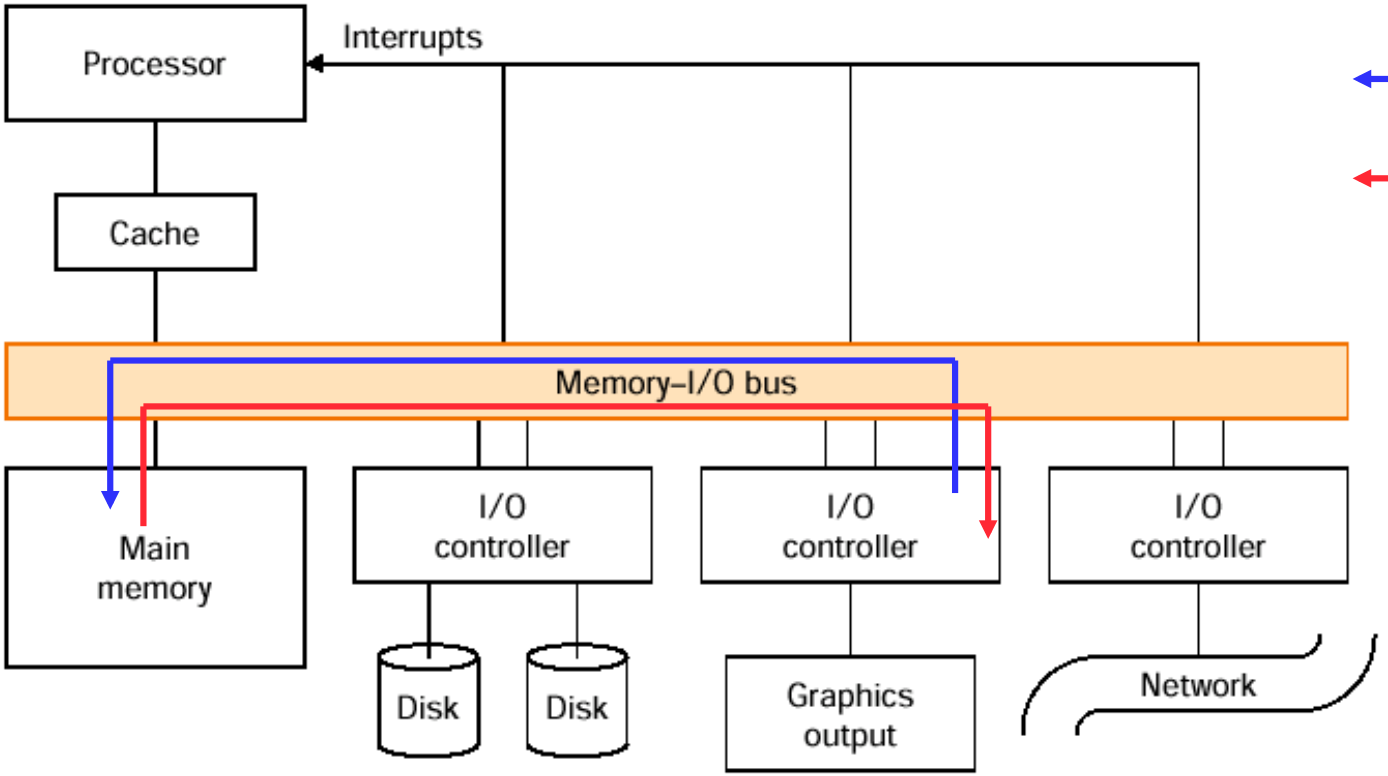
Input / Output

Salvatore Orlando

Input e Output



INPUT

OUTPUT

I dati trasferiti durante le operazioni di I/O possono passare (o meno) dal processore

programmed I/O vs. DMA

- **La figura illustra alcuni dispositivi di I/O, e le connessione con le altre componenti del calcolatore:**
  - **processore e memoria**
- **Progetto del sottosistema di I/O influenzato da vari fattori (oltre alle prestazioni) quali: espandibilità, tolleranza ai guasti**

Performance

- **Le prestazioni complessive di un sistema di dipendono**
  - **dalle prestazioni dei device e dei relativi controller**
  - **dalla connessione tra i device e il resto del sistema**
  - **dalle gerarchie di memoria**
  - **dall'OS**

- **Misure di prestazione**
  - **tempo di accesso (latenza di accesso)**
    - ***Costante* + *DIMENSIONE BLOCCO* / *BANDA DI TRANSFERIMENTO***
    - **La *Costante* è solitamente molto grande, per cui la *Banda di Trasferimento* diventa importante solo per blocchi grandi**
  - **throughput**
    - **quantità di dati trasferite per unità di tempo (B o b al secondo)**
      - banda reale misurata
      - per *grandi* blocchi di dati (costante poco significativa), il throughput può essere approssimato considerando solo la BANDA DI TRASFERIMENTO
    - **quantità di operazioni effettuate per unità di tempo**
      - misura importante per accessi di tipologia mixed (blocchi piccoli e grandi)

Misure di banda e tempi di trasferimento

- Nelle misure di banda rispetto ai dispositivi di I/O, i simboli K, M e G sono da considerarsi multipli di 10
  - es.: MB/s o Mb/s, dove $M=10^6$
- Questo può portare a piccoli errori, se si considera di dover trasferire K, M e GB, dove K, M e G sono multipli di 2
- Esempio
  - devo trasferire 10 MB, e il dispositivo di I/O ha una banda di 5 MB/s
  - Tempo di trasferimento approssimato = 10 MB / 5 MB = 2 s
  - Tempo di trasferimento esatto = 10 MB / 5 MB = $10 * 2^{20} / 5 * 10^6 > 2$ s
- Ci accontenteremo dei tempi approssimati

Tipi di dispositivi

- **Ci sono molti tipi di dispositivi**
  - **comportamento (cioè, input vs. output)**
  - **partner (cioè, chi c'è all'altro estremo del dispositivo)**
  - **banda di trasferimento (data rate)**

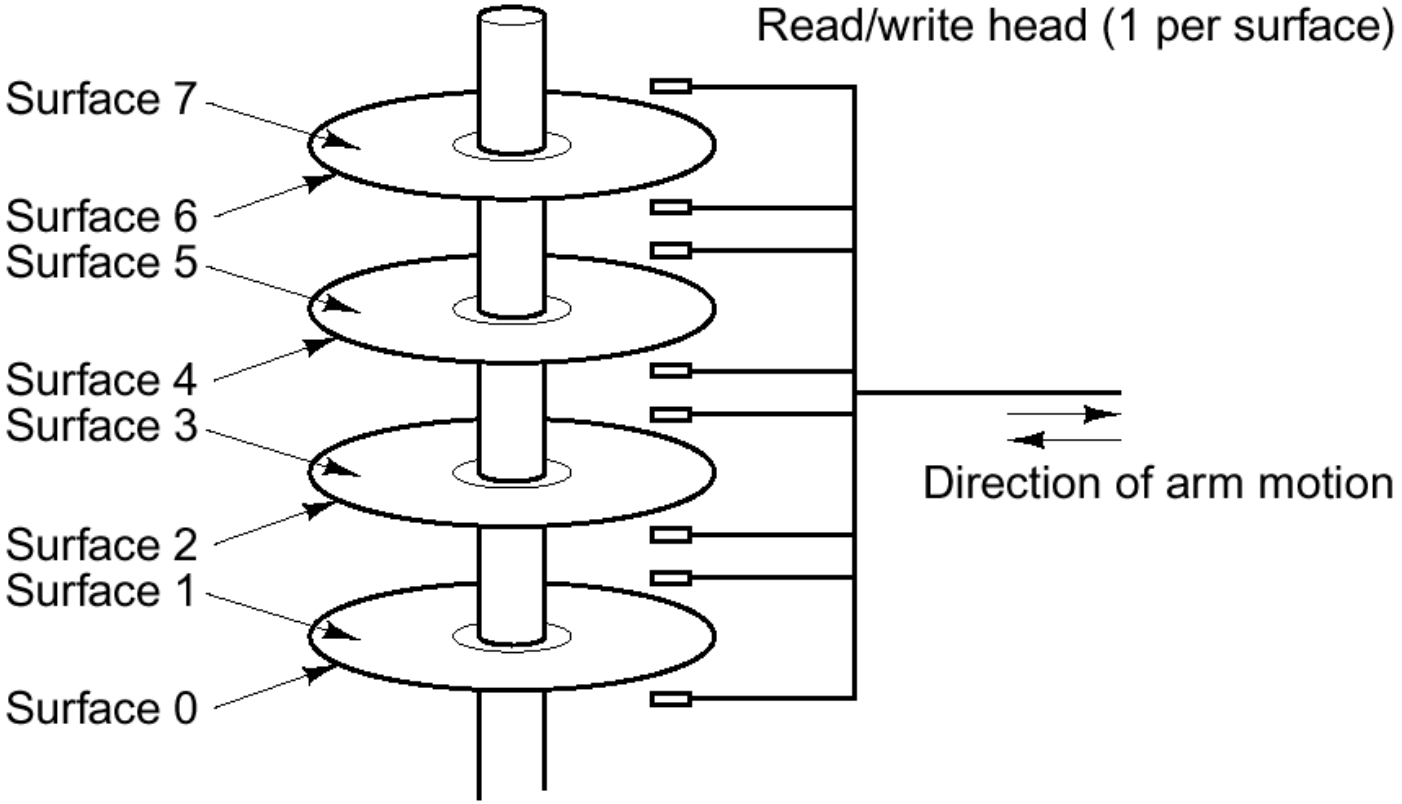
Dispositivo	Comportamento	Partner	Data rate (KB/sec)
Keyboard	input	human	0,01
Mouse	input	human	0,02
Voice input	input	human	0,02
Scanner	input	human	400
Voice output	output	human	0,60
Line printer	output	human	1
Laser printer	output	human	200
Graphics display	output	human	60.000
Modem	input or output	machine	2 - 8
Network/LAN	input or output	machine	500 - 6000
Floppy disk	storage	machine	100
Optical disk	storage	machine	1000
Magnetic tape	storage	machine	2000
Magnetic disk	storage	machine	2000 - 10.000

Impatto dell'I/O sulle prestazioni dei programmi

- L'impatto dell'I/O può essere alto
- Anche se i programmi non effettuano esplicitamente dell'I/O, le prestazioni possono essere influenzate dall'I/O a causa della memoria virtuale
    - caricamento iniziale delle pagine riferite, e page fault
- I sistemi più costosi (server) sono caratterizzati da sottosistemi di I/O più potenti
  - bus più larghi e veloci
  - controller più intelligenti
  - maggiore tolleranza ai guasti

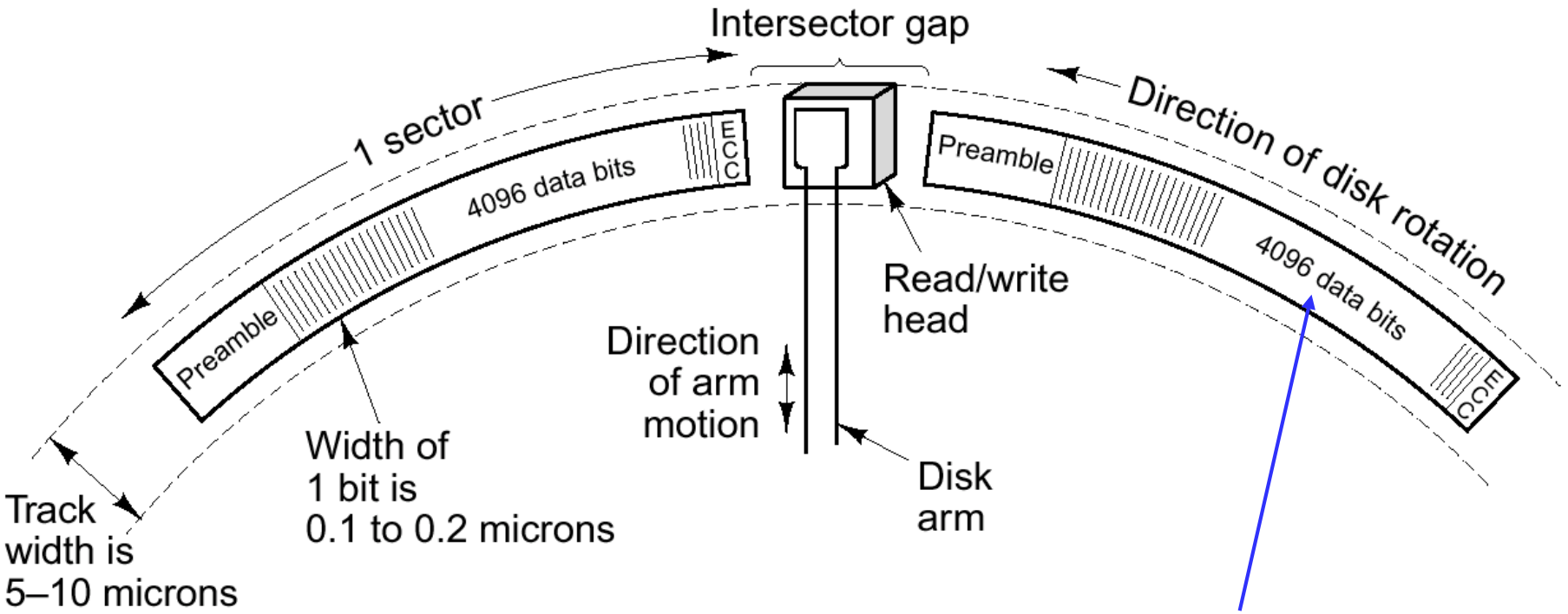
- Legge di Amdahl
  - Supponiamo che $T_{EXE} = T_{CPU} + T_{I/O} = 9/10\ T_{EXE} + 1/10\ T_{EXE}$
    - La CPU viene velocizzata di 5 volte, e T_{CPU} si riduce $\Rightarrow$ $1/5\ T_{CPU}$
    - Il sottosistema di I/O rimane inalterato ($T_{I/O} = 1/10\ T_{EXE}$ rimane costante)
  - Speedup = $T_{EXE} / (1/5 * (9/10 * T_{EXE}) + 1/10\ T_{EXE}) \cong 3.5$
  - Dopo l'ottimizzazione, il tempo di I/O diventa più importante, ovvero diventa una frazione maggiore del tempo totale di esecuzione

Disco



- **Piatti rotanti rivestiti di una superficie magnetica**
- **Più piatti, con una batteria di testine mobili**
- **Memoria *non volatile***
- ***Hard* e *Floppy***
- ***Diametri* piccoli, per rendere più efficace la dinamica di rotazione**

Caratteristiche tecniche di un disco



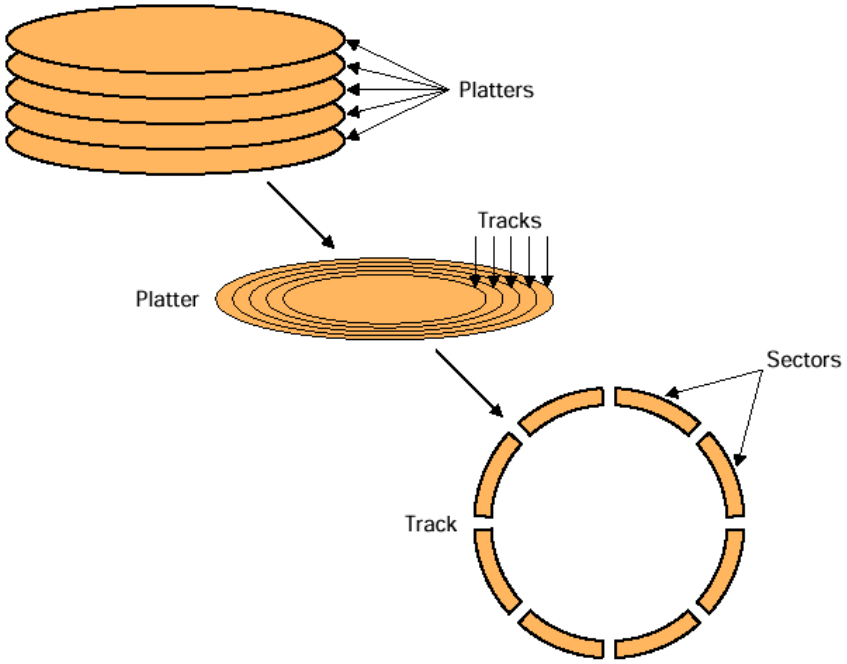
- **Piatti: 1-15**
- **Tracce: 10000-50000**
- **Settori per traccia: 100-500**
- **Dimensione settore: taglia tipica è 512 B**

Numero del settore, memorizzato assieme alla parte dati del settore stesso, e al codici correttori

- ***Cilindro*: questo numero definisce tutte le tracce sotto le varie testine**
- ***Indirizzo di un blocco*: Cilindro.Testina.Settore (CHS: Cilinder-Head-Sector)**

Disco

- **Per accedere i dati:**
  - **seek: posiziona le testine sul cilindro opportuno (da 3 a 14 ms avg.) - la traccia giusta sarà acceduta da una delle testine**
  - **latenza di rotazione: attendi il settore desiderato (0.5 / RPM)**
    - **RPM = da 5000 a 15000**
  - **transferimento dei dati: acquisisci i dati (uno o più settori) - da 10 a 40 MB/sec (M=10^6)**



Latenza di accesso ai dischi

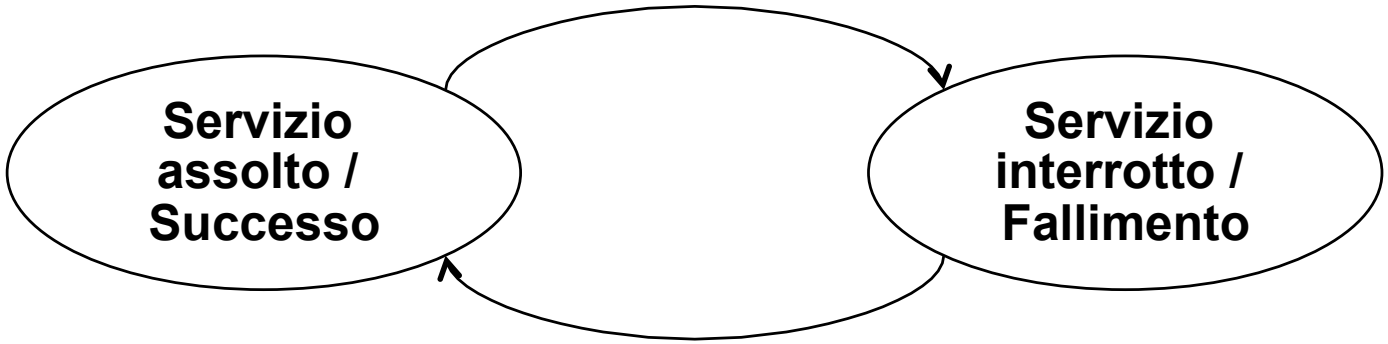
- Latenza = Costante + (Seek + Rotazione + Overhead controller) + Tempo trasferimento
- Tempo medio di seek
  - 3-14 ms, ma a causa della *località* questo tempo può essere estremamente inferiore (consideriamo un tempo di seek medio di 12 ms)
- Latenza di rotazione
  - RPM (Rotations Per Minute) = 1 / Tempo di rotazione in min.
    - 3600-5400-7200-10000 RPM (consideriamo 5400 RPM)
  - Latenza di rotazione = 0.5 * Tempo di rotazione in min. = 0.5 / RPM = 0.5 / 5400 min. = 60 * 0.5 / 5400 s. = 5.6 ms
- Overhead del controller = 2 ms
- Tempo di trasferimento
  - 5-10 MB/sec
  - Considerando di dover trasferire un settore di 512 B, con un rate di 5 MB/s
    - Size/Banda = 0.5 KB / 5 MB/s = 0.1 ms

- Latenza = 12 + 5.6 + 2 + 0.1 = 19.7 ms

Dependability

- Gli utenti chiedono di avere sistemi e sottosistemi *dependable (affidabili)*
- In particolare questo è vero per i sistemi di I/O
- *Sistema dependable*
  - Con un ottima qualità del servizio
  - Il cui comportamento è quello specificato
  - I cui fallimenti, ovvero i momenti in cui il cui comportamento devia da quello specificato e corretto, siano rari

- I sistemi passano tra questi due stati
  - Un sistema con un alto grado di *dependability* cade raramente nello stato *Fallimento*, o vi trascorre poco tempo nello



Dependability

- **Misure quantitative**
  - **Reliability (affidabilità)**
  - **Availability (disponibilità)**

- **MTTF (Mean Time To Failure) è la misura quantitativa usata per la *Reliability***
  - **Ad esempio, per i dischi, MTTF espresso in ore: 600.000 @ 25° C**

- **L' *Availability* misura per la percentuale del tempo in cui il sistema si trova nella stato di successo (servizio assolto)**

- **Abbiamo bisogno di definire il tempo per riparare il guasto, ovvero per transire dallo stato di *fallimento* a quello di *successo***
  - **MTTR (Mean Time To Repair)**

$$\textit{Availability} = \frac{\text{MTTF}}{(\text{MTTF} + \text{MTTR})}$$

Dependability

- **Per incrementare la *reliability* (MTTF), è possibile**
  - **Migliorare la qualità dei componenti ➔ *fault avoidance***
  - **Progettare il sistema in modo che continui ad operare in presenza di fallimenti (anche se in alcuni casi con prestazioni ridotte) ➔ *fault tolerance***

- **Nota che, se il sistema è correttamente progettato per essere f*ault tolerant*, un *fault* di un componente non porta al *failure* del sistema completo**

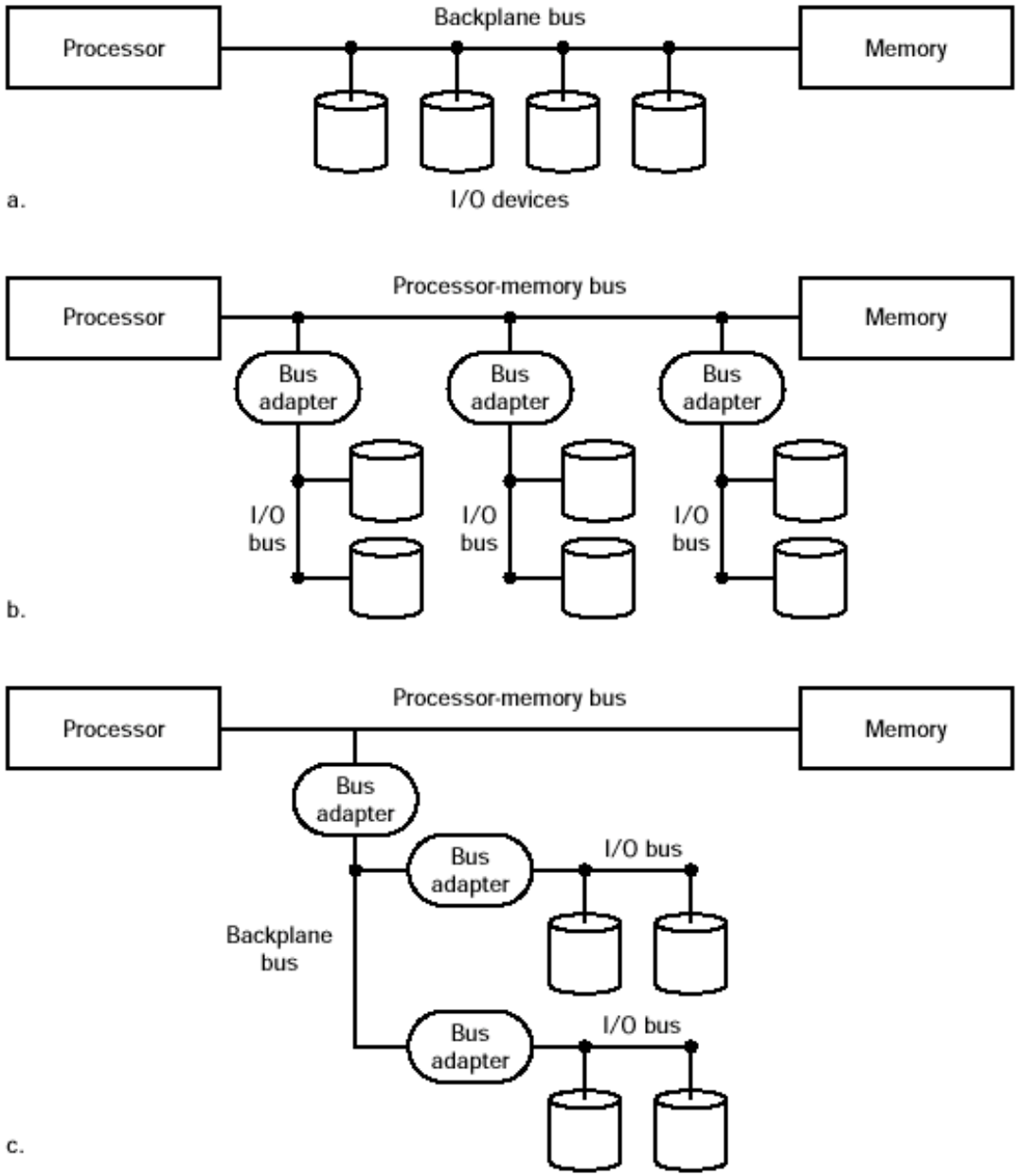
BUS

- **I bus forniscono l'interconnessione tra**
  - **Processori**
  - **Memoria**
  - **I/O**
- **Bus**
  - **canale di comunicazione CONDIVISO**
    - **uso esclusivo, e quindi conflitti nell'uso**
    - **collo di bottiglia del sistema**
  - **il bus è composto da un fascio di fili/linee**
    - **fili per il controllo**
      - richiesta di un'operazione di I/0
      - acknowledge della richiesta
    - **fili per i dati**
      - dati per effettuare la richiesta di op. di I/O (es. indirizzo di memoria, dati da scrivere)
      - dati per trasferire i risultati della richiesta di I/O
      - possono anche esserci fili separati per i dati e gli indirizzi (separazione utile per le operazioni di scrittura in memoria, ovvero per l'input)

Esempi di BUS

- **Processore-Memoria (System bus)**
  - **corto, alta velocità, proprietario**
- **Backplane**
  - **lunghi, mediamente veloci, standard (es. PCI)**
  - **collegano dispositivi anche diversi (con bande di trasferimento diverse)**
- **I/O**
  - **bus standard come SCSI, che gestisce una catena di dischi**
  - **più lenti dei bus precedenti, ma più lunghi**

Esempi di gerarchie di bus



- **a) vecchi PC**
- **b) bus di I/O interfacciati direttamente con il bus processore memoria**
- **c) gerarchia completa di bus. E' l'approccio usato oggi più comunemente per i sistemi di fascia medio/ alta**
  Con questo schema abbiamo poche connessioni sul bus ad *alta velocità processore-memoria*

Transazioni sul Bus

- La memoria è sempre “in attesa” di ricevere richieste di lettura/scrittura dal processore o dal controllore DMA
- Il processore o il controllore DMA sono in grado di iniziare un’operazione di I/O, iniziando una transazione sul bus

- Supponiamo che il controllore/dispositivo sia in grado di acquisire al bus, e di effettuare un accesso alla memoria. La transazione di I/O è la seguente:
  - il dispositivo invia una richiesta di I/O alla memoria, specificando il tipo (Read/Write) di accesso, con l’indirizzo della memoria

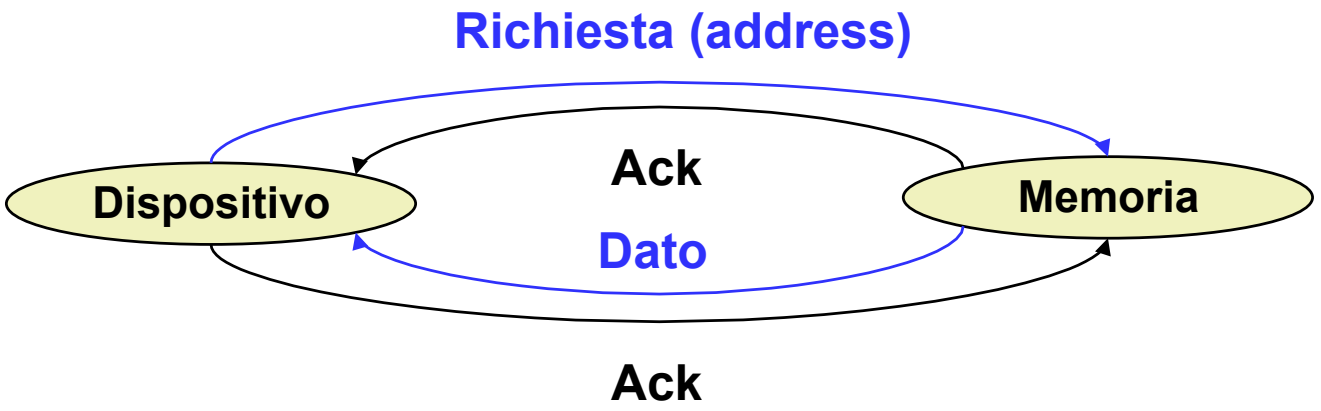
  - Write (Input): il dispositivo INVIA successivamente (o assieme all’indirizzo, se il bus ha abbastanza fili) i dati da scrivere in memoria
  - Read (Output): la memoria INVIA i dati letti al dispositivo

Bus sincroni e asincroni

- **Bus sincrono**
  - **linea di clock condivisa tra gli utenti del bus**
  - **il protocollo che implementa la transazione sul bus sfrutta l'esistenza del clock comune**
    - **Es.: al ciclo x il dispositivo pone sul bus una richiesta di lettura alla memoria, al ciclo x+Δ il dispositivo può leggere i dati, posti sulle linee dati del bus dalla memoria (Δ dipende dalla velocità della memoria)**
  - **uso limitato a bus corti, come quello *proprietari* processore-memoria**
  - **fenomeno del clock skew (disallineamento del segnale di clock) con bus lunghi**

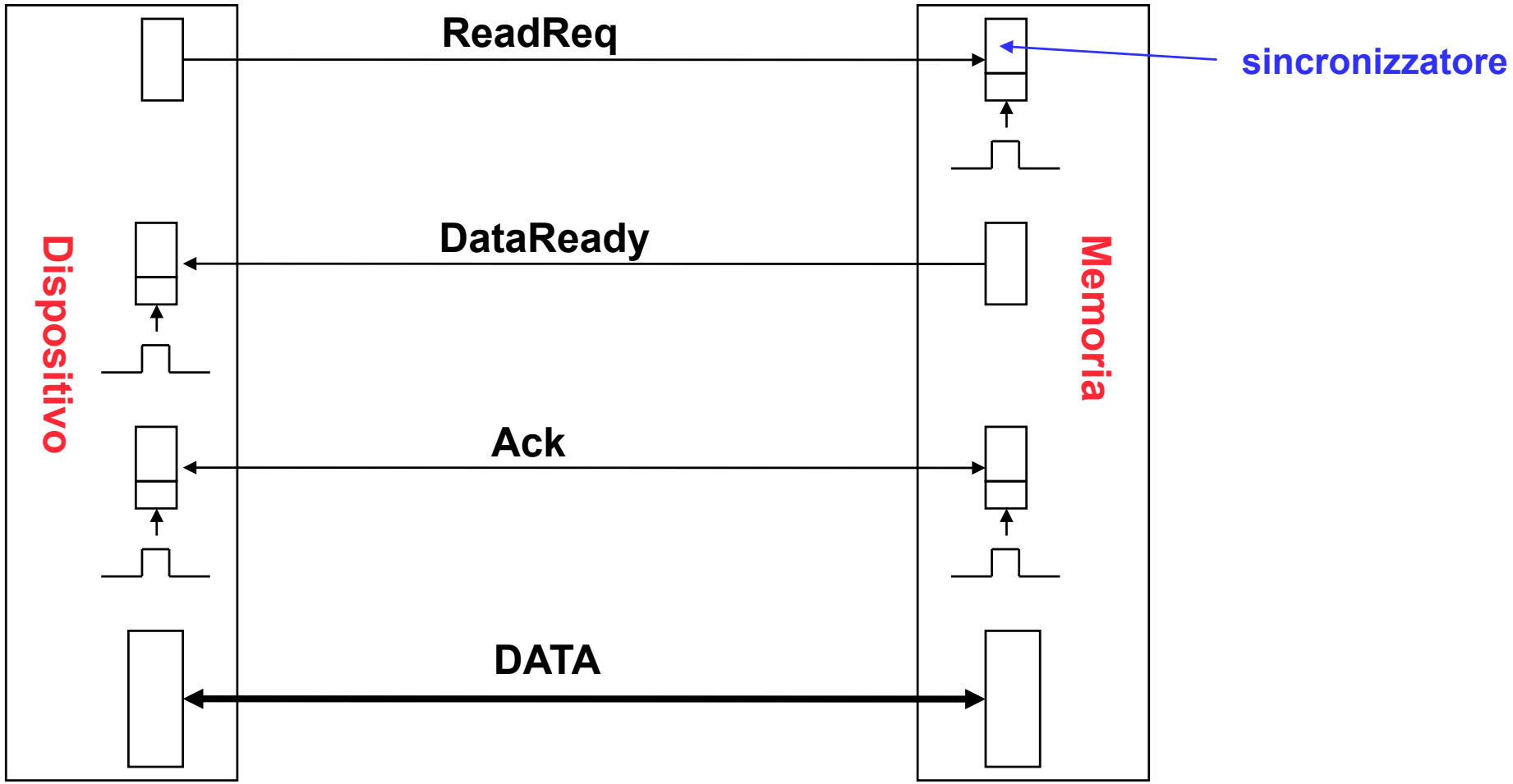
- **Bus asincrono**
  - **sono asincroni i bus *standard*, che possono ospitare periferiche con velocità differenti**
  - **Il protocollo di comunicazione per effettuare una transazione di I/O richiede una sincronizzazione (*handshaking*)**

Transazione di Output su bus asincrono



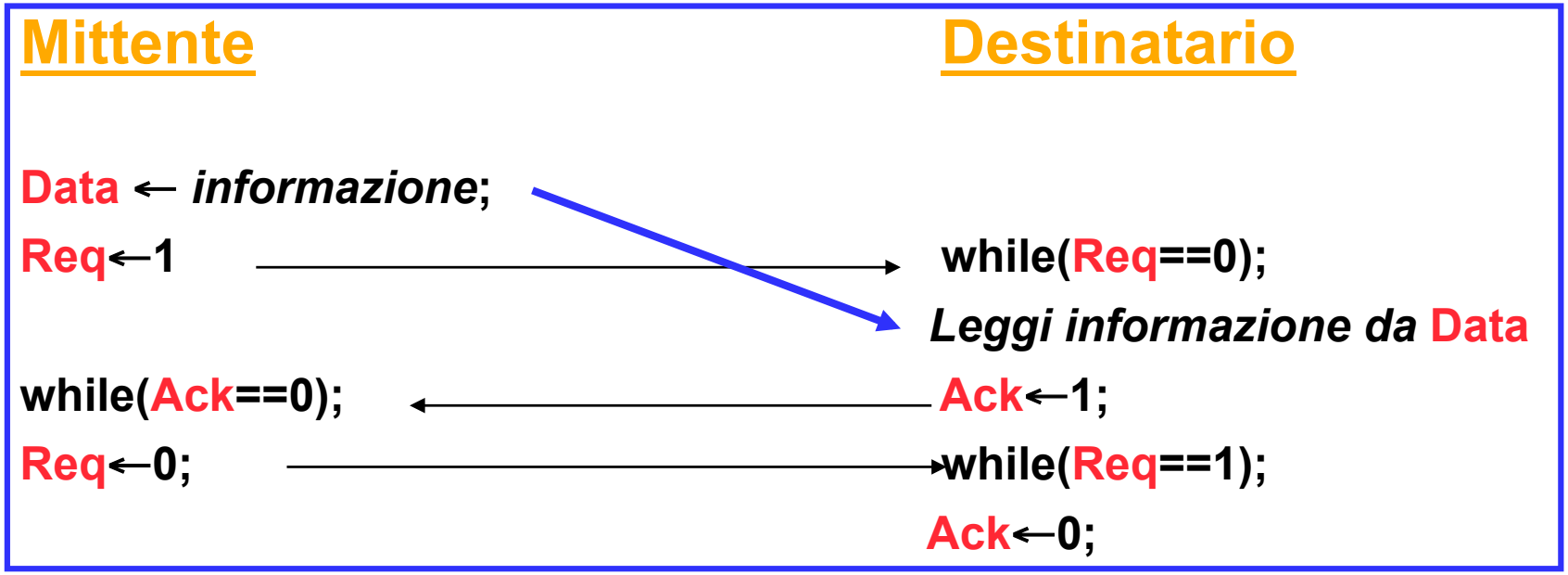
- **Per le comunicazioni vengono usati linee diverse del bus**
  - **linee dati (indirizzi e dati)**
  - **linee controllo (richieste, acknowledge)**

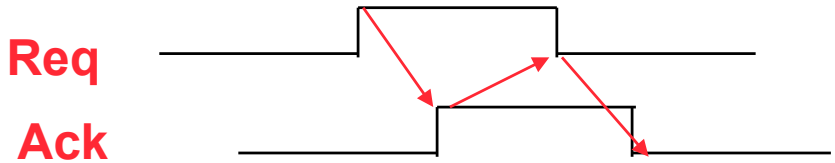
Transazione di Output sul bus asincrono



- ReadReq e DataReady servono per segnalare al partner che sulle linee DATA è presente un'informazione significativa
  - Indirizzo o Dati
- Ack viene usato per l'handshaking

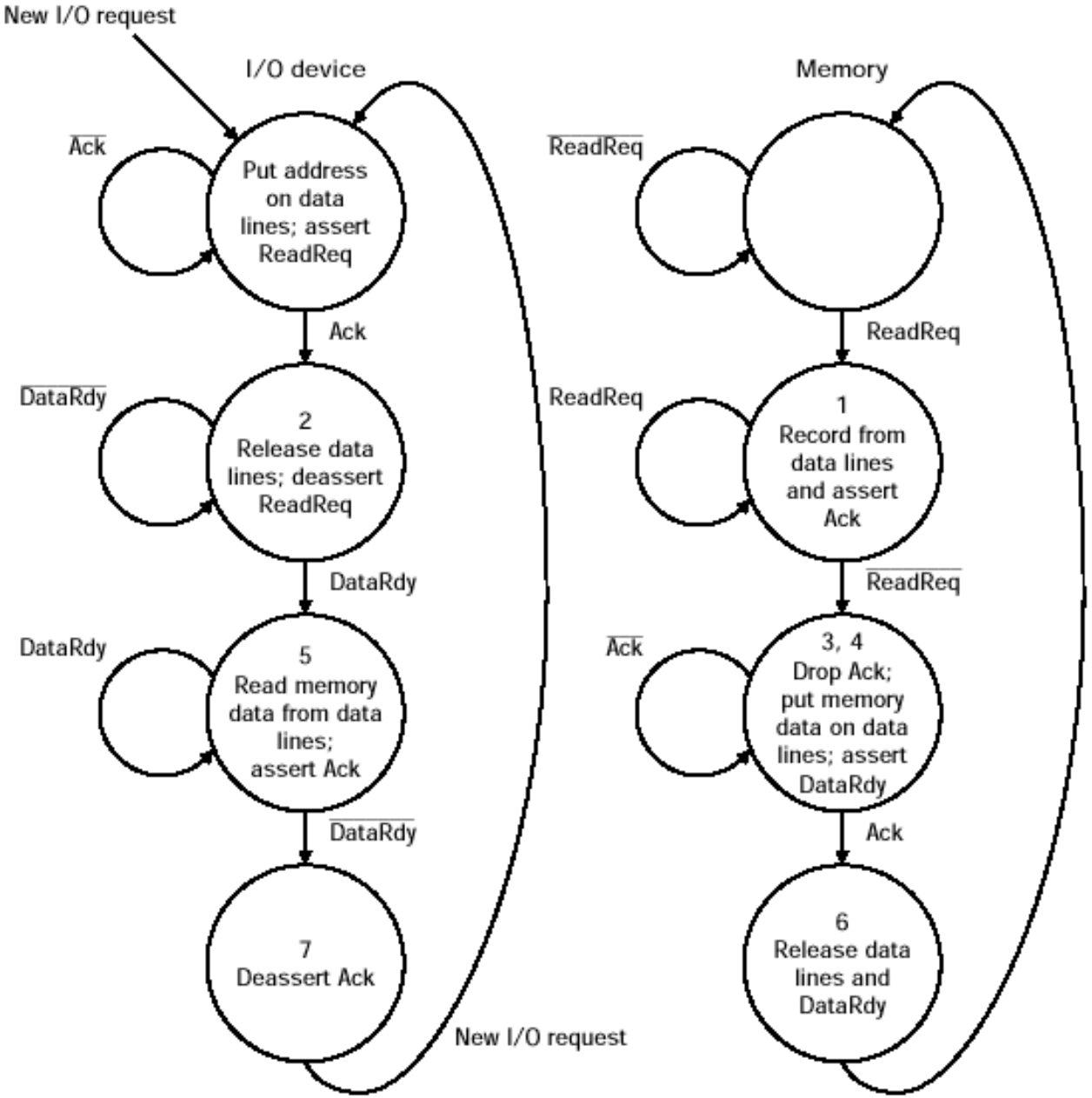
Half-Handshaking





- All'inizio Req e Ack sono uguali a 0, e tornano entrambi a 0 al termine dell'handshaking
- Lo schema precedente di scambio di messaggi è valido
  - sia per Mittente= Dispositivo e Destinatario=Memoria
  - sia per Mittente=Memoria e Destinatario=Dispositivo

Handshaking completo



Operazione di Output, iniziata da un dispositivo di I/O

- **il dispositivo fa la richiesta alla memoria, fornendo l'indirizzo**
- **la memoria risponde con il dato**
- **Per ogni comunicazione**
  disp → mem
  mem → disp
  si ha un half-handshaking

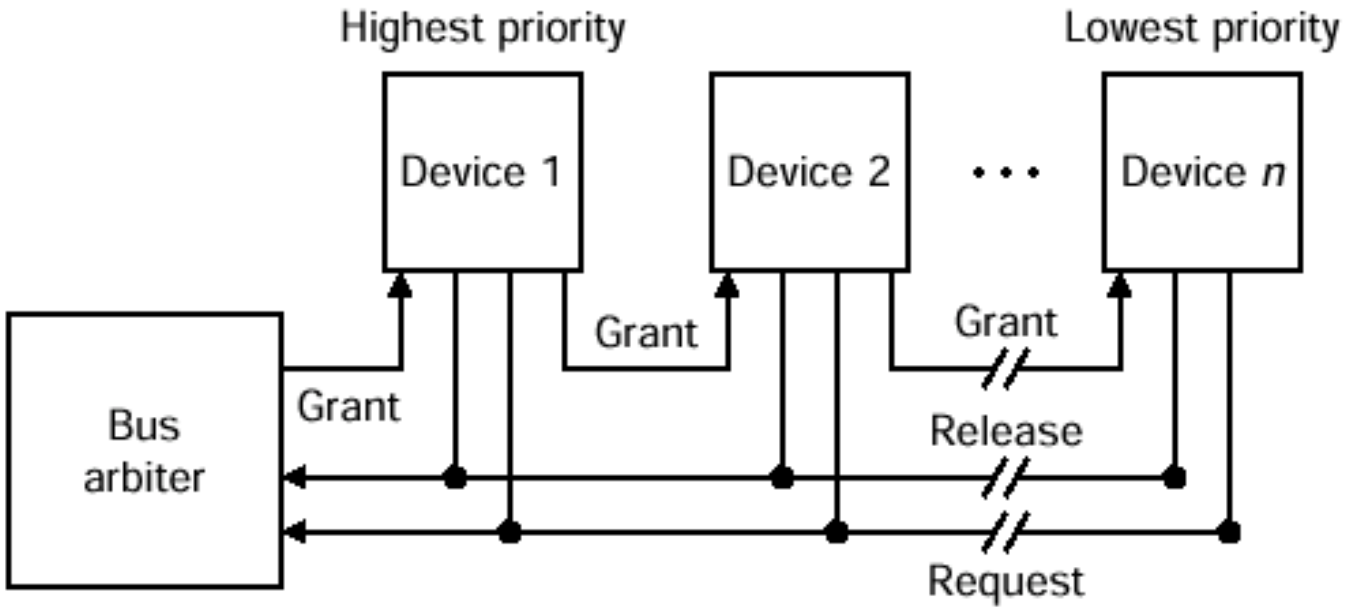
Arbitraggio bus

- **Bus condiviso**
- **Se, oltre il processore, altri dispositivi possono iniziare le transazioni sul bus, si rende necessario un arbitraggio del bus per evitare conflitti**
  - **gli attori che possono iniziare una transazione sul bus sono detti *master* del bus**

- **I vari attori che competono per l'uso del bus hanno quindi *linee aggiuntive* che li collegano con un arbitro del bus**
  - **richiesta - assegnazione - rilascio**

- **Problemi tipici dell'arbitraggio**
  - **centralizzazione / distribuzione dell'arbitro**
  - **priorità in caso di richieste concomitanti**
  - ***fairness* (equità), per evitare che una richiesta rimanga indefinitamente non soddisfatta a causa dell'arrivo di altre richieste**

Arbitri



- **Arbitro daisy chain centralizzato, dove i richiedenti sono posti in catena in funzione della priorità**
  - **problema: fairness**

- **Altri tipi di arbitri**
  - **centralizzato, con linee *multiple* di richiesta / rilascio**
  - **completamente distribuiti (usato in reti LAN come Ethernet, che per l'appunto non è altro che un bus)**

Invio di comandi di I/O

- Il processore deve poter comunicare con i controller dei dispositivi
  - leggere/scrivere i registri dei dispositivi
  - scrittura: dati in output + comandi
  - lettura: dati in input + stato [busy/idle/down]

- Due metodi per comunicare con i dispositivi
  - Memory mapped I/O
    - registri dei dispositivi visti come locazioni di memoria speciali
    - il processore vi accede con semplici *load / store*
    - la MMU si incarica di indirizzare tali richieste verso il dispositivo, invece che verso la memoria
  - Istruzioni speciali
    - l'ISA del processore comprende istruzioni particolari per effettuare l'I/O

Invio di comandi di I/O e protezione

- Il processore, per ragioni di protezione, dovrà potere iniziare un'operazione di I/O solo in modalità kernel
  - questo per permettere solo alle routine del SO di effettuare l'I/O

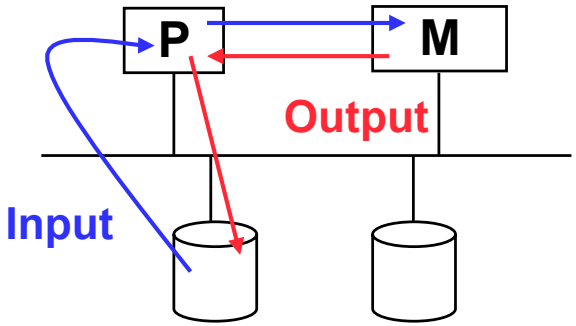
- Memory mapped I/O
  - gli indirizzi usati sono "speciali"
  - aree di spazio indirizzabile deve essere riservato per l'I/O invece che per la memoria
  - il sistema di Memoria Virtuale impedisce (generando un'eccezione) la traduzione di tali indirizzi se il programma è in esecuzione in modalità utente (indirizzi non presenti nello spazio di indirizzamento virtuale dell'utente)
- Istruzioni speciali
  - sono istruzioni eseguibili solo in modalità kernel, altrimenti si ha un'eccezione

Transazioni di I/O sul bus

- **Input:** dal DISPOSITIVO alla MEMORIA
- **Output:** dalla MEMORIA al DISPOSITIVO
- **Programmed I/O (polling)**
  - il processore è sempre coinvolto
  - INPUT: legge il dato dal dispositivo e lo scrive in memoria
  - OUTPUT: legge il dato dalla memoria e lo scrive sul dispositivo
  - il processore deve anche ***sondare*** (polling) i registri del dispositivo per controllare se il dispositivo ha *terminato* e/o è *pronto*
- **Interrupt-driven I/O**
  - rispetto al caso precedente, il dispositivo segnala al processore i segnali di *terminazione* e *pronto* (tramite interruzioni)
- **DMA (Direct Memory Access)**
  - il dispositivo ha la capacità (tramite il controller DMA) di accedere autonomamente alla memoria senza disturbare il processore
  - l'intervento del processore è limitato: si tratta di fornire al controller DMA informazioni su quanti dati leggere/scrivere (e dove)

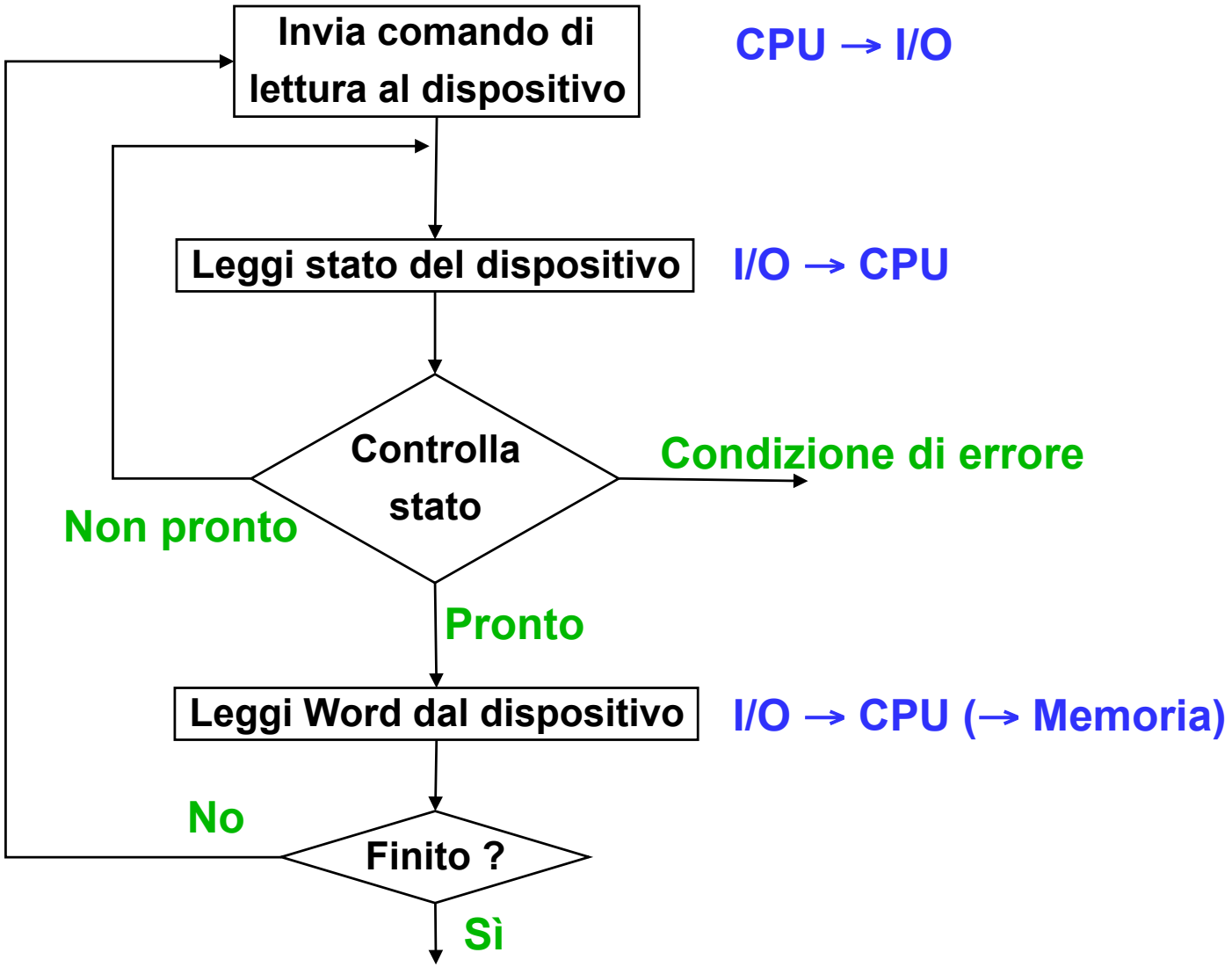
Tecniche di I/O

- **Programmed I/O (polling)**
  - **il processore è coinvolto in tutti i singoli trasferimenti**
    - **nota che per trasferire un blocco di dati sono necessari diverse transazioni sul bus**
  - **il dispositivo non ha modo di leggere/scrivere direttamente in memoria**
  - **polling: il processore controlla (*sonda*) periodicamente se il dispositivo ha terminato una certa operazione di I/O**
  - **L'overhead di ogni controllo è relativamente piccolo, ma può diventare difficile *fissare a priori la frequenza del polling del dispositivo***
    - **se molti *polling* vanno a vuoto, l'overhead diventa consistente**
  - **Modalità usabile per dispositivi lenti, come i mouse**



Programmed I/O (polling)

- Esempio di input di un blocco

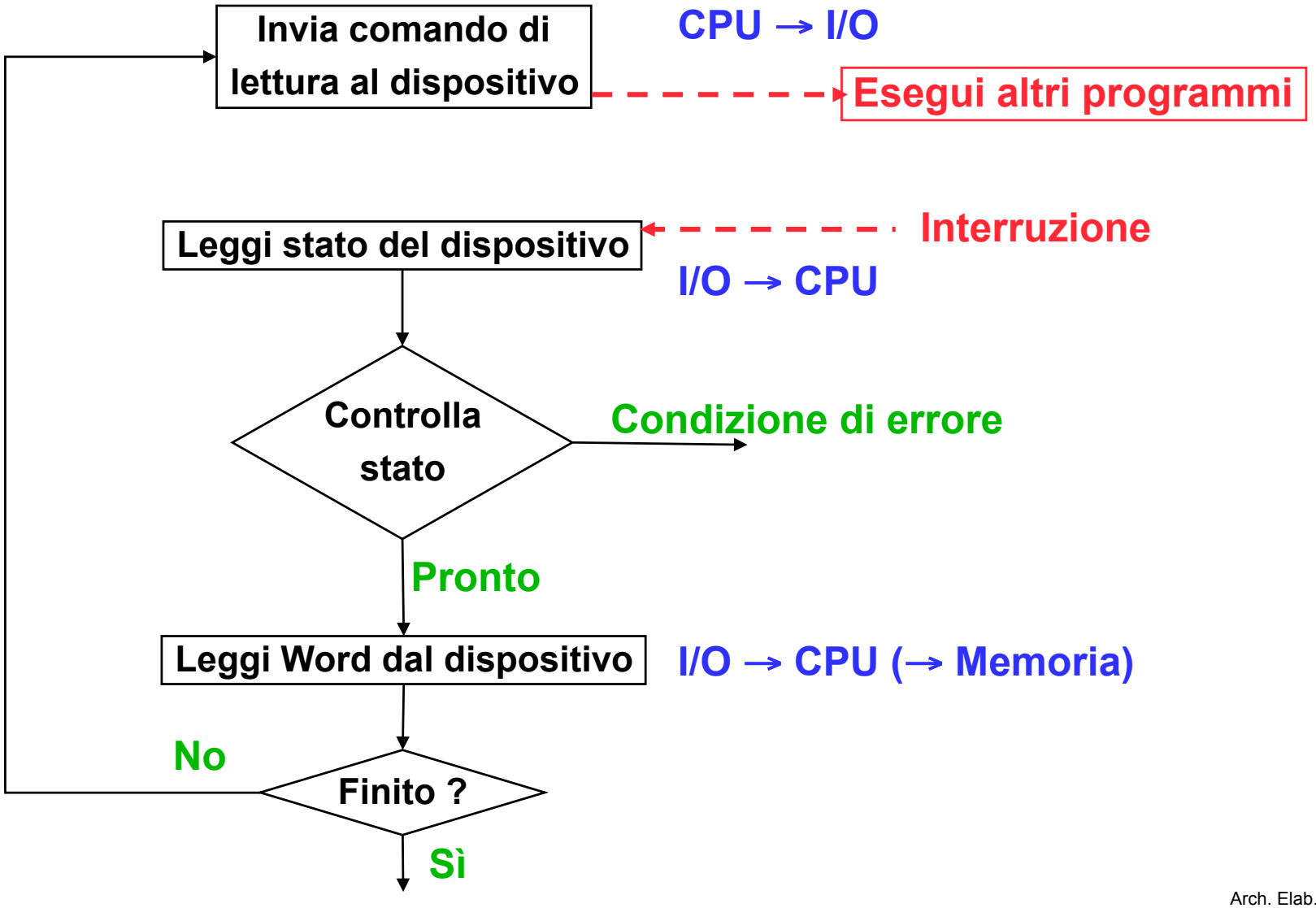


Tecniche di I/O

- **Interrupt-driven I/O**
  - **il polling è costoso, soprattutto se il processore è costretto a ciclare inutilmente finché il dispositiva non diventa pronto**
  - **invece di ciclare effettuando il polling, il processore può impegnare più proficuamente il proprio tempo eseguendo altri programmi**
  - **il dispositivo segnalerà al processore gli eventi importanti tramite interruzioni asincrone (ad esempio, un'interruzione viene inviata per segnalare la terminazione di un'operazione di I/O)**
  - **gestione interruzioni effettuata da una routine del SO (interrupt handler)**
    - **costo della singola interruzione anche maggiore del costo del polling**
    - **costo dovuto**
      - **al salvataggio/ripristino totale o parziale dello stato del programma in esecuzione, ed**
      - **all'individuazione dell'interruzione da servire (possono giungere più interruzioni, con diversa priorità)**

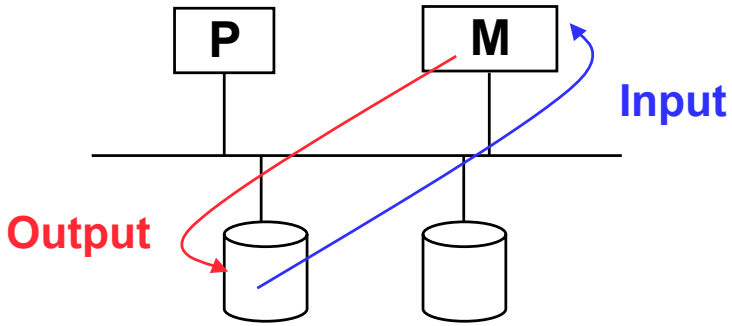
Interrupt-driven I/O

- **Esempio di input di un blocco**

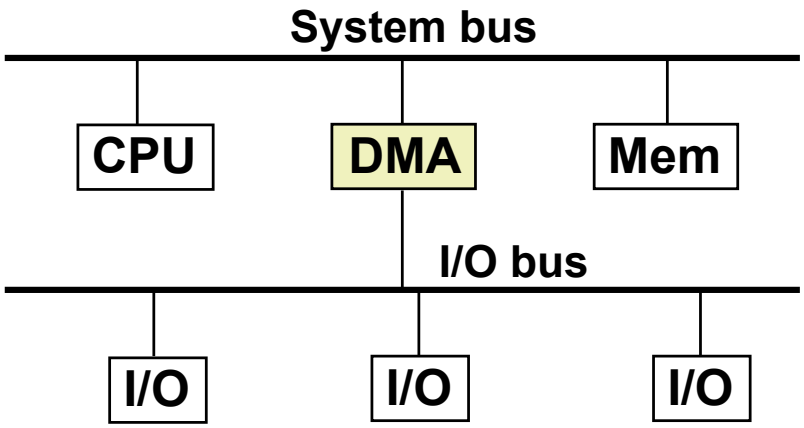


Tecniche di I/O

- **DMA (Direct Memory Access)**
  - le tecniche di *programmed I/O (polling)* e di *interrupt-driven I/O* costringono il processore ad intervenire per i trasferimenti di ogni singola word di un blocco di dati !!
  - per evitare di coinvolgere il processore, è stata introdotta la tecnica del DMA (Direct Memory Access)
  - è necessario un nuovo modulo di DMA, che fa le veci del processore per i trasferimenti di I/O
  - il processore si limita solo a programmare il controllore DMA, che autonomamente porta avanti la transazione
    - al termine del trasferimento, il modulo DMA invia un'interruzione al processore



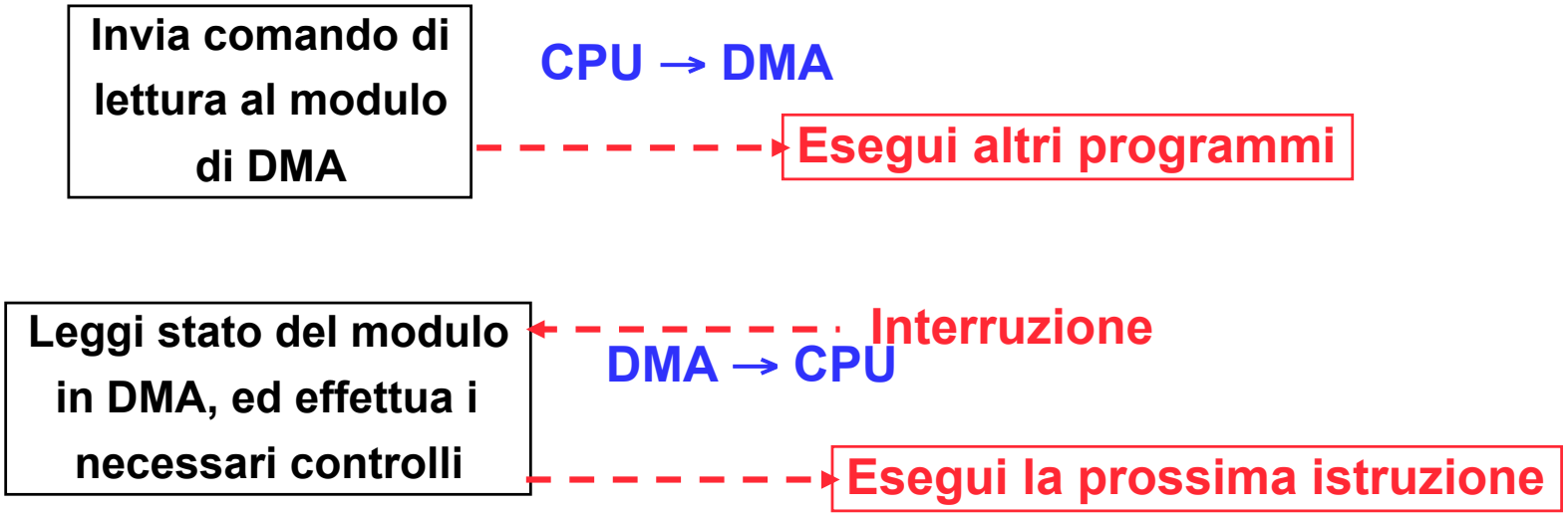
Trasferimento tramite DMA senza coinvolgere la CPU



Esempio di configurazione di DMA

DMA

- Esempio di input di un blocco



- CPU coinvolta solo all'inizio e alla fine del trasferimento
- CPU invia informazioni al modulo DMA su
  - tipo di trasferimento (read/write) e indirizzo I/O device
  - indirizzo iniziale in memoria e numero di Byte da trasferire
- Modulo DMA trasferisce una word per volta, mentre il processore effettua altri programmi

Polling vs Interrupt

- Overhead polling: 400 cicli con processore a 500 MHz
- MOUSE
  - tipico dispositivo lento, che può tranquillamente essere sondato raramente, senza che siano persi dati
  - frequenza di polling: 30 volte al s.
  - Overhead tot. al s. = 30 * 400 / (500 M) s. = 30 * 400 / (500 K) ms. = 0.024 ms.
  - Frazione del tempo durante il quali il processore viene usato per il polling:
    0.024 / 1000 = 0.000024 = 0.0024 %
  - Overhead accettabile
- FLOPPY DISK
  - trasferimenti *continui* in unità da 16 b, con banda di trasferimento di 50 KB/s
  - frequenza di polling: 50 KB / 2 B = 25 K al s.
  - Overhead tot. al s. = 25 K * 400 / (500 M) s. = 25 K * 400 / (500 K) ms. = 20 ms.
  - Frazione del tempo durante cui il processore viene usato per il polling:
    20 / 1000 = 0.02 = 2 %
  - Overhead accettabile

Polling vs Interrupt

- **HARD DISK**
  - **trasferimenti *continui* in unità di 4 word, con banda di trasferimento di 4 MB/s**
  - **frequenza di polling: 4 MB / 16 B = 250 K al s.**
  - **Overhead tot. al s. = 250 K * 400 / (500 M) s. = 400/2K s. = 200 ms.**
  - **Frazione del tempo durante cui il processore viene usato per il polling: 200 / 1000 = 0.2 = 20 %**

- **L'hard disk che in continuazione trasferisce dati (in piccoli blocchi) costituisce un grosso problema per il processore, che rimane impegnato per il 20% del tempo per operazioni di I/O**

- **Può essere utile utilizzare il meccanismo dell'interrupt in questo caso ?**
  - **Il meccanismo dell'interrupt è più costoso del polling**
  - **Se l'interrupt costa 500 cicli (invece dei 400 per il polling) e il disco trasferisce continuamente (al massimo della banda)**

    ⇒ overhead totale maggiore di quello del polling (25 %)

Polling vs Interrupt

- **Interrupt utile quando l'hard disk non trasferisce in continuazione, ma solo per una frazione del tempo**
  - **situazione plausibile, perché il disco potrebbe non essere in grado di trasferire in continuazione, ma potrebbe dover via via spostare la testina !!**
- **Supponiamo che il disco trasferisca dati solo per il 5% del tempo**
  - **trasferimenti in unità di 4 word, con banda di trasferimento di 4 MB/s**
  - **Overhead interrupt: 500 cicli con processore a 500 MHz**
- **Frequenza massima di trasferimento: 4 MB / 16 B = 250 K trasf. al s.**
- **Poiché il disco lavora solo per il 5% del tempo**
  - **Frequenza reale di trasf.: 5% (250 K) trasf. al s. = 12.5 K trasf. al s.**
  - **No. di cicli sprecati per gli interrupt: 12.5 K * 500 = 6.25 M cicli al s.**
  - **Overhead totale al s. = 6.25 M / (500 M) s. = 0.0125 s. = 12.5 ms.**
  - **Frazione del tempo durante cui il processore viene usato per gestire le interruzioni:**
    12.5 / 1000 = 0.0125 = 1.25 %

Trasferimento in DMA

- **Se il disco trasferisce in continuazione, e dobbiamo trasferire grossi blocchi di dati, l'unica soluzione è effettuare il trasferimento tramite DMA**

- **Disco trasferisce *continuamente* con banda di trasferimento di 4 MB/s**
- **Blocchi trasferiti tramite DMA di dimensione: 8 KB**
- **Frequenza processore: 500 MHz**
- **Inizializzazione DMA : 1000 cicli**
- **Overhead interrupt: 500 cicli**

- **Numero di trasferimenti al secondo: 4 MB / 8 KB s. = 0.5 K trasf. al s.**
- **No. di cicli sprecati per attivazione/finalizzazione di ogni singolo trasferimento tramite DMA: 1000 + 500 cicli**
- **Overhead tot. al s. = 1500 * no. trasf. / Freq. clock = 1500 * 0.5 K / (500 M) s. = 750 / (500 K) s. = 750 / 500 ms. = 1.5 ms.**
- **Frazione del tempo durante cui il processore viene usato per gestire il DMA:**

  1.5 / 1000 = 0.0015 = 0.15 %