Giuseppe Pelagatti

Programmazione e Struttura del sistema operativo Linux

Appunti del corso di
Architettura dei Calcolatori e Sistemi Operativi (AXO)

Parte IO: Input/Output e File System

cap. IO1 – Input/Output a livello Hardware

IO1. Input/Output a livello Hardware

1.Accesso alle periferiche da parte del processore

Istruzioni di Input/Output

Per svolgere funzioni utili il processore deve poter interagire con le periferiche del sistema. A questo scopo molti processori utilizzano delle istruzioni macchina specializzate, ad esempio **IN** e **OUT**, che fanno riferimento agli indirizzi dei registri delle periferiche; tali indirizzi sono detti **Port**. Tramite l'esecuzione delle istruzioni IN e OUT è quindi possibile leggere e scrivere tali registri.

Registri delle periferiche

Ogni periferica possiede alcuni registri che servono alla sua gestione; alcuni registri, detti **registri dati**, servono a contenere i dati che la periferica deve leggere o scrivere, altri registri, detti di **registri di controllo e stato**, servono a contenere delle indicazioni sulle operazioni che la periferica deve svolgere oppure sullo stato della periferica.

In particolare, nel registro di stato esiste un bit detto **Ready** che indica che la periferica è "pronta" (oppure, se Ready=0, che è "occupata").

Si tenga presente che il significato del bit Ready di una periferica è relativo alla possibilità, per il processore, di svolgere un'operazione di lettura o scrittura di un dato. Pertanto una periferica di ingresso, come una tastiera, è in stato di pronto quando un tasto è stato premuto e quindi esiste un carattere nel suo registro dati che il processore può leggere; una periferica di uscita, come una stampante, è pronta se può ricevere un dato dal processore per stamparlo.

Tipicamente una stampante funziona nel modo seguente (ovviamente questa è una stampante teorica, governata un carattere alla volta; le stampanti reali attualmente sono dotate di una loro memoria capace di ricevere molti caratteri alla volta):

1. appena "accesa" Ready=1, registro dati vuoto
2. quando la CPU scrive un dato nel registro dati Ready va a 0
3. quando la stampante ha finito di stampare il dato e il registro dati è nuovamente vuoto, Ready torna a 1

Una tastiera invece funziona nel modo seguente:

1. appena "accesa" Ready=0, registro dati vuoto
2. quando viene premuto un tasto, Ready va a 1
3. quando la CPU legge il dato, Ready torna a 0

Funzionamento a Controllo di programma

Il modo più semplice per gestire una periferica è detto a **controllo di programma**. Anche se questa modalità di gestione è inadeguata per essere applicata dal sistema operativo, è istruttivo vedere come funziona; lo schema è il seguente:

1. leggi il registro di stato della periferica (IN sul Port del registro di stato)
2. se Ready=1, procedi, altrimenti ritorna al passo 1
3. esegui l'operazione voluta (ad esempio, lettura di un dato dalla tastiera oppure invio di un dato alla stampante) (IN oppure OUT sul Port del registro dati)

Ad esempio, in figura 1 è illustrata la sequenza di istruzioni di ingresso/uscita, cioè di interazioni tra il processore e una stampante durante l'esecuzione di un programma di stampa di un carattere "A" sulla stampante. Si noti che il processore è obbligato a restare nella situazione iniziale fino a quando la stampante non cambierè stato.

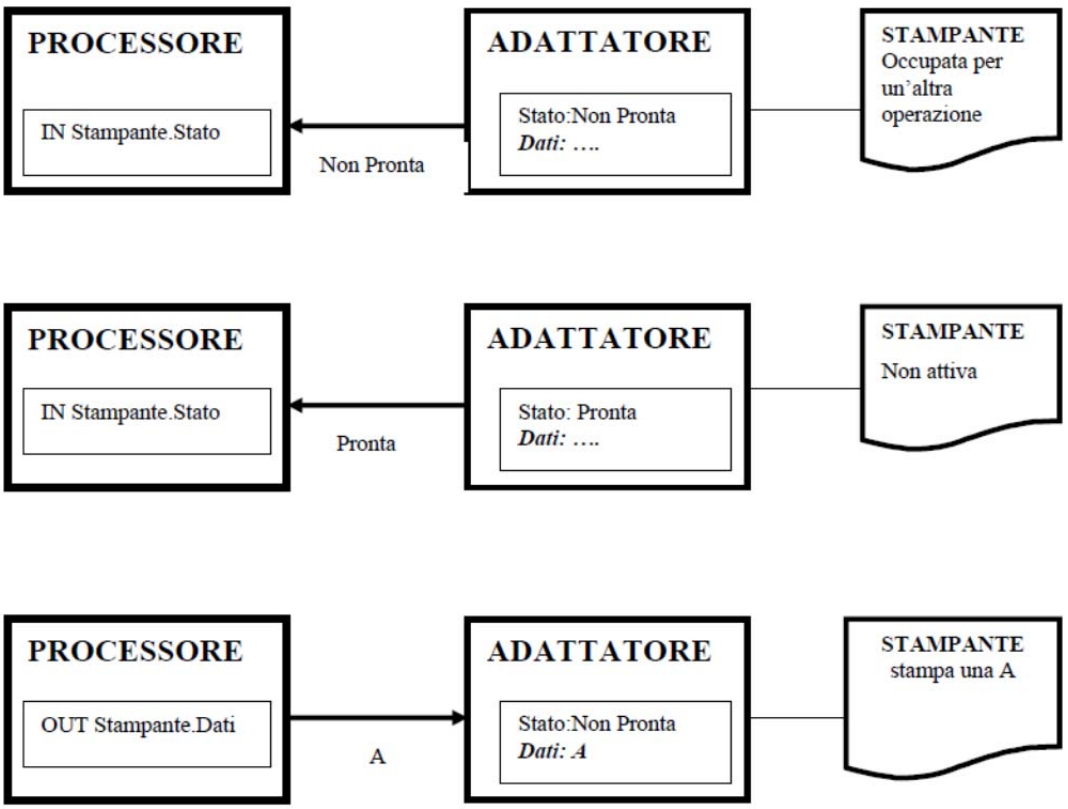


Figura 1 – Stampa di una "A" a controllo di programma

Adattatori

La figura mostra inoltre che è opportuno distinguere tra l'**adattatore** della stampante, che interagisce con il processore e contiene i registri di stato e dati, e la stampante vera e propria.

Dal punto di vista della terminologia, esiste molta confusione in questo campo e in molti testi e documentazioni di sistemi vengono utilizzati termini come interfaccia, controllore o anche semplicemente scheda (board) al posto del termine adattatore. In questo testo utilizzeremo sempre il termine adattatore.

Gli adattatori accoppiano i diversi tipi di CPU con i diversi tipi di periferica; in questo modo non è necessario costruire periferiche diverse per i diversi tipi di CPU – è sufficiente costruire diversi adattatori.

2. Gestione delle Periferiche e Meccanismo di Interrupt

La gestione delle periferiche a controllo di programma ha il difetto di obbligare il processore a restare in un ciclo di attesa che la periferica diventi pronta; nel Sistema Operativo invece vogliamo che il processore ponga il processo in attesa e passi ad eseguire altre attività fino a quando la periferica non diventa pronta.
Questo obiettivo viene ottenuto utilizzando il meccanismo di interrupt e richiede che una periferica segnali tramite un apposito interrupt il passaggio da occupata a pronta, ovvero la transizione del bit Ready da 0 a 1.
In figura 2 questo meccanismo è applicato alla stampa di un carattere A sulla stampante; la figura illustra un possibile svolgimento temporale degli eventi. La differenza fondamentale rispetto allo svolgimento di figura 1 consiste nell'assenza del ciclo di attesa iniziale; il processore viene impegnato solamente per svolgere funzioni utili.

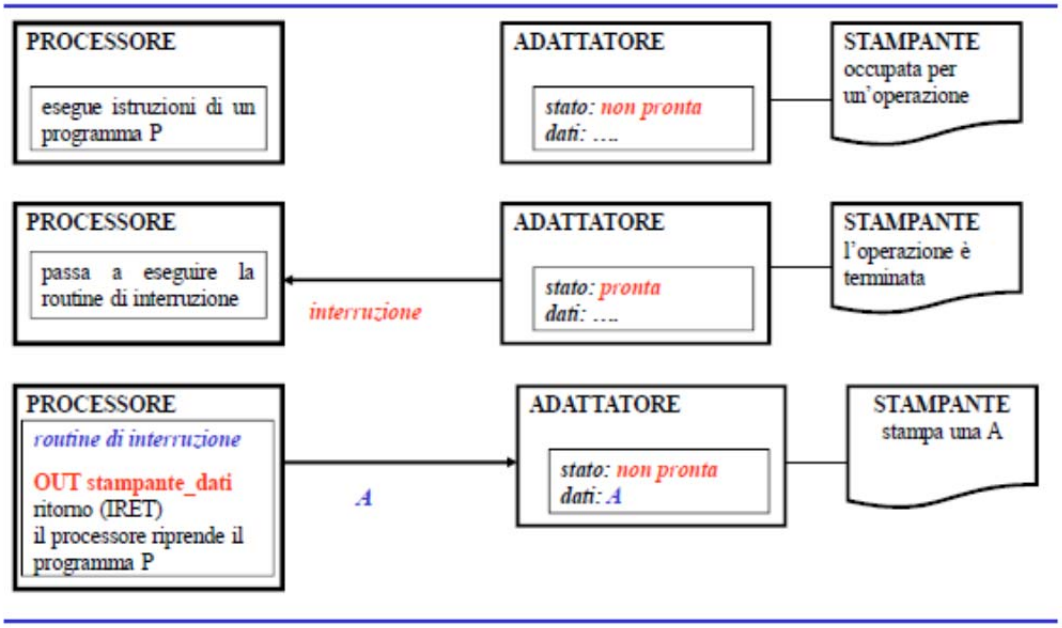


Figura 2 – Stampa di una "A" tramite meccanismo di interrupt

3. I dispositivi di memorizzazione non-volatile

La memoria di lavoro (RAM) di un calcolatore è di tipo volatile, cioè perde il suo contenuto quando viene tolta l'alimentazione. Per questo motivo ogni computer deve possedere almeno una memoria di tipo non-volatile, sulla quale sono memorizzati il sistema operativo, i programmi e i dati.
La principale memoria non-volatile è costituita dai Dischi Magnetici (HDD – Hard Disk Drives), ma a partire dai primi anni 2000 si sono diffuse sempre più le memorie a stato solido (SSD – Solid State Drive o Solid State Disk), prima nel settore dei dispositivi mobili e successivamente nei PC.

Indipendentemente dalle diverse strutture geometriche e modalità di accesso fisico all'informazione dei diversi tipi di dischi e SSD, l'indirizzamento dei dati si basa sul concetto di **LBA** (Logical Block Address); LBA è uno schema di indirizzamento nel quale l'intero disco è rappresentato come un vettore lineare di blocchi (Figura 3), ognuno costituito da un certo numero di bytes (generalmente 512 o un suo multiplo).
Nel seguito useremo il termine **Volume** per indicare qualsiasi dispositivo di memorizzazione non volatile di massa, siano essi HDD oppure SSD, dotato di uno schema di indirizzamento LBA.
Il blocco costituisce anche l'unità fondamentale di informazione che viene trasferita con una sola operazione tra il disco e la memoria centrale.

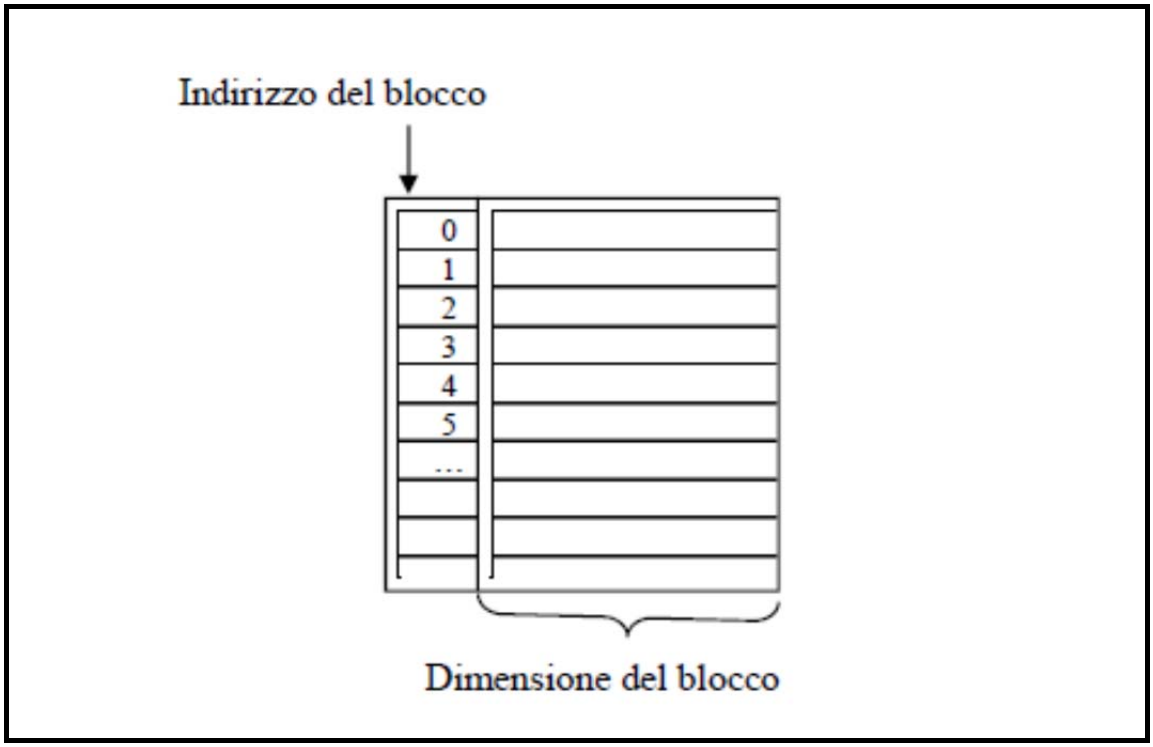


Figura 3

Il funzionamento fisico dei dispositivi che realizzano il volume può essere caratterizzato nel modo seguente, indipendentemente dal tipo di tecnologia utilizzata:

- il posizionamento all'inizio di un blocco (**latenza**) richiede un tempo molto lungo (nell'ordine delle decine di ms)
- una volta posizionato all'inizio del blocco, il trasferimento (lettura o scrittura) è molto veloce (molti Mb al secondo)

Questo modello di funzionamento è spiegabile nel caso dei dischi magnetici con le seguenti considerazioni (Figura 4):

- i dati sono registrati sul disco in cerchi concentrici detti **tracce**
- le tracce sono suddivise in **settori** (angolari)

- l'accesso fisico avviene per settori interi
- i blocchi (le unità di trasferimento) sono costituiti da uno o più settori
- per iniziare a leggere un settore è necessario eseguire due operazioni meccaniche:
    1. posizionare la testina di lettura/scrittura in corrispondenza della **traccia** desiderata (**seek time)**
    2. attendere che la rotazione del disco porti l'inizio del settore sotto la testina (**latenza rotazionale**)
- quando si inizia il trasferimento, grazie alla grande densità dei bit lungo la traccia, i bit che passano sotto la testina nell'unità di tempo è molto alto

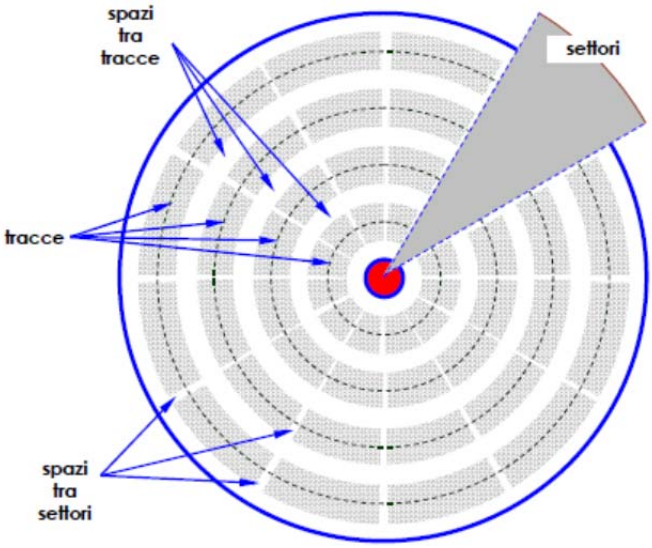


Figura 4

Talvolta vengono sovrapposti diversi dischi in un unico dispositivo in modo da aumentare la capacità senza replicare i sottosistemi di rotazione e controllo del posizionamento delle testine (Figura 5); si osservi che senza spostamento del braccio portatestine si possono leggere tutte le tracce corrispondenti sulle diverse superfici.

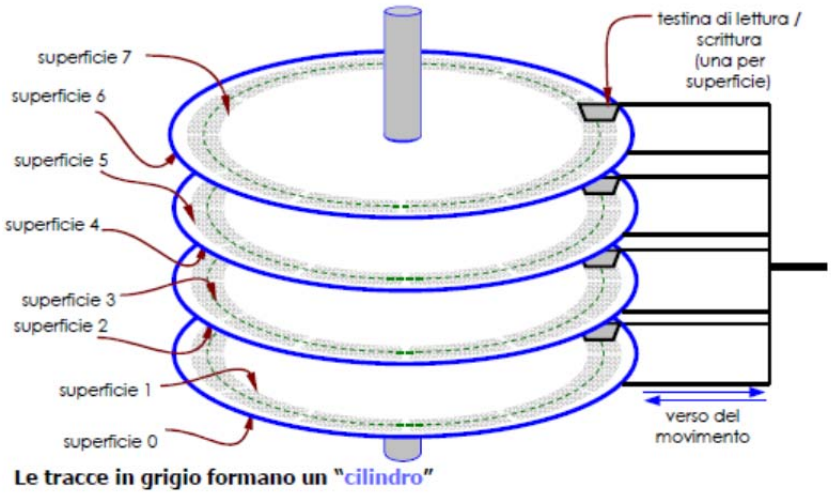


Figura 5

Infine, si consideri l'ordine di accesso ottimale a un certo numero di settori (Figura 6); per ottimizzare la lettura consecutiva di settori, la strategia può essere complessa:

- per accedere i settori 26, 100, 724 e 9987 in ordine crescente sono necessarie diverse rotazioni del disco (frecce disegnate esternamente)
- riordinando gli accessi come mostrato dalle frecce interne: 724, 100, 26, 9987 è possibile leggere tutti i settori in un'unica rotazione del disco, a condizione che gli spostamenti radiali della testina riescano a completarsi durante gli intervalli di rotazione disponibili

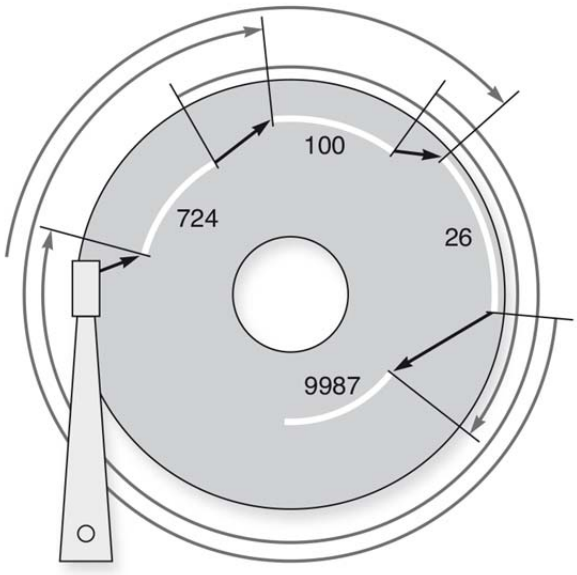


Figura 6

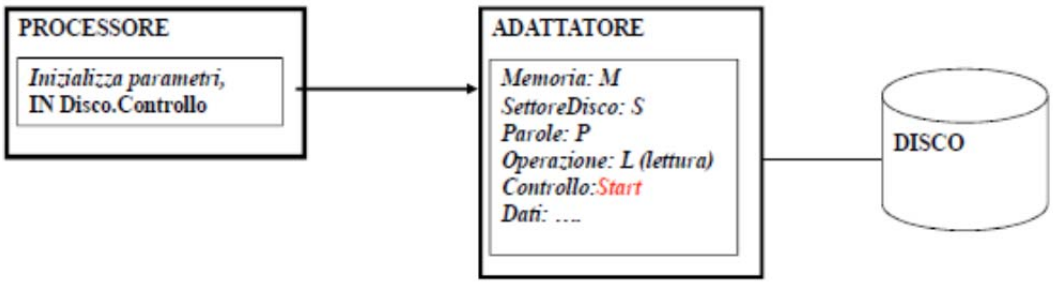
Noi non approfondiamo questi aspetti, ma ci limitiamo a trarne la seguente indicazione, utilizzata dal sistema operativo: ***è possibile e utile scegliere un ordine di accesso ottimale quando si devono leggere molti settori di un volume***.

4. Accesso diretto alla memoria da parte delle periferiche (DMA)

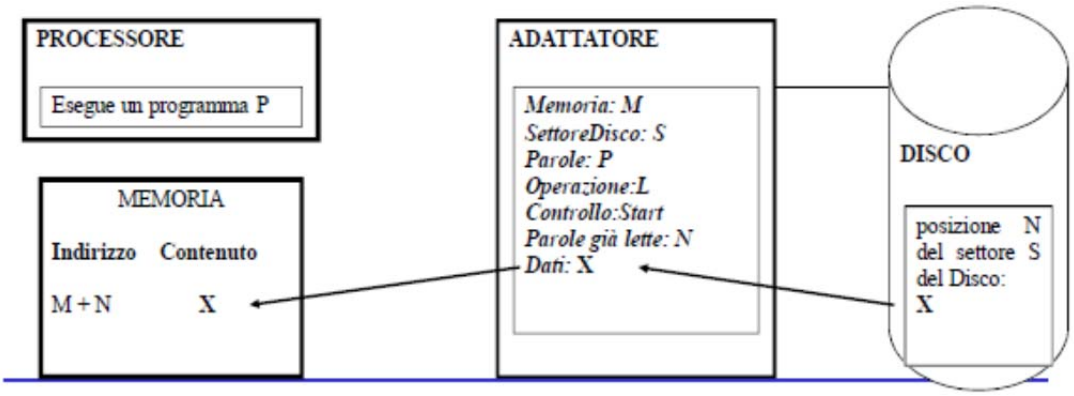
Alcune periferiche veloci (ad esempio i dischi) non richiedono al processore di intervenire per la lettura o scrittura di ogni singolo byte ma possono trasferire il contenuto di molte parole da (o alla) memoria autonomamente. Tipicamente la periferica possiede dei registri nei quali il processore scrive le seguenti informazioni:

- l'indirizzo della memoria dal quale iniziare il trasferimento
- l'indirizzo sulla periferica dal quale iniziare il trasferimento
- il numero di parole da trasferire
- la direzione del trasferimento (lettura o scrittura in memoria)

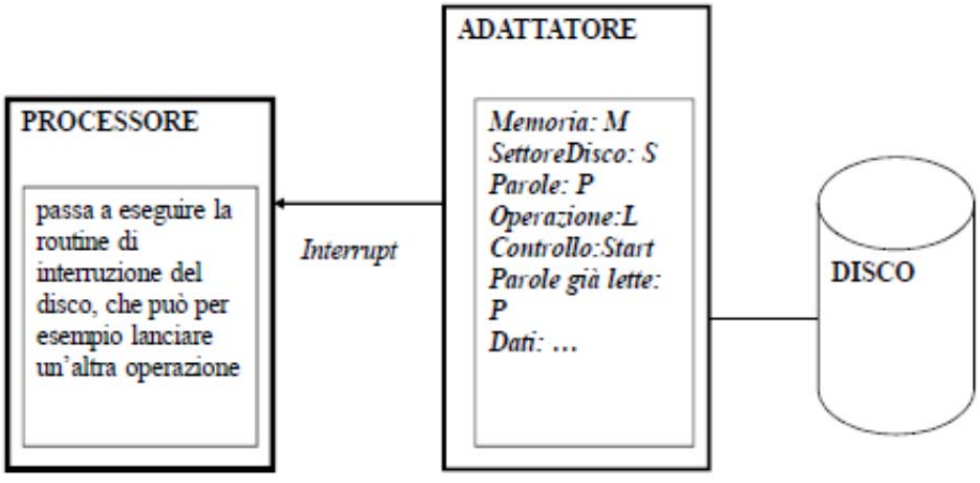
Il processore, dopo aver inizializzato tali registri, ordina alla periferica di iniziare l'operazione settando il bit "start" nel registro di comando e poi passa ad eseguire altri programmi; dopo aver completato tutta l'operazione la periferica genera un interrupt per avvisare il processore del completamento.
In figura 7 è mostrato questo meccanismo con riferimento a un disco al quale viene richiesta un'operazione di lettura. Si tenga presente che i dati sul disco sono organizzati in settori; ogni settore è identificato dal proprio numero.



a) il processore inizializza l'adattatore del disco e avvia l'operazione



b)l'adattatore DMA sta trasferendo dal disco alla memoria; N parole sono già state trasferite, l'indirizzo corrente in memoria è M+N, l'indirizzo corrente sul disco è l'N-esimo dall'inizio del settore



c) l'operazione è terminata (parole lette=P) e l'adattatore genera un interrupt

Figura 7 – Lettura di un disco tramite DMA

5. Il sistema di interconnessione tra la CPU e gli adattatori delle periferiche

In Figura 8 sono rappresentati i principali componenti del sistema di interconnessione tra il Processore e gli altri sottosistemi funzionali. L'interconnessione è costituita da insiemi di linee detti BUS. E' opportuno distinguere tra:

- bus interni del processore, che appartengono alla struttura Hardware del processore e non ci interessano in questo contesto
- bus esterni, che servono a connettere i diversi sottosistemi funzionali alla CPU; a loro volta i bus esterni possono essere strutturati in modo da possedere diverse specializzazioni:
    - bus di memoria
    - bus di IO

Figura 9 mostra che sul bus di IO vengono connessi gli adattatori o interfacce che fanno da ponte tra il meccanismo di funzionamento del bus di IO e le linee o bus specializzati per diverse tipologie di periferiche (SCSI, Centronics, ecc…)
Un Adattatore è quindi un componente che accoppia due tipi di bus; l'esistenza degli adattatori permette di collegare i diversi tipi di processori con i diversi tipi di periferica.

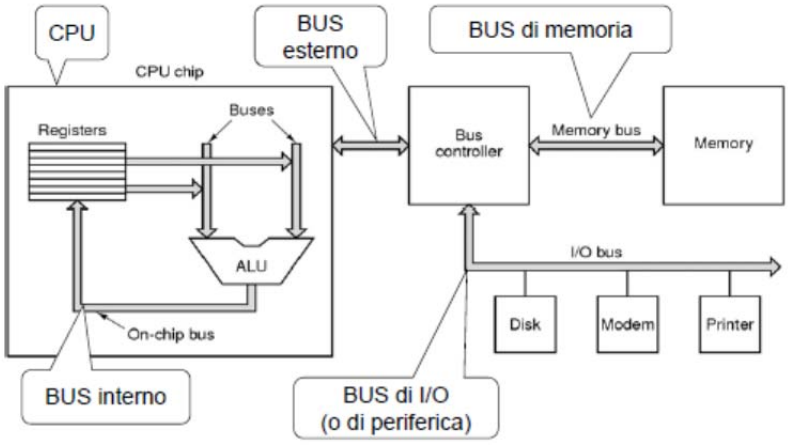


Figura 8

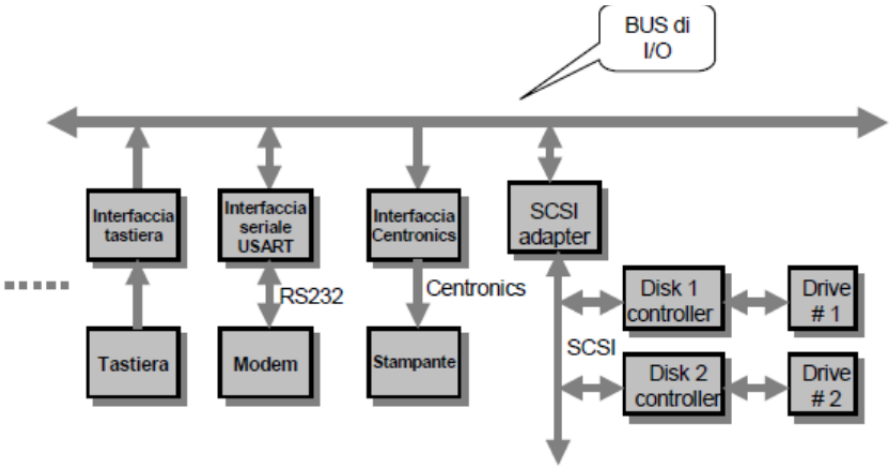


Figura 9

Le transazioni sul bus

Chiamiamo **transazione** (di bus) un insieme di operazioni sul che permette di raggiungere un obiettivo; in particolare consideriamo i seguenti due tipi di transazioni:

- **transazione di trasferimento**: trasferisce una certa quantità di informazione tra due unità funzionali
- **transazione di interrupt**: permette a una periferica di segnalare un interrupt alla CPU

La quantità di informazione trasferita da una singola transazione di bus varia in genere da 8 a 64 bit.

Transazione di Interrupt

Una transazione di interrupt è l'insieme di operazioni che conduce da una segnalazione di interrupt da parte di una periferica alla sua presa in carico da parte della CPU.

Logicamente richiede lo scambio di due segnali:

- un segnale dalla periferica verso la CPU per segnalare l'interrupt; chiameremo questo segnale **INT_REQ** (richiesta di interrupt)
- un segnale dalla CPU verso la periferica per segnalare l'accettazione dell'interrupt; chiameremo questo segnale **INT_ACK** (accettazione dell'interrupt)

Esistono molti diversi schemi Hardware per organizzare questi segnali, che differiscono notevolmente nel numero di linee richieste. Il modo più semplice consisterebbe nel prevedere una coppia di linee tra la CPU e ogni singola periferica; questo approccio, detto a *linee indipendenti*, ha però dei gravi difetti:

- il numero di linee cresce linearmente con il numero di periferiche connesse in una data configurazione di sistema
- la CPU deve possedere un numero di connettori sufficiente a supportare la configurazione più complessa accettabile, quindi tale numero risulterà ampiamente sovradimensionato per configurazioni più piccole

Esistono molti modi diversi di superare questi difetti; noi consideriamo ad esempio il meccanismo detto in ***daisy chain*** (daisy chain significa "festone").

Nel meccanismo di interrupt in daisy chain si utilizzano due sole linee per i segnali INT_REQ e INT_ACK, collegate secondo la seguente modalità:

- **INT_REQ** collegata in **wired_or** (or filato)
- **INT_ACK** collegata in **daisy chain** (festone)

Il collegamento in wired or permette a molte unità di collegarsi su una stessa linea e si comporta nel modo seguente:

- se nessuna unità emette il segnale INT_REQ la linea resta a riposo
- se una o più unità emettono il segnale INT_REQ la linea indica l'esistenza di tale segnale, che quindi rappresenta l'OR di tutti gli INT_REQ emessi

Per ragioni puramente tecnologiche (vedi figura 10) in realtà lo stato di riposo della linea corriponde al valore 1 (elettricamente a tensione V_{cc}) e il segnale di INT_REQ è rappresentato sul bus dal valore 0 (eletricamente a terra); per questo motivo nel processore è indicata la presenza di un negatore sull'ingresso del segnale e il segnale INT_REQ è indicato come negato.

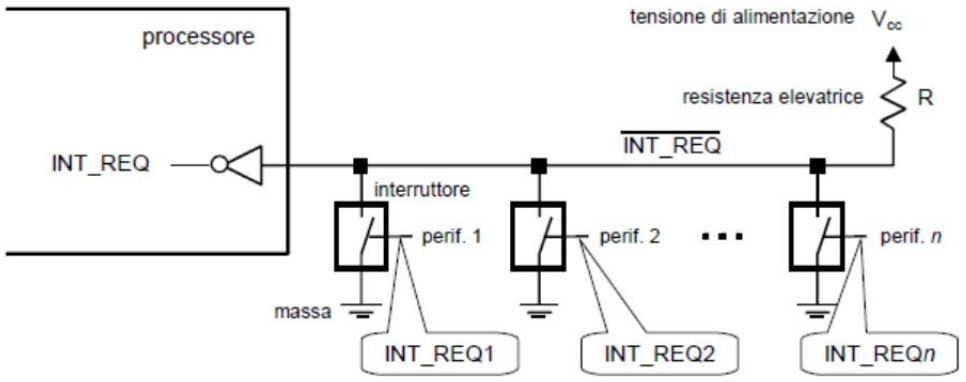


Figura 10

Il collegamento in daisy chain del segnale INT_ACK è mostrato in figura 11; il segnale viene inviato dalla CPU alla prima periferica, che lo propaga alla successiva, e così via.
Il funzionamento del sistema è il seguente:

- se una o più unità hanno fatto richiesta su INT_REQ, la CPU osserva tale segnale e invia INT_ACK
- INT_ACK viene propagato da una periferica all'altra fino a quando raggiunge una periferica che aveva fatto richiesta (esempio periferica 2)
- la periferica toglie la propria richiesta da INT_REQ

La CPU dovrà a questo punto invocare la routine di interrupt che dovrà interrogare i registri di stato delle periferiche (*polling*) nello stesso ordine della connessione fisica per scoprire la periferica da servire e poi potrà svolgere la funzione richiesta dall'interrupt.
Per rendere il meccanismo in grado di funzionare è necessario aggiungere una serie di regole ulteriori che omettiamo.

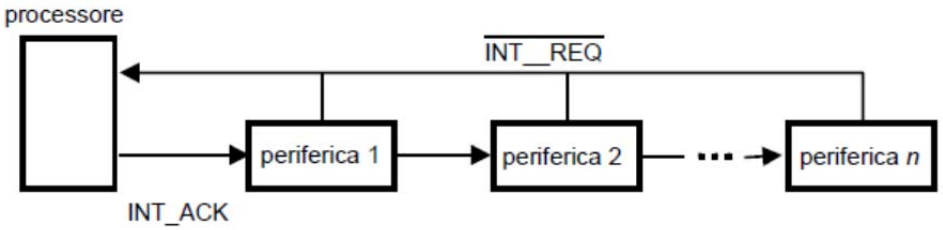


Figura 11

Transazione di trasferimento

Una **transazione di trasferimento** si può generalmente scomporre in due fasi principali, ognuna costituita da diverse operazioni:

- fase di **arbitraggio**: serve a selezionare un'unità, detta **MASTER**, che controlla il bus durante l'operazione
- fase di **trasferimento** vero e proprio, durante la quale avvengono le seguenti operazioni
  1. il MASTER seleziona un'altra unità, detta **SLAVE**, con la quale operare
  2. il MASTER indica la direzione del trasferimento:
     a. lettura (dallo SLAVE verso il Master)
     b. scrittura (dal MASTER verso lo SLAVE)
  3. viene effettuato il vero e proprio trasferimento di unità di informazione tra MASTER e SLAVE

Nota Bene: le unità che svolgono il ruolo di Master e di Slave sono fissate per ogni singola operazione di trasferimento del bus, ma possono variare tra operazioni diverse

Le unità che possono diventare Master sono:

- tutte le CPU (per eseguire le istruzioni di IN e OUT)
- i Co-processori (per trasferire informazioni alle/dalle CPU o alla/dalla memoria)
- gli adattatori in DMA (per trasferire informazioni alla/dalla memoria)

Fase di Arbitraggio

Anche per la fase di arbitraggio, come per le transazioni di interrupt, esistono molti schemi alternativi, Ci limitiamo a descrivere lo schema Centralizzato in Daisy Chain per il Prossimo Master

- centralizzato significa che esiste una singola unità specializzata che svolge il ruolo di **Arbitro**
- **Prossimo Master** significa che l'arbitraggio è svolto mentre è ancora in corso una fase di trasferimento, quindi:
  - esiste già un Master Corrente che governa un'operazione di trasferimento
  - l'arbitraggio procede in parallelo per determinare l'unità che diventerà Master Corrente quando l'attuale fase di trasferimento sarà conclusa

Il ruolo di Arbitro nei sistemi monoprocessore è spesso svolto dalla CPU invece che da un'unità specializzata
In Figura 12 sono mostrati i segnali utilizzati in questo schema, con il processore che svolge il ruolo di arbitro; in base alle convenzioni già viste possiamo dire che:

- la linea BUS_BUSY è in wired or, può essere asserita da tutte le unità e tutte le unità la possono leggere
- la linea BUS_REQ è collegata in wired, tutte le unità la possono asserire ma è letta solo dall'arbitro (processore)
- la linea BUS_GRANT è collegata in daisy chain

Il significato di BUS_REQ e BUS_GRANT è molto simile ai quello dei segnali INT_REQ e INT_ACK, con la differenza che la richiesta si riferisce alla mastership del bus invece di richiedere un interrupt e il grant è l'attribuzione della mastership.
Il segnale BUS_BUSY viene asserito dall'unità che è Master Corrente dal momento in cui lo diventa al momento in cui ha terminato di usare il bus per un trasferimento per indicare che il bus è occupato. Dato che questa linea è leggibile da tutte le unità, l'unità che diventa Prossimo Master può controllarla e attendere che il bus diventi libero prima di iniziare a utilizzarlo per il proprio trasferimento.

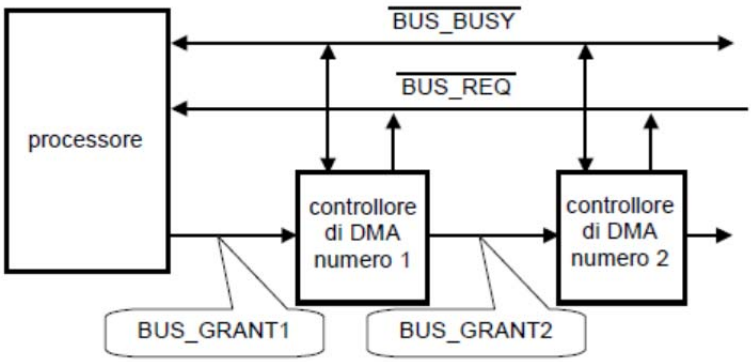


Figura 12

Fase di trasferimento
Abbiamo visto che la fase di trasferimento richiede le seguenti operazioni:

1. il MASTER seleziona un'altra unità, detta **SLAVE**, con la quale operare
2. il MASTER indica la direzione del trasferimento:
    a. lettura (dallo SLAVE verso il Master)
    b. scrittura (dal MASTER verso lo SLAVE)
3. viene effettuato il vero e proprio trasferimento di unità di informazione tra MASTER e SLAVE

La prima operazione richiede di utilizzare delle opportune linee di indirizzamento del bus per individuare il registro di periferica o la parola di memoria interessata.
La seconda operazione richiede una linea, detta generalmente R/W, per indicare se l'operazione è di lettura oppure scrittua.
La terza operazione utilizza le linee Dati del bus per trasferire l'informazione.

Linee del bus
Riassumendo, un bus basato sui meccanismi descritti dovrebbe possedere le seguenti linee:

1. INT_REQ e INT_ACK
2. BUS_BUSY, BUS_REQ e BUS_GRANT
3. RW
4. linee di indirizzo
5. linee dati

Spesso nella terminologia corrente si indicano le linee diverse da indirizzi e dati come linee di controllo.

Sincronizzazione delle operazioni
Abbiamo descritto il funzionamento logico del bus omettendo il problema tecnicamente più difficile: la gestione corretta dei tempi delle diverse operazioni, ovvero la loro sincronizzazione. Questo problema è troppo complesso per essere affrontato qui; ci limitiamo a indicare i principali motivi per cui si tratta di un problema complesso:

- la velocità di risposta delle diverse unità è diversa

- la complessità delle operazioni che le diverse unità sono in grado di svolgere è diversa
- i segnali richiedono tempo per propagarsi da un'unità a un'altra
- questo tempo inoltre può variare in base alla "distanza" (fisica e strutturale) tra le unità coinvolte
- anche i segnali che partono allo stesso istante da una sorgente possono arrivare alla destinazione sfasati tra loro

Oltre a dover superare queste difficoltà un bus deve essere progettato in modo da soddisfare stringenti requisiti di velocità: un bus troppo lento può vanificare la velocità delle unità che vi sono collegate.