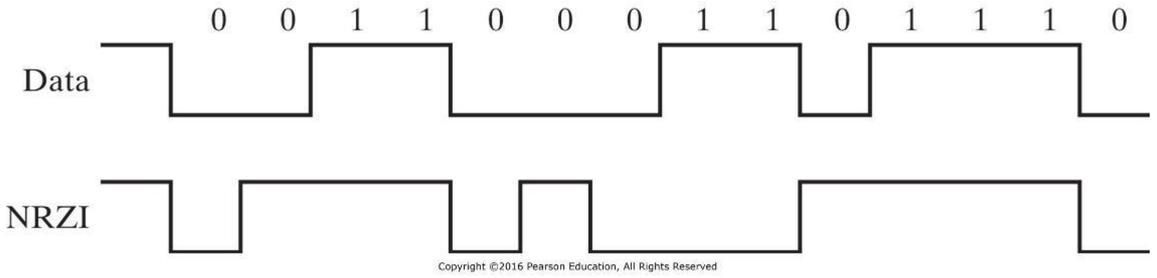


Esempio: USB

- Collegamento USB:
  - Seriale (usa però 4 cavi)
  - A pacchetti (quindi più device possono usar lo stesso bus di trasmissione)
  - La sincronia è mantenuta usando una speciale codifica per i bit: NRZI



Se ho 0	Inverto il segnale
Se ho 1	Mantengo il segnale; dopo 6 bit 1 consecutivi, inserisco uno 0 (inversione)



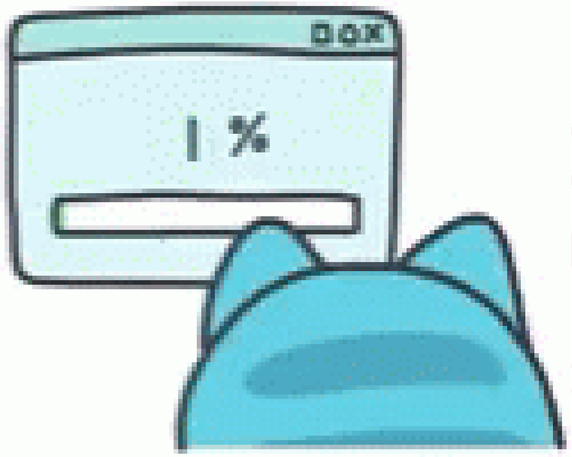
Versione	Velocità massima	Anno
1.0	1.5 Mbps	1996
1.1	12 Mbps	1998
2.0	480 Mbps	2000
3.0	4.8 Gbps	2008
3.1	10 Gbps	2013

Gestione del trasferimento

- Durante un evento di I/O il processore deve gestire il trasferimento dei dati dal dispositivo alla memoria (o viceversa)
- Ci sono tre modi diversi per affrontare la gestione dell'I/O in un elaboratore:
  - I/O Programmato
  - Interrupt
  - DMA
- Possiamo valutare questi approcci sulla base di:
  - Costo e complessità
  - Impatto sul normale funzionamento del computer
  - Tempo di risposta

Impatto sul funzionamento del computer

- Quando avviene un evento di I/O asincrono, l'elaboratore è impegnato a fare altro: la gestione dell'I/O ovviamente non deve in nessun modo perturbare quello che l'elaboratore stava facendo!



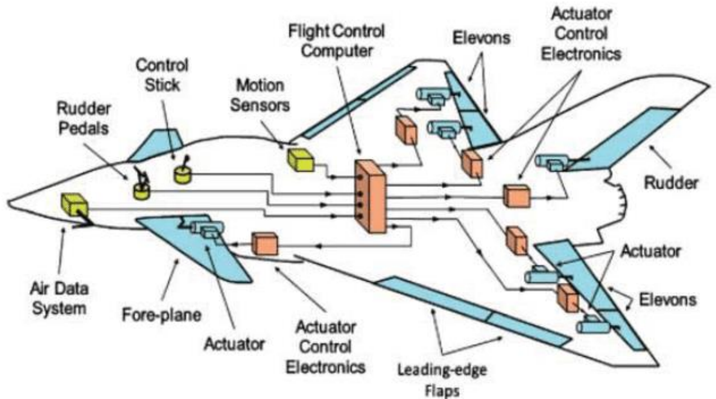
- Inoltre la gestione di un evento di I/O non deve richiedere una elaborazione troppo onerosa, altrimenti le performance del sistema ne risentirebbero

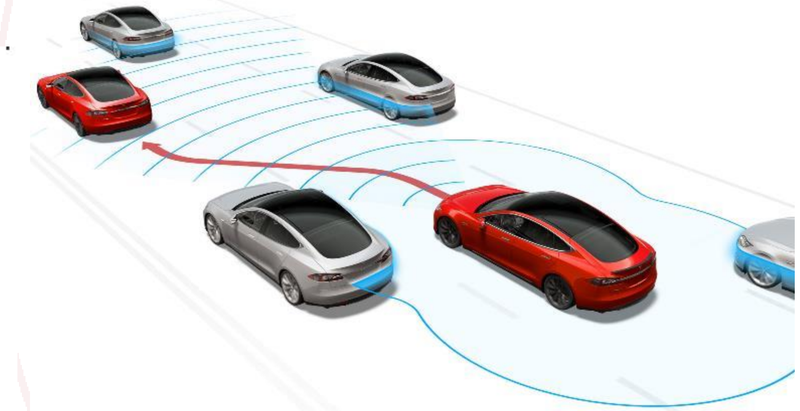
Tempo di risposta

- Il tempo necessario a gestire l'evento di I/O deve essere il più breve possibile altrimenti potremmo avere:
  - Perdita di dati
  - Peggioramento della user experience

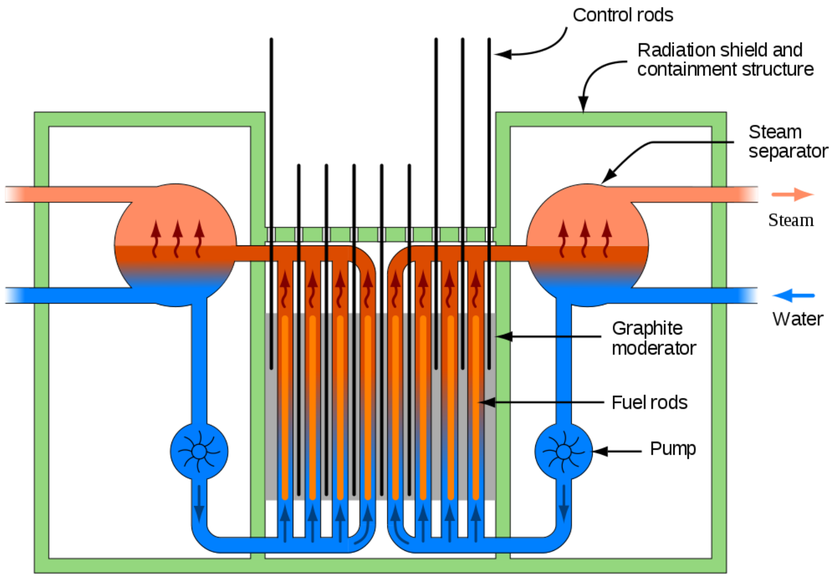
- O anche effetti più disastrosi: in alcuni sistemi di controllo il tempo di risposta è fondamentale

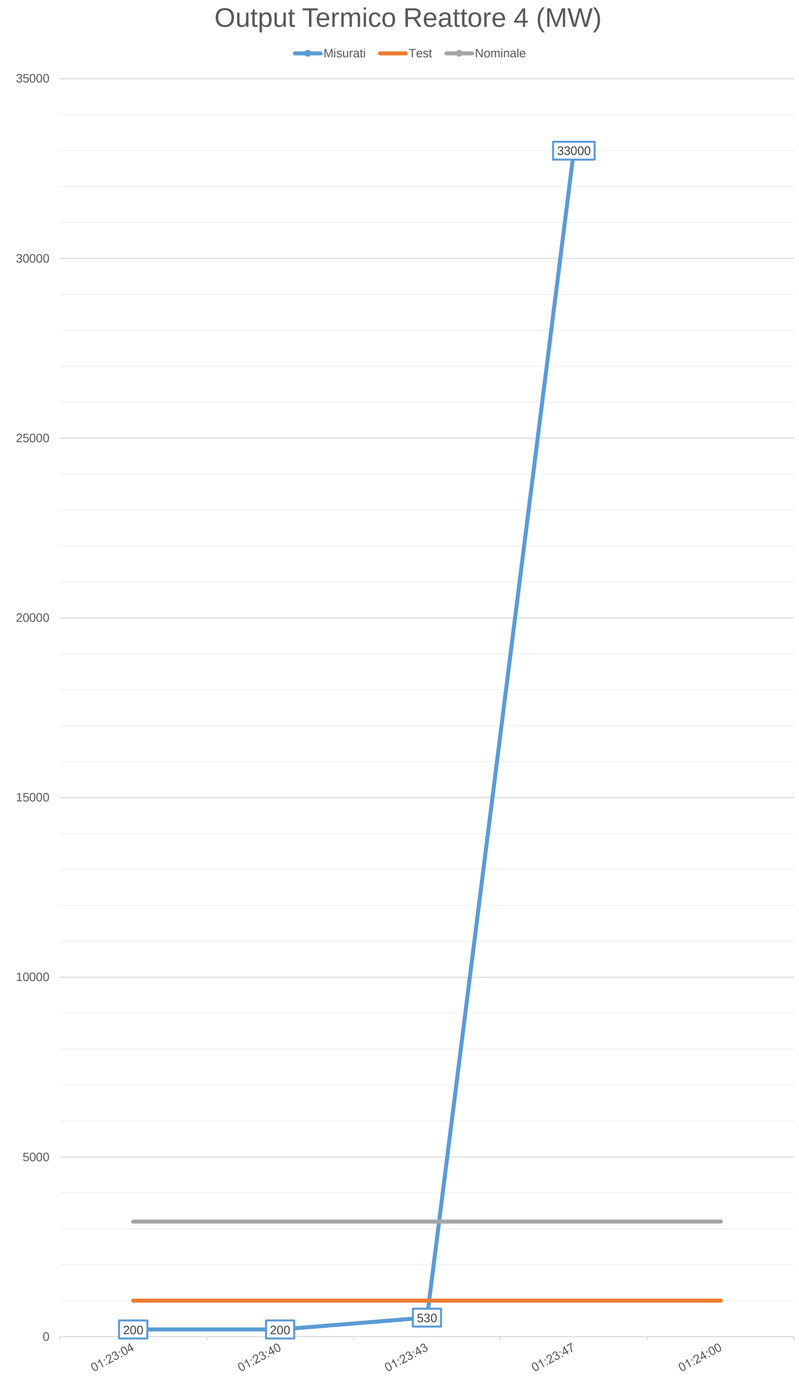
PING: ∞





Negli elaboratori utilizzati per il controllo automatico di alcuni sistemi fisici un ritardo nella risposta può portare a risultati catastrofici

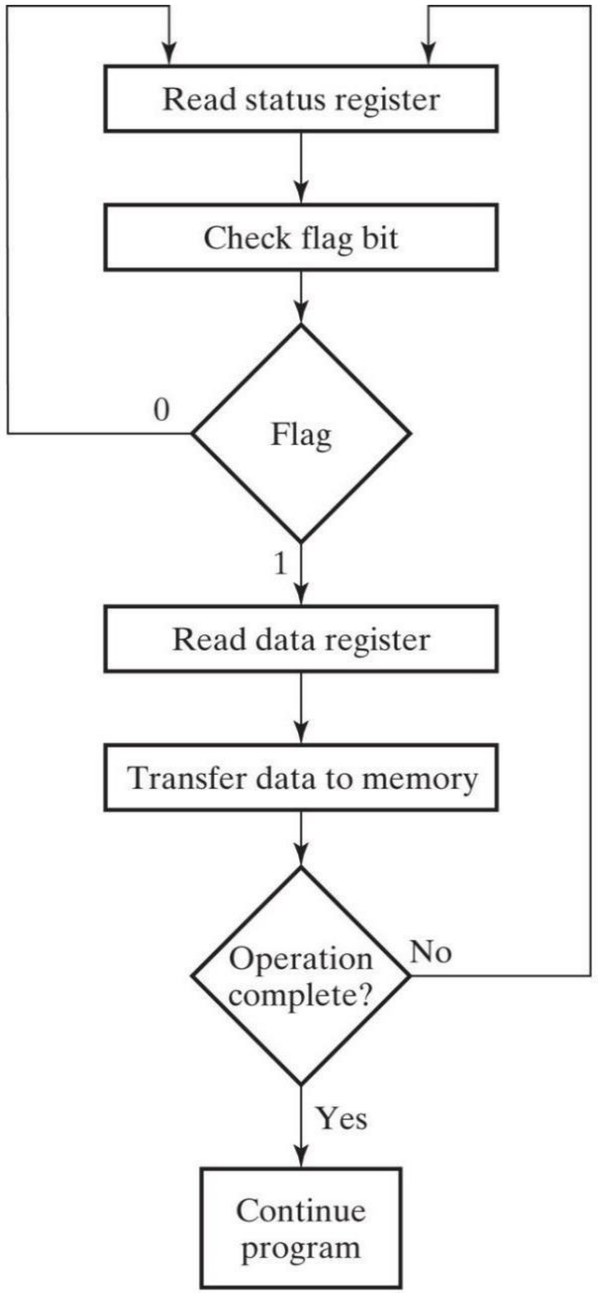




https://en.wikipedia.org/wiki/RBMK
https://en.wikipedia.org/wiki/Chernobyl_disaster

I/O PROGRAMMATO

- La gestione tramite I/O programmato è la più semplice: il processore si fa carico di gestire il trasferimento dei dati dal device alla memoria (o viceversa)
- Usando l'interfaccia di I/O si può fare con un semplice protocollo:
  - Controllo se lo status register dell'interfaccia mi dice che un nuovo dato è pronto
  - Se non è pronto aspetto
  - Se è pronto lo trasferisco in memoria

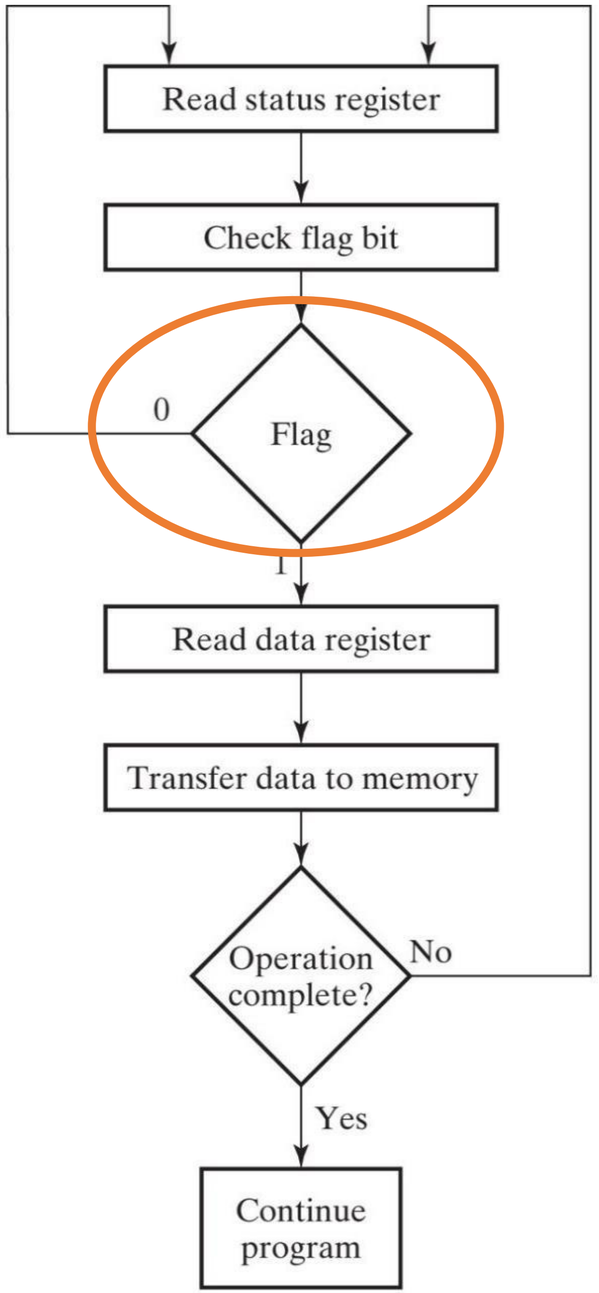


I/O PROGRAMMATO

- Fintanto che non c'è un nuovo dato, la CPU è bloccata ad aspettare il nuovo dato

BUSY WAITING

- Questo è grave perché i dispositivi di I/O sono molto più lenti della CPU, che perde un sacco di cicli:
- Es: un HDD a 7200 rpm ha un tempo di accesso di circa 9ms: in questo tempo, mentre il disco accede al dato, una CPU da 1GHz avrebbe potuto fare circa 10 milioni di operazioni



I/O PROGRAMMATO: pro e contro

PRO

- Molto facile da realizzare
- Tempo di risposta molto veloce(*): appena il dato è pronto viene letto / inviato

(*) se il dispositivo è pronto …

CONTRO

- Mentre è in attesa che il dispositivo sia pronto, il processore non può fare niente altro!
- L'inefficienza peggiora più alta è la differenza di velocità tra il processore e il dispositivo di I/O

Viene usato per monitorare i dispositivi (es. tastiera)