

INPUT/OUTPUT

256

Una delle principali attività del S.O. è il controllo dei dispositivi di input/output:

- Inviare comandi ai dispositivi,
- Catturare le interruzioni,
- Trattare le condizioni di errore,
- Fornire un'interfaccia tra i dispositivi fisici e il resto del sistema.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Hardware di I/O

257

Due differenti punti di vista:-

Hardware visto come componenti elettronici:	Hardware visto come moduli funzionali software:
<ul style="list-style-type: none">- Circuiti Integrati- Fili- Alimentatori- Motori	<ul style="list-style-type: none">- Comandi accettati dall'hardware- Funzioni fornite a corredo- Gestione degli errori- Interfaccia software.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Dispositivi di I/O

I dispositivi di I/O possono essere divisi in tre grandi famiglie:

Dispositivi a Blocchi:

Un dispositivo a blocchi memorizza le informazioni in blocchi di lunghezza fissa, ognuno dei quali ha un proprio indirizzo la cui lunghezza varia da 512 (standard) a 32768 byte.

Proprietà:

Un dispositivo a blocchi può leggere o scrivere ciascun blocco in maniera indipendente dagli altri: i dischi sono l'esempio più classico, ma anche le unità a nastro possono essere considerati dispositivi a blocchi.

Dispositivi a carattere:

Un dispositivo a caratteri invia o riceve un flusso di caratteri senza tener conto di alcuna struttura di blocco, non è indirizzabile, e non può eseguire alcuna operazione di posizionamento randomico.

Dispositivi Particolari:

- *I clock non sono indirizzabili a blocchi, e non generano o ricevono caratteri, essi non fanno altro che causare interruzioni ad intervalli di tempo ben definiti.*
- *I dispositivi a mappa di memoria non appartengono né ai dispositivi a blocchi né ai dispositivi a carattere*

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Alcuni Dispositivi di I/O e loro frequenza di dati

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (ATA-2) disk	16.7 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

Presenti lo scorso
"Secolo"!!!!

Controllori dei Dispositivi

Tipicamente i dispositivi di input/output sono costituiti da due parti: Meccanica e Elettronica.

La parte meccanica è il *dispositivo stesso*.

La parte elettronica è chiamata *controllore del dispositivo*, si presenta sotto forma di un circuito stampato o di una scheda.

La scheda del controllore generalmente contiene un connettore, nel quale può essere inserito il cavo proveniente dal dispositivo stesso; alcuni controllori possono gestire più dispositivi (due, quattro, otto).

Se l'interfaccia del dispositivo è uno standard de facto o risponde a degli standard internazionali (ANSI, IEEE, ISO), i produttori possono fabbricare dispositivi e/o controllori conformi a tali standard (SCSI, IDE).

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Un Esempio di Controllore

L'interfaccia tra dispositivo e controllore è a bassissimo livello.

Consideriamo per esempio il caso di un disco formattato con 256 settori di 512 byte per traccia:

INVIO: Il dispositivo fisico manda al controllore una successione di bit ($4096 = 512 \times 8$ ossia un blocco) preceduti e seguiti da un certo numero di bit informativi rappresentati il preamble (numero del cilindro e del settore, sua lunghezza, etc..) e dal checksum (codice di errore) rispettivamente.

CONTROLLO: Compito del controllore è di convertire il flusso seriale di bit in un blocco di byte e di eseguire l'eventuale correzione d'errore valutando il checksum.

ASSEMBLA: Il controllore assembla il blocco in un suo buffer interno e successivamente lo scrive nella memoria di massa.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Dispositivo e CPU: Comunicazioni

Dall'esempio di prima si deduce che ogni controllore ha dei **buffer** o **registri** per la comunicazione con la CPU.

Manipolando opportunamente tali **registri**, il S.O. può chiedere al controllore del dispositivo di:

Richieste al controllore del dispositivo

- Spedire o accettare dati
- Accendersi o spegnersi
- Eseguire azioni alternative
- Conoscere lo stato del dispositivo.

Azione o Funzione da compiere

Attraverso il **buffer** dati il S.O. può leggere o scrivere dati da e per il dispositivo.

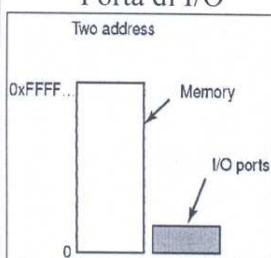
Parametri

Esempio: le schede grafiche che gestiscono I/O con il terminale hanno la RAM video per visualizzare i pixel sul monitor, ossia un buffer in cui il S.O. e i programmi possono scrivere o leggere le informazioni da visualizzare sul monitor.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Comunicazione Tra CPU e Registri del Controllore

Porta di I/O



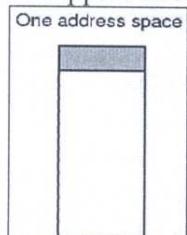
Ogni registro è identificato da un numero di porta (intero a 1 o 2 byte).

Le istruzioni per leggere o scrivere su un registro del controllore sono:

- IN Reg, Porta (IN R0,4)
- OUT Porta, Reg (OUT 14, R1)

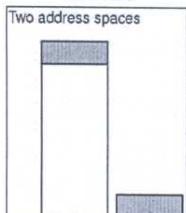
Rispettivamente la CPU trasferisce nel o dal registro della CPU il valore presente sulla Porta identificata dal valore numerico.

I/O Mappato in Memoria



Tale approccio consiste nel mappare i registri del controllore in memoria centrale, ad ogni registro viene assegnato un indirizzo di memoria unico a cui non corrisponde nessuna parola della memoria.

Ibrido



I/O del buffer dati mappato in memoria e porte di I/O separate per i registri del controllore.

Il Pentium usa tale approccio: da 640 a 1M per il Buffer dati e le porte di I/O con valore numerico da 0 a 64K (16 Bit).

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Analisi dello schema di comunicazione

Quando la CPU vuole leggere una parola sia da una porta sia dalla memoria, effettua i seguenti passi:

- Trasferisce l'indirizzo, di cui ha bisogno, sulle linee di indirizzamento bus.
- Imposta un segnale di READ sulla linea di controllo BUS
- Una linea parallela determina se la richiesta riguarda lo spazio di I/O o di memoria: se la richiesta è rivolta alla memoria, sarà la memoria a rispondere, in caso contrario sarà un dispositivo a rispondere.

Precisazione: Nel caso di I/O mappato in memoria, ogni dispositivo confronterà le linee di indirizzamento con gli spazi di memoria ad esso assegnati, se riconoscerà l'indirizzo (*ricade tra quelli ad esso assegnati*) risponderà alla richiesta.

Nota: non si possono creare ambiguità o conflitti in quanto non è possibile che un indirizzo sia assegnato contemporaneamente alla memoria e all'indirizzamento del dispositivo.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Virtù dell'I/O Mappato in Memoria

Facilità d'uso: Con un I/O mappato in memoria i registri possono essere visti come generiche variabili e gestiti comodamente con un *qualsiasi linguaggio di programmazione* di alto livello (C, Pascal), quindi il controllore del dispositivo può essere scritto con esso. Contrariamente all'uso delle porte in cui è preferibile usare l'Assembler per la loro manipolazione.

Sicurezza:

Non occorrono meccanismi di protezione speciali per impedire ai processi utente di eseguire dell'I/O.

- Conflittualità: Il S.O. evita l'inserimento in uno spazio di indirizzamento virtuale dello spazio di indirizzamento contenente i registri del dispositivo.
- Suddivisione: Il S.O., definendo i registri dei diversi controllori in pagine diverse dello spazio di indirizzamento, può lasciare all'utente il controllo di certi dispositivi inibendo l'accesso ad altri.
- Riduzione: assegnando spazi diversi ai controllori di specifici dispositivi si può ridurre il kernel del S.O. ed inoltre evitare l'interferenza tra dispositivi analoghi.

Elasticità:

Ogni istruzione che fa riferimento alla memoria può fare riferimento ai registri mappati di un dispositivo, quindi istruzioni di test e/o confronto possono essere usate sui registri per valutarne il contenuto senza overhead.



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Vizi dell'I/O Mappato in Memoria

266

Caching: Supponiamo di essere in un sistema con cache e di volere testare l'indirizzo mappato in memoria relativo ad un registro di un dispositivo che rappresenta l'attività o l'inattività del dispositivo.

- Quando sarà effettuata la richiesta di lettura della memoria, i relativi indirizzi saranno inseriti nella cache e le successive letture di quell'indirizzo saranno effettuate dalla cache.
- il sistema non si renderà conto della transizione in quanto il dispositivo cambierà il valore nella memoria e non nella cache.

Soluzione: cache selettiva sulla memoria....inserimento di overhead.

Un solo spazio di indirizzamento: tutti i moduli della memoria e i dispositivi devono analizzare i riferimenti alla memoria e decidere a quali rispondere.

Bus Singolo: facile da analizzare.

Bus Multiplo: i dispositivi non hanno modo di vedere gli indirizzi della memoria (tali indirizzi viaggiano su un bus specifico) e quindi non hanno modo di rispondere.

TRE soluzioni prospettate:

S1: inviare la richiesta sul bus della memoria, se questa non risponde inviare la stessa richiesta sul bus dei dispositivi.

S2: dispositivo SPIA che controlla quali indirizzi sono destinati ai dispositivi

S3: al boot del sistema identificare e destinare una parte di indirizzamento ai dispositivi e marcarlo come non memoria. Gli indirizzi che cadono in questo spazio sono inviati al bus dei dispositivi.

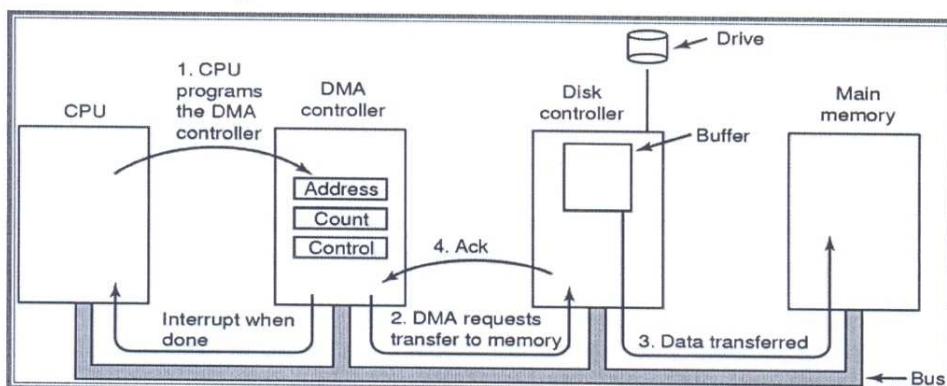
Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I/O Direct Memory Access

267

Direct Memory Access (DMA), controllore di accesso in memoria, esso permette di svincolare la CPU dalla comunicazione con i dispositivi.

Il DMA può essere inserito all'interno dei dispositivi (tanti DMA quanto sono i dispositivi) oppure un unico DMA integrato nella scheda madre.



Dovunque si trova il DMA (scheda madre o dispositivo), il DMA ha accesso al bus di sistema indipendentemente dalla CPU e i suoi registri possono essere letti o scritti dalla CPU.

- Registro di controllo per le **porte** di I/O.
- Registro di **Indirizzamento** della Memoria.
- Registro **Contatore** in Byte.
- Registri di **Controllo o di stato**.
- Registro di controllo per la **direzione** (Lettura o Scrittura)
- Registro di controllo sulla **dimensione del dato**(Byte, Parola)
- Registro di controllo relativo al **numero di Byte** da trasferire.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

DMA - Come Lavora

- 1) la CPU programma il DMA (istruzione di lettura o scrittura, indirizzo di partenza e numero di byte) in modo che esso conosca dove e quali dati trasferire, invia un comando al controllore del disco per trasferire i dati dal disco al buffer interno del disco.
- 2) Il DMA inizia il trasferimento effettuando una richiesta di lettura sul bus verso il controllore del disco. (NOTA: questa richiesta è una richiesta convenzionale, ossia al controllore del disco risulta trasparente la sua provenienza, CPU o DMA)
- 3) Il DMA pone sulle linee di indirizzamento l'indirizzo di memoria dove il controllore del disco deve copiare il dato, il controllore del disco trasferisce il dato da o verso la memoria.
- 4) Infine il controllore del disco da un segnale di ricevuta (acknowledgement) al DMA di operazione effettuata.
- 5) Il DMA incrementa l'indirizzo di memoria da usare e decrementa il contatore dei byte.
- 6) if contatore dei byte $\neq 0$ goto 2
- 7) Il DMA invia un'interruzione alla CPU per comunicarle che il trasferimento è terminato.

Trasferire i dati dal disco al buffer interno del disco: Tale strategia si rende necessaria in quanto il controllore del disco può controllare il CheckSum prima di iniziare il trasferimento (sicurezza sull'esattezza dei dati). Inoltre i bit del disco arrivano ad una frequenza costante e quando il bus di trasferimento è occupato questi possono risiedere da qualche parte (BUFFER del DISCO) senza essere perduti e senza appesantire il DMA di una eventuale nuova gestione per tale anomalia.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

DMA - Singoli / Multipli

I DMA singoli possono gestire una unità di I/O alla volta e la loro gestione avviene come esposta in precedenza.

I DMA multipli gestiscono più dispositivi contemporaneamente. Al suo interno troviamo tanti insiemi di registri quanto sono i canali verso i dispositivi.

La CPU carica i registri relativi a ciascun dispositivo, ogni trasferimento deve riguardare un diverso controllore di dispositivo e ogni volta che la parola o il byte è trasferito il DMA decide quale dispositivo servire.

Il DMA può essere programmato con un modello Round-Robin o a Priorità.

Il modo di trasferimento dati è duplice:

- **Singola parola o byte:** Il DMA prende possesso del BUS dati e se anche la CPU necessita di esso deve aspettare. (FURTO DEL CICLO) In questo caso la CPU viene leggermente rallentata da una interruzione per il trasferimento dei dati del DMA.
- **In modalità Blocco:** il DMA richiede al dispositivo di acquisire il bus, produce una serie di trasferimenti e successivamente lo rilascia. (MODO BURST). Al costo di una singola richiesta di BUS si trasferiscono più parole, più efficiente per il trasferimento ma meno adatto per grosse mole di dati (la CPU, in questo caso, potrebbe rimanere bloccata per molto tempo).

DMA - Indirizzi Virtuali

Modo Alternativo: Il controllore del dispositivo invia la parola al controllore del DMA (prima richiesta di bus) e successivamente il controllore del DMA scrive la parola al posto giusto (seconda richiesta di bus).

Difetti: due richieste di bus

Pregi: copie di dati tra dispositivi, copie tra memoria e memoria
(la parola è residente nel controllore del DMA).

I DMA usano indirizzi fisici quindi il SO deve convertire gli indirizzi da virtuali a fisici.

Schema Alternativo: Il DMA fa uso degli indirizzi virtuali e usa L'MMU per la conversione da virtuale a fisico.

Perché un buffer per il DMA:

Correzione dei dati: Checksum;

Sincronizzazione: - DMA non pronto a ricevere i dati,

- attesa del controllore del DMA per il bus,
- con un buffer si evitano le criticità temporali.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Principi del Software di I/O

- **Indipendenza dal dispositivo:** Scrivere programmi indipendenti dal dispositivo stesso, ossia che non necessitano di specificare il tipo di dispositivo in anticipo.

Esempio: un programma che deve usare un file deve accedere al file in maniera trasparente dal supporto che lo contiene.

- **Denominazione uniforme:** Il nome di un file o di un dispositivo deve essere indipendente dal dispositivo stesso e dovrebbe essere formato da una stringa di caratteri.

- **Trattamento degli errori:** Trattamento degli errori deve essere effettuato quanto più vicino possibile all'hardware. In molti casi il recupero dell'errore si può realizzare a livelli bassi in modo trasparente per i livelli alti.

- **Trasferimento sincrono e asincrono:** La maggior parte dell'I/O avviene in modo asincrono: la CPU fa partire il trasferimento e passa a fare altro. I programmi utenti sono più facili da gestire se le operazioni di I/O sono sincrone (read).

- **Bufferizzazione:** Il processo di bufferizzazione subentra allorquando i dati non possono essere trasferiti immediatamente e devono essere memorizzati in un contenitore (buffer) per un futuro riutilizzo.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Principali metodi Software di I/O

272

Esistono tre modi diversi per effettuare l'I/O:

- **I/O programmato;**
- **I/O guidato dalle interruzioni del dispositivo;**
- **I/O tramite DMA.**

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I/O PROGRAMMATO

273

L'I/O programmato è il più semplice tra i metodi e affida il compito di fare tutto il lavoro alla CPU:

In pratica questo metodo usa la CPU per pilotare il dispositivo

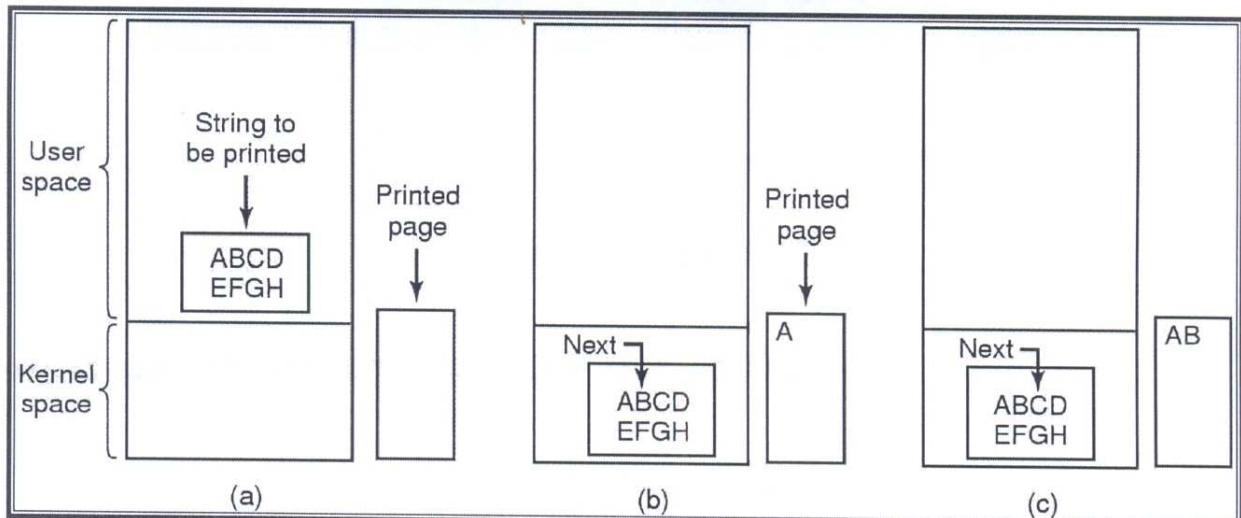
Esempio: sia esso a caratteri sia esso a blocchi.

Supponiamo che l'utente voglia stampare una stringa di 8 caratteri su di una stampante:

- 1) Assemblare la stringa in un buffer nello spazio utenti,
- 2) Acquisizione da parte del processo della risorsa (stampante) in scrittura:
 - se la stampante è occupata il processo si interrompe sino all'ottenimento della risorsa;
 - ottenuta la risorsa il processo utente effettua una chiamata al sistema in cui comunica la stringa da stampare.
- 3) Il S.O. copia il buffer contenente la stringa nello spazio del kernel;
- 4) Il S.O. copia il primo carattere nel registro dati della stampante, utilizzando ad esempio l'I/O mappato in memoria, questa azione attiva la stampante.
- 5) **Dopo la stampa del carattere la stampante modificherà il suo registro di stato per dare la disponibilità di stampa al S.O.**
- 6) Dopo avere stampato il primo carattere il S.O. controlla se la stampante è pronta per stampare il secondo carattere.

Questo ciclo termina quando l'intera stringa è stata stampata.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12



```

copy_from_user(buffer, p, count); Attesa Attiva /*p is the kernel buffer*/
for (i = 0; i < count; i++) {
    while (*printer_status_reg != READY) ; /* loop on every character */
    *printer_data_register = p[i];          /* loop until ready */
}                                         /* output one character */
return_to_user();

```

L'I/O programmato è semplice, ma ha lo svantaggio di *legare la CPU* a tempo pieno fino al completamento dell'operazione di I/O.

Se il tempo richiesto per la stampa è breve allora *l'attesa attiva* è accettabile, nei sistemi embedded in cui la CPU è dedicata a tale scopo, l'attesa attiva è ragionevole.

Nei sistemi in cui la *CPU è occupata* in maniera massiccia è opportuno usare altri metodi per effettuare l'I/O.

I/O Guidato dalle Interruzioni

Tale metodo permette alla CPU di fare qualcos'altro mentre è in attesa che la stampante sia pronta a ricevere il carattere da stampare.

Esempio: Supponiamo che una stampante sia in grado di stampare 100 caratteri al secondo, quindi la stampa di un carattere richiede un tempo complessivo di 10 ms, in questo tempo la CPU potrebbe processare altre istruzioni.

Quando viene fatta la chiamata di sistema per la stampa, come nel caso precedente, il buffer utente che contiene la stringa viene copiato nello spazio del Kernel, e il primo carattere, quando la stampante è pronta a riceverlo, viene inviato verso la stampante (I/O mapped), a questo punto il S.O. chiama lo scheduler e la CPU è usata per elaborare altri processi.

Il **processo** che ha richiesto la stampa della stringa è **bloccato** fino a quando tutta la stringa non è stata stampata.

Quando la **stampante** ha stampato il carattere ed è pronta a ricevere il successivo, **genera un'interruzione**, che ferma il processo corrente, e viene eseguita la procedura di servizio delle interruzioni delle stampanti.

NOTA: Lo svantaggio di tale metodo risiede nel richiedere tante interruzioni quanto sono i caratteri da stampare, tali interruzioni richiedono tempo sottratto alla CPU.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I/O Tramite il DMA

Questo metodo tende ad eliminare lo spreco di tempo dovuto alle eccessive interruzioni. Il DMA emette i caratteri e li passa alla stampante uno alla volta senza disturbare la CPU.

In pratica l'I/O tramite DMA altro non è che un I/O programmato in cui non è la CPU a fare il lavoro bensì il DMA.

Guadagno: le frequenze di interruzioni passano da carattere a buffer stampato.

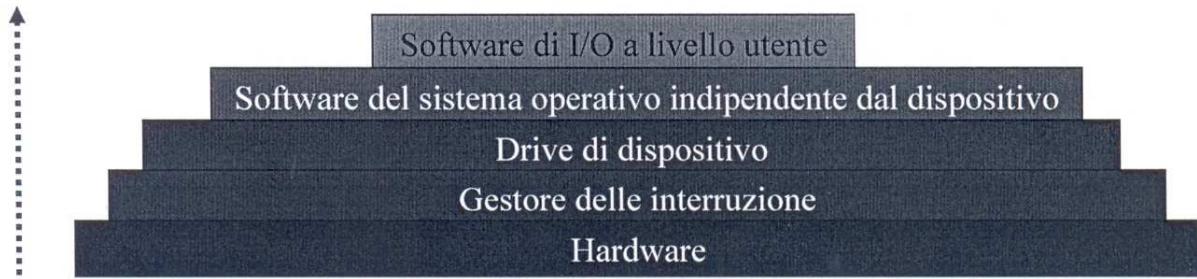
Considerazioni:

- 1) se ci sono molti caratteri, le interruzioni rallentano notevolmente l'I/O; I/O tramite DMA può essere una buona alternativa;
- 2) se il DMA non è in grado di far andare la stampante alla sua massima velocità e la CPU non è eccessivamente utilizzata (sono presenti tempi di inattività) allora l'I/O guidato dalle interruzioni o l'I/O programmato possono essere alternative migliori.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I Livelli del software dell'I/O

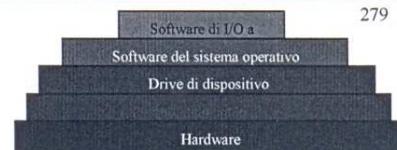
Quattro livelli per il software di I/O :



Le loro funzionalità variano da sistema a sistema.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Gestore delle interruzioni



Le interruzioni di I/O sono estremamente costose ed è opportuno *nasconderle nella parte più bassa* del S.O., così da evitare possibili interferenze con i processi di più alto livello.

Quando un'interruzione è stata prodotta, la procedura che risolve tale interruzione farà in modo da *soddisfare l'interruzione*, ossia fare ripartire il processo bloccato.

COME?

- 1) Manderà un segnale UP su di un semaforo;
Oppure
- 2) Farà una SIGNAL su di una variabile condizionale;
Oppure
- 3) Manderà un messaggio al processo bloccato.

QUINDI

L'effetto sarà di far riprendere il processo bloccato.

APPARENTEMENTE SEMPLICE E NOISO!

MA, VEDIAMO ALCUNI DETTAGLI

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Gestore delle interruzioni

Descriviamo alcuni passi:

- 1) Salvare i registri che non sono stati salvati dall'hardware delle interruzioni.
- 2) Impostare il contesto per la procedura che gestisce l'interruzione.
- 3) Organizzare una pila per tale procedura.
- 4) Mandare un segnale al controllore delle interruzioni o ripristinare le interruzioni.
- 5) Copiare i registri del processo corrente nella tabella dei processi.
- 6) Eseguire la procedura di servizio delle interruzioni, essa estrarrà le informazioni dai registri del controllore del dispositivo che ha prodotto l'interruzione.
- 7) Scegliere quale processo eseguire; generalmente quello a priorità più alta.
- 8) Organizzare la MMU e la TLB (Translation Look-aside Buffer) per il processo successivo.
- 9) Caricare tutti i registri del processo.
- 10) Eseguire il nuovo processo.

Dopo tutto questo lavoro, non so voi, ma io mi sento stanco per il processore.

Quindi, un'interruzione non è affatto semplice né noiosa né tantomeno facile.
Inoltre, le difficoltà crescono se la macchina su cui sono presenti le interruzioni è
una macchina a memoria virtuale.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Driver di dispositivo



Ogni dispositivo necessita di un codice specifico per il controllo del dispositivo stesso.

MOUSE

Movimento del mouse,
Pulsante pressato.

DISCO

Settori, tracce, cilindri, testine;
Movimento braccio;
Tempi di posizionamento.

CODICE SPECIFICO = DRIVER DEL DISPOSITIVO

Driver del dispositivo

Scritto dal

Produttore per ogni S.O.

Linux
 WINxx
 VMS
 Unix
 Mac

Driver di dispositivo

Per accedere all'hardware del dispositivo, ossia hai registri del controllore, un DRIVE può essere inserito o all'interno del Kernel o nello spazio utente.

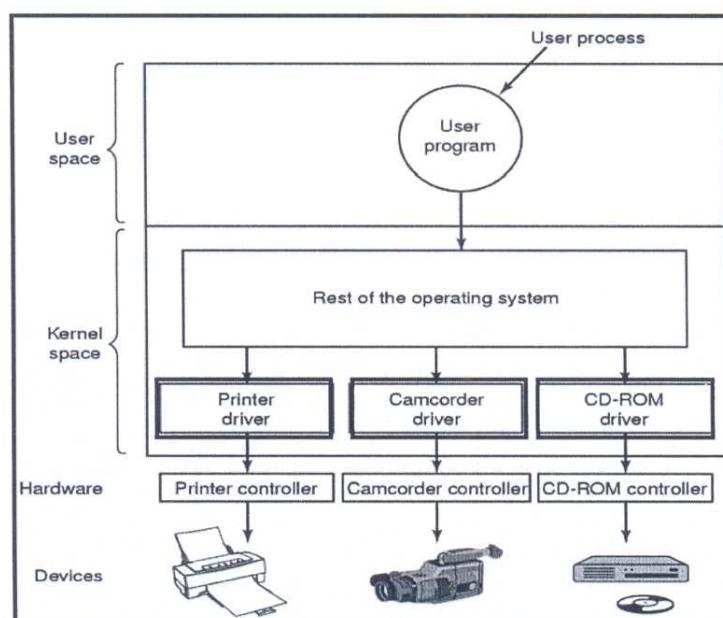
Drive nello spazio utente.	Drive nel Kernel.
<ul style="list-style-type: none"> - <u>Evita i crash del sistema</u>, in quanto un drive così definito non interferisce con il S.O. - <u>Duplicazione dei drive</u>: lo stesso dispositivo verrà gestito da tanti drive quanti sono gli utenti interessati al dispositivo. 	<ul style="list-style-type: none"> - Può interferire con il sistema provocando crash. - Riduce lo spazio di memoria utilizzato.

In alcuni casi degli attuali sistemi operativi i drive più comuni sono definiti all'interno del Kernel del S.O. (statici o dinamici?)

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Driver di dispositivo

I driver dei dispositivi sono generalmente l'ultimo scalino dei livelli del S.O. e risiedono al di sotto del S.O.



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Driver di dispositivo

Il sistema operativo suddivide i dispositivi in due grandi famiglie e si interfaccia con esse attraverso le opportune librerie.

Dispositivi a BLOCCHI

Dispositivi a CARATTERE

Le interfacce per la gestione dei dispositivi contengono le procedure per la manipolazione del dispositivo.

Esempi possono riguardare la lettura di un blocco di istruzioni per quanto riguarda i dispositivi a blocco o la scrittura di una stringa di caratteri per quanto riguarda la scrittura di una stringa su di un dispositivo a carattere.

In molti sistemi di elaborazione il sistema operativo è un unico programma binario al cui interno sono “inseriti” tutti i drive che il sistema può usare.

In alcuni S.O. l’inserimento di un nuovo dispositivo può implicare la **ricompilazione del Kernel o il reboot del sistema** prima di poter usare il nuovo dispositivo.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Driver di dispositivo

I PC hanno un modello in cui i driver vengono caricati dinamicamente durante l’esecuzione nel S.O.

Ogni drive di dispositivo ha alcune funzioni generiche che possono essere espresse come segue:

- richiesta di lettura o/e scrittura da software diversi dallo specifico drive; in generale la richiesta è effettuata su valori virtuali;
- inizializzare il dispositivo
- gestire l’alimentazione
- gestire il Log degli eventi.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Driver di dispositivo

Quali passi compie il driver di un dispositivo?

Una loro esemplificazione:

- Controllare i parametri di input e verificarne la loro validità.
- Tradurre i dati virtuali in valori reali (es: nel dispositivo disco tradurre l'indirizzo lineare in termini di testina, traccia, settore, cilindro).
- Controllare se il dispositivo è in uso o inattivo.
- Controllare il dispositivo attraverso una lista di comandi caricati negli appositi registri del controllore del dispositivo.
- Attendere risposte dal controllore del dispositivo: se la risposta è immediata il drive non viene bloccato altrimenti sarà bloccato e sbloccato successivamente con una interruzione.
- Se i controlli e le operazioni di lettura o scrittura sono andati in porto il driver dovrà passare le informazioni al software che ne ha fatto richiesta.

Piccole complicazioni:

Consideriamo un S.O. in cui è permesso inserire o disinserire dispositivi a caldo (es: device bus USB).

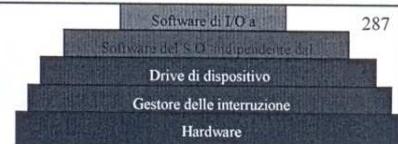
Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Mentre il sistema è impegnato a leggere un blocco l'utente può decidere di spegnere il dispositivo.

Cosa fare in questi casi?

abortire il trasferimento di I/O e rimuovere dal sistema ogni richiesta pendente verso il dispositivo.

Software del S.O. indipendente dal dispositivo



Definizione: La funzione principale del software indipendente dai dispositivi è di eseguire quelle operazioni di I/O che sono comuni a tutti i dispositivi, e inoltre di fornire un interfaccia uniforme al software di livello utente.

Funzioni svolte dal software indipendente dai dispositivi:

1. Interfaccia uniforme per driver di dispositivo
2. Bufferizzazione
3. Segnalazione degli errori
4. Allocazione e rilascio di dispositivi dedicati
5. Fornire una dimensione del blocco indipendente dal dispositivo.

Il confine tra il software indipendente dal dispositivo e i driver non è quasi mai ben delineato e dipende fortemente sia dal sistema sia dal dispositivo.

Interfaccia uniforme per driver di dispositivo

Interfaccia per rendere i dispositivi simili tra di loro.

Interfacce Multiple	Interfacce Unificate
Situazioni in cui ogni drive di dispositivo ha una propria interfaccia impone chiamate che <u>differiscono da drive a drive</u> e inoltre le chiamate al kernel variano da driver a driver. In questa configurazione interfacciare un nuovo driver richiede un <u>grande sforzo di programmazione</u> .	Di contro <u>l'inserimento di un driver che sia conforme a una interfaccia comune</u> può risultare estremamente semplice. I programmati di drive sanno cosa poter richiedere al Kernel (set di istruzioni ammesse) e quali funzioni fornire al sistema (funzioni operative del dispositivo).

Un altro aspetto dell'interfaccia comune è il modo in cui sono denominati e identificati i dispositivi.

In questo modo usando due indirizzi numerici:..

Numero principale di dispositivo - identifica il DRIVE per il dispositivo

Numero di dispositivo secondario - identifica il DISPOSITIVO o unità su cui si deve leggere o scrivere.

Si IDENTIFICANO tutte le unità o i dispositivi presenti nel sistema, inoltre la protezione di tali unità la si ottiene con la stessa politica di protezione dei file.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Bufferizzazione

Il concetto di bufferizzazione è estremamente importante nelle operazioni che riguardano i dispositivi, permette di risolvere molti problemi legati al trasferimento dei dati da un dispositivo verso il sistema.

Esempio di lettura dei dati da un modem:

Supponiamo di avere un processo che voglia leggere da un modem, esso effettuerà una chiamata read e si bloccherà in attesa di un carattere proveniente dal modem, il modem invierà a sua volta un interruzione al processo quando invierà un carattere, questa sbloccherà il processo consegnandogli il carattere, il processo dopo aver acquisito il carattere effettuerà una ulteriore operazione di read bloccandosi.

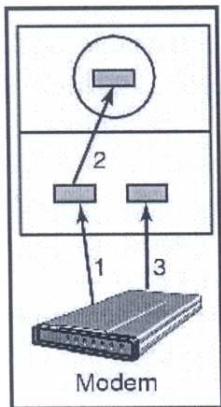
Questo approccio NON va bene: il processo si blocca troppo frequentemente e il kernel interviene troppo spesso(overhead) per gestire le interruzioni.

Soluzione alternativa: Il processo utente fornisce un buffer con n caratteri, la procedura delle interruzioni mette i caratteri in ingresso in questo buffer sino a riempirlo, successivamente sblocca il processo.

NOTA: tale soluzione diminuisce i tempi di overhead quindi migliora le performance; MA!!!, se il buffer è scaricato dalla memoria (paginazione)?, soluzione: la blocchiamo in memoria centrale, e se ho molti buffer di memoria....avrei molte pagine bloccate con conseguente riduzione della risorsa memoria e se,...e se...., e se....?

Bufferizzazione

Soluzione a doppia bufferizzazione:



Questa è la soluzione più usata, si considerano due buffer nel kernel e uno nello spazio utente, quando il primo buffer del kernel si riempie risulta possibile riversarlo nel buffer utente, e contemporaneamente alla copiatura del primo buffer i nuovi dati vengono caricati nel secondo buffer.

In sintesi mentre un buffer viene copiato l'altro viene riempito, quindi i due buffer lavorano in alternanza.

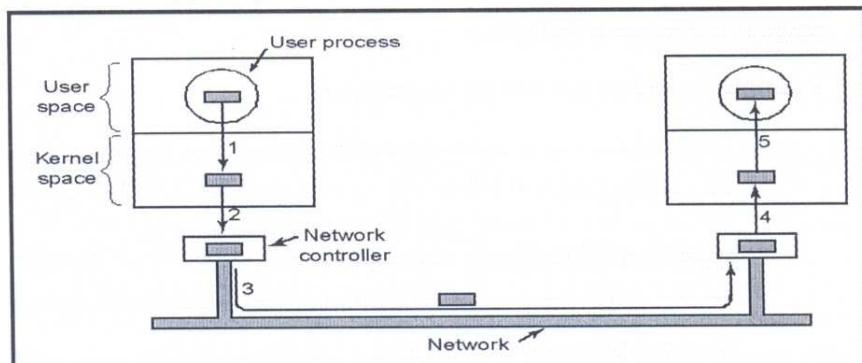
Tale soluzione può essere adottata anche dai processi che necessitano di I/O!!!!!

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Bufferizzazione

Aspetti Negativi:

La bufferizzazione usata troppo spesso riduce le prestazioni del sistema:



Perchè:

Passo 1: Il kernel copia il pacchetto nel buffer del kernel e lascia procedere l'utente;

Passo 2: Successivamente il S.O. copia tale contenuto del buffer nel registro del controllore del dispositivo.

Passo 3: A questo punto il pacchetto viene copiato sulla rete.

Per il ricevitore i passi andranno dal 3 al 1.

Quindi nei 6 passi (trasmettitore e ricevitore) è presente un gran numero di istruzioni di copiatura che rallentano il trasferimento.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Segnalazione degli Errori

Nelle operazioni di I/O gli errori sono molto frequenti, molti di essi sono specifici dei dispositivi e gestiti dal driver specifico del dispositivo.

Errori di I/O di programmazione:

- un processo chiede qualcosa di impossibile (scrivere su di un dispositivo di input, o leggere da uno di output)
- fornire indirizzi di buffer non validi o dispositivi non presenti.

Azione:

si riporta un codice di errore al processo chiamante.

Errori di I/O veri e propri:

- cercare di scrivere su di un blocco di disco che è danneggiato,
- cercare di acquisire da una telecamera che è spenta,

Azione:

In tali circostanze è il drive che decide cosa fare, se NON è in grado di prendere una decisione rimanda il problema al software indipendente dal dispositivo, e così via in salita sino ad arrivare al processo chiamante.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Assegnare e rilasciare dispositivi dedicati

Alcuni dispositivi possono essere usati solo da un processo alla volta, per tale motivo si dicono dedicati.

Le strategie sono molteplici, per un corretto uso di tali dispositivi si possono considerare le seguenti due strategie:

- 1) il processo interessato fa una chiamata diretta al dispositivo (open) se questa fallirà vorrà dire che il dispositivo è occupato; il dispositivo sarà rilasciato con una close dal processo che lo detiene.
- 2) un approccio alternativo è quello di fare una richiesta al SO, se il dispositivo è occupato il processo sarà messo in attesa sino a quando il dispositivo uno risulterà disponibile.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Dimensione dei blocchi indipendenti dal dispositivo

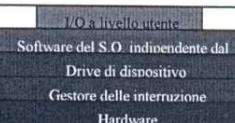
Il software indipendente da dispositivo nasconde le possibili diversificazioni di dimensione del blocco tra i dispositivi (es. dischi).

Esso uniforma il blocco tra i dispositivi creando un blocco virtuale composto da uno o più blocchi fisici (dipende dal dispositivo), sarà il controllore del dispositivo a trasformare il blocco virtuale in blocco fisico.

Analoga strategia viene usata per i dispositivi a carattere.

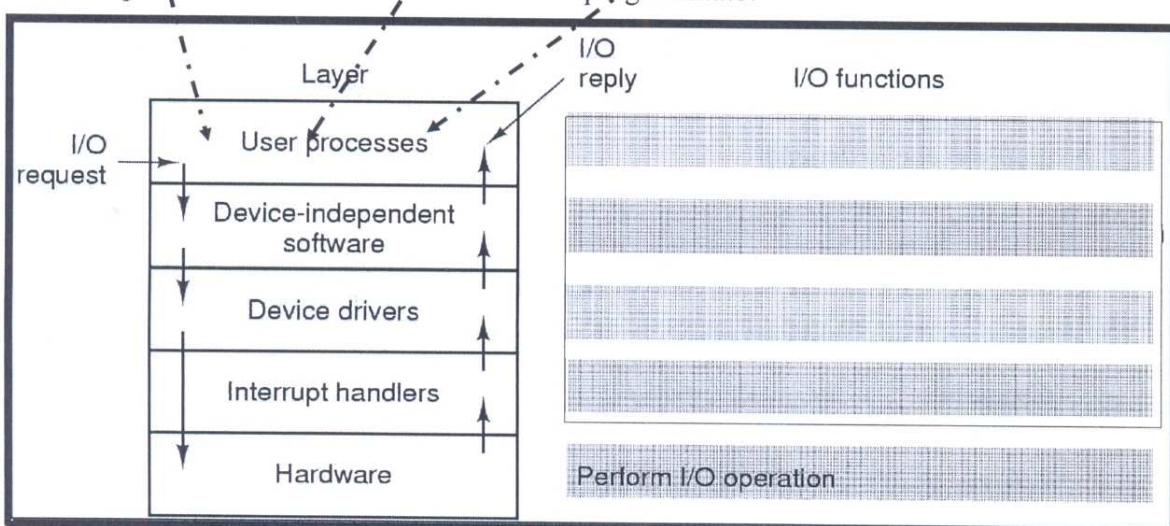
Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Software di I/O a livello utente

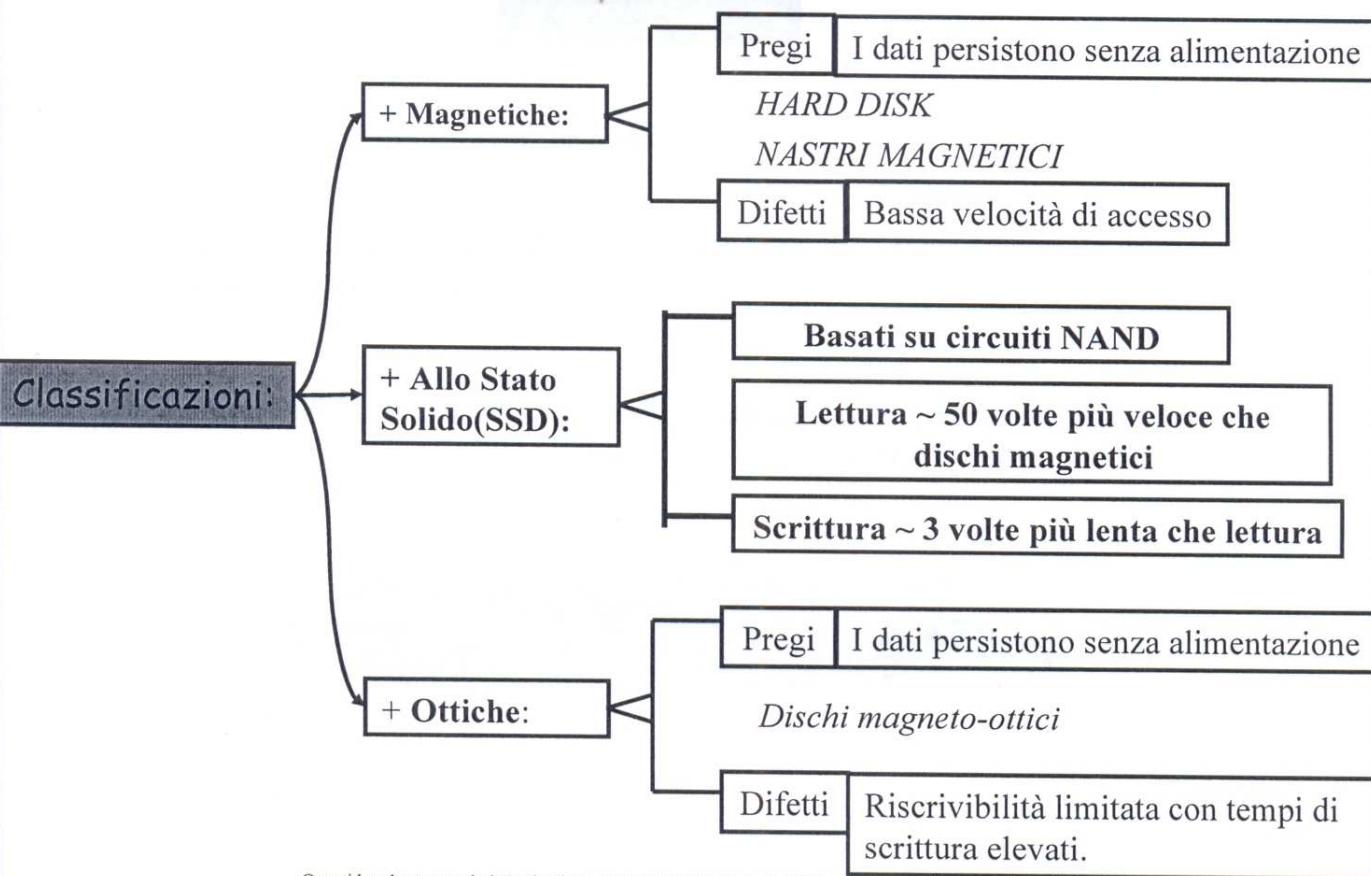


Una parte del software di I/O è **residente nelle librerie richiamate dall'utente**, esse sono inserite nel programma utente al momento della compilazione o del link.

- Generalmente alcune di esse fanno ben poco I/O e si limitano a passare parametri (write, read).
- Altre funzioni di libreria come la *scanf* o *printf* del linguaggio C hanno una gestione un po' più articolata e tra i loro compiti è presente la formattazione del testo.
- Altra categoria è il sistema di SPOOL, lo spooling è una maniera particolare di trattare i dispositivi dedicati in un ambiente multiprogrammato.



I DISCHI



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

SULLA MEMORIA DI MASSA



Definiamo memoria di massa un componente informatico in grado di contenere tutti i programmi che possono servire all'utente insieme ai propri dati;
inoltre è in grado di mantenere i dati per un tempo indefinito anche in assenza di elettricità.

I principali dispositivi per realizzare le memorie di massa sono basati su tecnologia magnetica e ottica.

La Memoria di massa può avere capacità che varia da pochi GIGAbyte a centinaia di TERAbyte.

Il suo accesso può essere sequenziale (nastri magnetici) o ad accesso diretto (dischi magnetici o ottici).

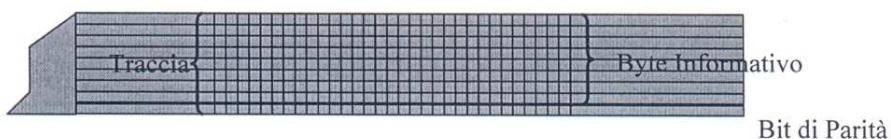
TIPI DI MEMORIA DI MASSA

Nastri Magnetici

Sono nastri di materiale magnetizzabile avvolto su supporti circolari, o su cassette.

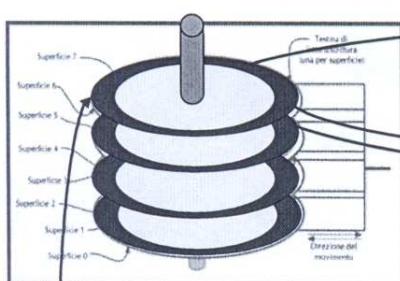
Tali supporti poiché sono ad accesso sequenziale, di bassa velocità di accesso e con grandi capacità di memorizzazione sono utilizzati largamente per effettuare copie di sicurezza(backup).

Il nastro è suddiviso in piste orizzontali parallele, ciascuna di 9 bit (un byte informativo + un bit di parità).

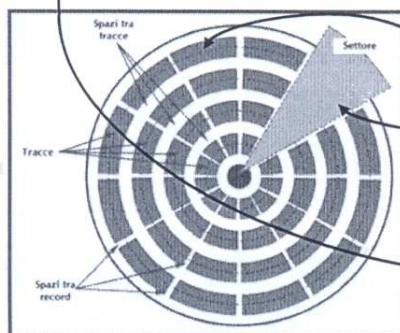


Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Dischi Magnetici



- Un disco magnetico è costituito da un insieme di piatti con due superfici che ruotano attorno ad un *perno centrale*.
- Ogni superficie è dotata una *testina di lettura/scrittura*.



- Le superfici sono organizzate in cerchi concentrici detti *tracce* e in spicchi di uguale grandezza detti *settori*.
- Tutte le tracce equidistanti dal centro, poste su piatti diversi, formano una superficie detta *cilindro*.
- I dati sono scritti occupando posizioni successive lungo le tracce.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Dischi Magnetici

Caratteristiche:

- I dischi sono organizzati in cilindri, ognuno dei quali contiene tante testine quante sono le superfici del disco.
- Operazioni di SEEK su due o più dischi contemporaneamente: mentre il controllore e il software aspettano per il posizionamento su di un disco, questi possono iniziare l'operazione di posizionamento su di un altro.
- Operazioni di lettura o scrittura su di un disco possono essere effettuate mentre si sta effettuando un'operazione di posizionamento su di un altro disco.
- Su sistemi in cui sono presenti più dischi è possibile operare simultaneamente sino al punto da effettuare trasferimenti tra dischi e buffer del controllore contemporaneamente.

NOTA_1: È concesso un trasferimento alla volta tra controllore e memoria centrale PERCHÉ?

NOTA_2: La dimensione dei settori per traccia diminuisce all'avvicinarsi al centro del disco PERCHÉ?

Mappatura Logica	Mappatura Reale
Equidistribuzione dei settori	Mappatura secondo una geometria differenziata

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

PARAMETRI DELLE MEMORIE

parametri	memoria di massa	memoria centrale	memoria cache	registri
tipo	elettromeccanica	elettronica	elettronica	elettronica
funzione	input/output archiviazione	memoria di lavoro del processore	memoria di lavoro del processore	supporto al processore
permanenza	permanente	volatile	volatile	volatile
capacità di memoria	alta (Gb)	media (Mb)	bassa (Kb)	molto bassa (byte)
modalità di accesso	sequenziale diretto	casuale	casuale	casuale
tempo di accesso	alto (ms)	basso (ns)	molto basso (ns)	molto basso (ns)
costo	basso	medio	alto	molto alto
interazione processore	bassa	media	alta	molto alta

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

D I F F E R E N Z E T R A L E M E M O R I E D I M A S S A

disco	tecnologia	capacità di memoria	tempo di accesso	costo	operazioni consentite
flessibile	magnetica	alcuni Mb	centinaia di ms		lettura/scrittura
rigido	magnetica	da alcuni a centinaia di Gb	alcuni ms		lettura/scrittura
cd	ottica	650 Mb	centinaia di ms		sola lettura 1 scrittura/lettura 1000 scritture/lettura
dvd	ottica	da 4.7 a 17 Gb	centinaia di ms		sola lettura 1000 scritture/lettura

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I R A I D

L'elaborazione parallela ha portato, negli ultimi anni, ad una notevole incremento delle prestazioni della CPU. Il riflesso analogo si ha per quanto riguarda I/O con la metodologia RAID sull'uso dei dischi.

R.A.I.D.: Redundancy Array Inexpensive Disks

Successivamente trasformato in:

R.A.I.D.: Redundancy Array Independent Disks

RAID In contrapposizione ai	SLED
Batteria di dischi economici	Unico disco grande e costoso

Sei livelli di metodologie RAID

per la gestione dei dischi.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I 6 RAID

RO

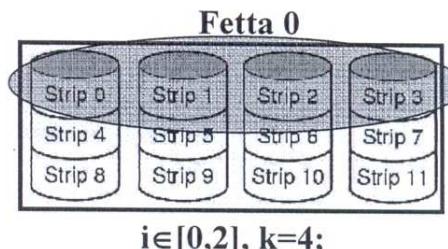
I dischi RAID hanno la proprietà di distribuire i dati su dischi diversi, favorendo l'elaborazione parallela.

Esempio: Una batteria di dischi SCSI (7 o 15) può essere vista come un unico disco, in questo modo non si devono fare eccessivi cambiamenti al software per la loro gestione.

R.0: ogni disco virtuale viene visto come suddiviso in n fette di k settori (k può coincidere con il numero dei dischi).

Esempio: fetta 0 settori da 0 a 3;
fetta 1 settori da 4 a 7;

In generale per la fetta i compresa tra 0 e $n-1$ i suoi settori saranno compresi tra $i*k$ a $(i+1)*k - 1$, dove k è il numero di settori per fetta.



$$i \in [0,2], k=4;$$

Caratteristiche: Il controllore RAID (generalmente HARDWARE, ma esistono esempi di controllori software), trasformerà il singolo comando in quattro comandi separati, ognuno dei quali sarà rivolto ad ogni singolo disco, operando in tal senso in modalità parallela, senza che il software se ne accorga.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I 6 RAID

RO

Vantaggi di R.0:

- 1) Lavora meglio nel caso di grandi richieste, **se** la richiesta è maggiore del numero dei drive(disco) moltiplicato per la dimensione della fetta **allora** alcuni drive riceveranno richieste multiple.

Svantaggi di R.0:

- 1) Lavora peggio con quei S.O. che normalmente richiedono un settore alla volta.
- 2) Affidabilità potenzialmente peggiore. Se uno SLED ha un fallimento ogni X accessi, e una batteria RAID è composta da 4 di tali dischi, allora essa avrà un fallimento dopo $X/4$ accessi a causa della parallelizzazione.
- 3) Tale metodologia non è ridondante quindi non è un vero e proprio RAID.

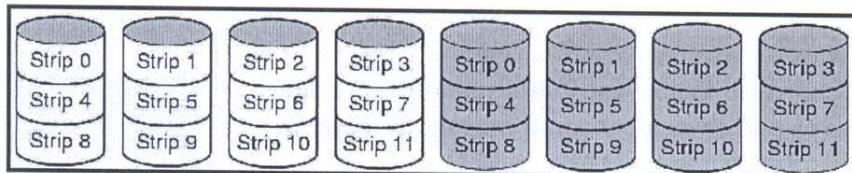
Non posso recuperare i dati in caso di crash del sistema

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I 6 RAID

R1

R.1: I RAID di livello 1 sono dei veri e propri RAID, è il modello più semplice di RAID, in esso è presente una semplice duplicazione dei dischi. Su ciascun set i dati sono gestiti secondo la metodologia R0.



Vantaggi:

- 1) Backup contemporaneo alla scrittura (nessun tempo dedicato al backup)
- 2) Le operazioni di scrittura avvengono contestualmente sulle due posizioni analoghe, non sono peggiori di quelli di uno SLED.
- 3) Le operazioni di lettura avvengono indistintamente dalle due posizioni analoghe, migliorando i tempi di risposta, in molti casi raddoppiano.
- 4) Tolleranza ai guasti eccellente, sostituzione del disco e backup in linea.

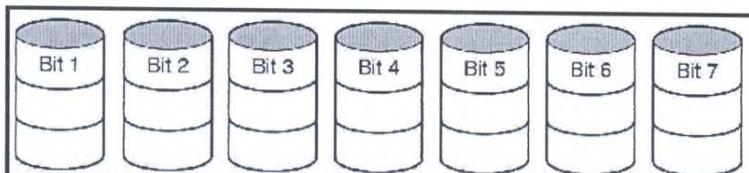
Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I 6 RAID

R2

R.2: Il RAID di livello 2 è analogo ai precedenti con una distribuzione per parola o bit sui singoli dischi, i dischi devono essere meccanicamente sincronizzati.

Esempio: dato un Byte, dividerlo in due parti ciascuna di 4 bit, aggiungere ad ognuna 3 bit di parità secondo la codifica di **Hamming**, distribuire i bit nel relativo disco RAID.



Vantaggi:

- 1) Throughput totale elevato, la scrittura di una parola è parallela.
- 2) Fallimento di un disco non porta a perdite di informazioni, la codifica di Hamming recupererà i dati, inoltre la sostituzione del disco comporterà un tempo di riallineamento del disco distribuito nel tempo (i dati nel nuovo disco saranno caricati dopo il recupero con Hamming).

Svantaggi:

- 1) Sincronizzazione meccanica dei dischi (rotazione e testine).
- 2) Molto lavoro dedicato alla codifica di Hamming.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I 6 RAID

R3

R.3: I RAID di livello 3 sono una esemplificazione dei RAID di livello 2 in cui solo un bit è dedicato alla parità.

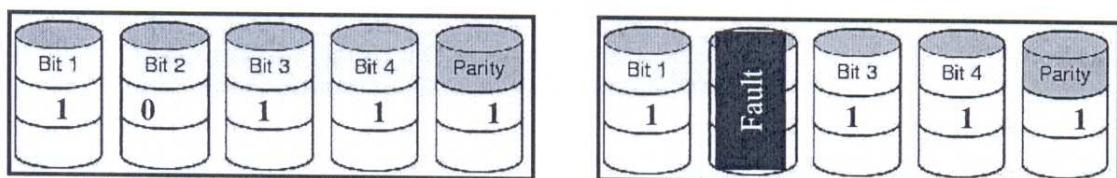
Anche in questo caso vi è una sincronizzazione meccanica.

Gestione dell'errore:

L'obiettivo della correzione dell'errore di R.3 e la gestione dell'errore in caso di crash del disco. In questo caso conoscendo quale bit è andato perduto e valutando il bit di parità si riesce a recuperare l'informazione perduta e quindi l'intero disco.

NON CORREGGE GLI ERRORI CASUALI.

Esempio:



Il Bit di Parità ci garantisce che il bit presente nel disco Fault era 0.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I 6 RAID

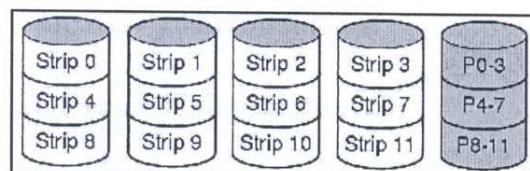
R4

R.4: I RAID di livello 4 non lavora sulla parola bensì su di una fetta, per tale motivo NON dipende da sincronizzazioni meccaniche.

Il disco di parità conterrà una fetta di parità costituita da k fette.

Svantaggi:

- 1) ogni volta che si aggiorna un settore devono essere letti tutti gli altri settori corrispondenti per aggiornare la relativa fetta di parità.



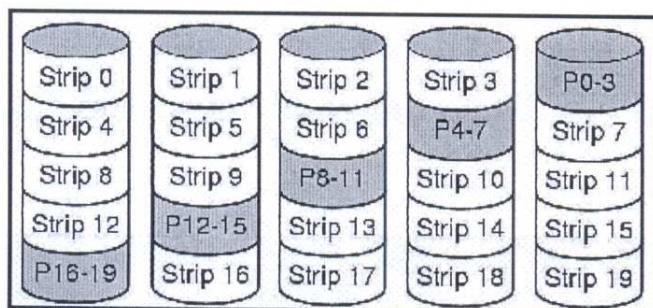
Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I 6 RAID

R5

R.5: I RAID di livello 5 lavora come i RAID di Livello 4 ma distribuisce i Byte di parità sui i dischi.

I disco di parità sarà distribuito sui k dischi.



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Chi dovrebbe usare i RAID?

Coloro che hanno bisogno di gestire grandi quantità di dati e di avere un sistema robusto ai guasti hardware.

Le principali ragioni per usare la tecnologia RAID sono:

- aumento della velocità
- aumento della capacità di archiviazione
- grande efficienza nel recupero in seguito ad un crash del sistema

Distanza di Hamming

- Indica il **numero di bit differenti** che ci sono tra due parole di uguale lunghezza
 - la distanza di Hamming tra 1000 1001 e 1011 0001 è 3;
 - può essere calcolata eseguendo l'**OR esclusivo** tra le due parole (il numero di bit a 1 del risultato indica la distanza)
 - $1000\ 1001 \text{ XOR } 1011\ 0001 = 0011\ 1000$, in cui sono presenti 3 uno.
- Se due parole di codice hanno distanza pari a **d**, ci vorranno **d** errori per convertire l'una nell'altra.
 - un errore singolo provoca la lettura di un dato errato che ha distanza di Hamming pari a 1 dal dato esatto;
 - un errore doppio provoca la lettura di un dato errato che ha distanza di Hamming pari a 2 dal dato esatto;

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Distanza di Hamming

- Codice rilevatore/correttore di errori:
 - parola di **n** bit $\Rightarrow 2^n$ configurazioni **possibili**;
 - parte dati di **m** bit $\Rightarrow 2^m$ configurazioni **legali**.
- La **distanza di Hamming** di un codice è il minimo delle distanze tra tutte le configurazioni legali.
- La proprietà di rilevazione/correzione di un codice dipende dalla sua distanza Hamming:
 - per rilevare un errore di cardinalità **d** serve un codice con distanza **$d + 1$** (**d** errori non possono cambiare una parola di codice legale in un'altra parola di codice legale);
 - per correggere un errore di cardinalità **d** serve un codice con distanza **$2d + 1$** (**d** errori trasformano una parola legale in una parola illegale che è comunque più vicina alla parola legale originaria che non ad altre parole legali).

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Rilevazione errori: parità

- Codice di parità: ai dati si aggiunge un **bit di parità**
 - **parità pari**, il numero di 1 nella parola di codice è pari (e.g. al dato **0010** si aggiunge un **1** per ottenere **00101**);
 - **parità dispari**, il numero di 1 nella parola di codice è dispari (e.g. al dato **0010** si aggiunge uno **0** per ottenere **00100**).
- La distanza tra due parole di codice è almeno **2** (distanza di Hamming del codice)
 - un **errore singolo** produce una parola di codice con **parità sbagliata** (e.g. un errore sul II bit cambia **00101** in **01101**);
 - ci vogliono **due errori** per passare da una parola di codice valida a un'altra parola di codice valida (per passare da **00101** a **11101** sia il I che il II bit debbono essere errati).
- Questo codice consente di **rilevare gli errori singoli**.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Correzione dell'errore

- Esempio di codice per la correzione dell'errore
 - il codice ha quattro parole valide:

A = 0000000000	B = 0000011111
C = 1111100000	D = 1111111111

 - il codice ha una **distanza di Hamming pari a 5**, perciò è in grado di correggere errori doppi.
- Consideriamo un esempio di correzione:
 - si legge **X = 0000000111** che non appartiene al codice;
 - viene calcolata la distanza tra **X** e le parole "legali" del codice: **AX=3, BX=2, CX=8, DX=7**;
 - si presume che in origine la parola fosse la più vicina, cioè **B**
 - se l'errore è doppio l'ipotesi è valida e permette di correggere l'errore;
 - se l'errore è triplo (o superiore) l'ipotesi non è valida e l'errore non viene corretto, ma viene "legalizzato";
 - bisogna assicurarsi che la probabilità di errore triplo sia molto bassa!!

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Correzione di errori singoli

- Consideriamo che la **parola dati sia di m bit** e che quindi ci siano al massimo 2^m **parole "legali"**;
- aggiungiamo **r bit di controllo** ed otteniamo $n=m+r$ **bit complessivi**, ogni parola "legale" è di n bit;
- nell'ipotesi di errore singolo, ognuna delle 2^m **parole "legali"** ha n **parole "illegali"** a **distanza 1**, quindi richiede **$n+1$ configurazioni** (altrimenti non sarebbe possibile correggere l'errore singolo);
- con n bit ci sono al massimo 2^n configurazioni, quindi $(n+1)*2^m \leq 2^n$, cioè $(m+r+1) \leq 2^r$;
- fissato m , questo vincolo definisce il **limite inferiore del numero di bit di controllo**.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

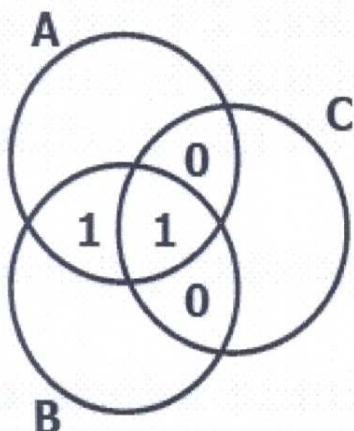
Overhead per la correzione

Bit x parola (m)	Bit x controllo (r)	Bit totali (n)	Overhead (percentuale)
8	4	12	50%
16	5	21	31%
32	6	38	19%
64	7	71	11%
128	8	136	6%
256	9	265	4%
512	10	522	2%

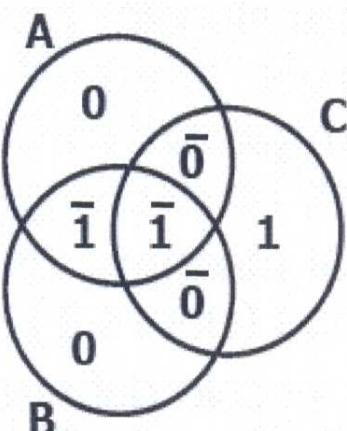
Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Esempio di codice correttore d'errore in parole di 4 bit

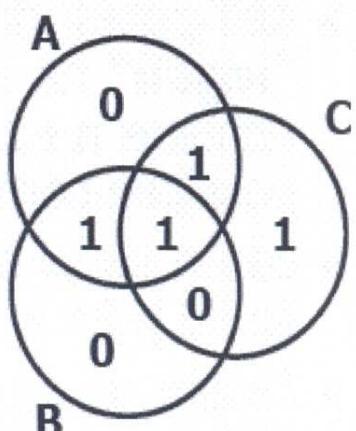
Codifica di 12_{dieci}
come 1100_{due}



Parità pari su
gruppi di 3 cifre
(001 1100)



Errore in AC



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Codice di Hamming /1

- La parola **dati** è di **m** bit, cui vengono aggiunti **r** bit di **codice** formando una parola di **n=m+r** bit.
- I bit vengono numerati da **sinistra a destra** a partire da 1 (il bit più a sinistra è al posto 1, quello più a destra al posto n).
- I bit che si trovano in una posizione corrispondente a una **potenza di 2** ($2^0=1$, $2^1=2$, ...) sono bit di **parità**, gli altri contengono i dati:
 - si consideri per esempio **m = 16** ed **r = 5** (quindi **n = 21**);
 - i bit di **codice** si trovano in posizione **1, 2, 4, 8 e 16**;
 - i bit di **dati** si trovano in posizione **3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20 e 21**.

dato

1	1	1	1
---	---	---	---

0	0	0	0
---	---	---	---

1	0	1	0
---	---	---	---

1	1	1	0
---	---	---	---

codice

	1			111	0000	1010	1110			0										
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Codice di Hamming /2

- Ogni posizione viene scritta come **somma di potenze di due**:
 - per esempio, la posizione 5 corrisponde a $1+4$ cioè 2^0+2^2 ;
 - si tratta della rappresentazione binaria della posizione ($5_{\text{dieci}} = 101_{\text{due}}$);
- Ciascuna posizione è controllata da tutti i **bit di codice** che compaiono nella **sua rappresentazione binaria**:
 - la posizione 5 è controllata dai bit di codice in posizione 1 e 4.
 - il bit di codice in posizione 1 controlla tutte le **posizioni dispari**, cioè 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 e 21;
 - il bit n. 2 controlla i bit n. 2, 3, 6, 7, 10, 11, 14, 15, 18 e 19.
- Ciascun bit di codice è configurato in modo da garantire la **parità pari sull'insieme che controlla**:

dato	1111	0000	1010	1110
codice	1	111	0000	101
	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16

1 → 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21.	2 → 2, 3, 6, 7, 10, 11, 14, 15, 18, 19.	4 → 4, 5, 6, 7, 12, 13, 14, 15, 20, 21.
8 → 8, 9, 10, 11, 12, 13, 14, 15.	16 → 16, 17, 18, 19, 20, 21.	

1 → 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21.
 2 → 2, 3, 6, 7, 10, 11, 14, 15, 18, 19.
 4 → 4, 5, 6, 7, 12, 13, 14, 15, 20, 21.
 8 → 8, 9, 10, 11, 12, 13, 14, 15.
 16 → 16, 17, 18, 19, 20, 21.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

1 → 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21. 2 → 2, 3, 6, 7, 10, 11, 14, 15, 18, 19. 4 → 4, 5, 6, 7, 12, 13, 14, 15, 20, 21.
 8 → 8, 9, 10, 11, 12, 13, 14, 15. 16 → 16, 17, 18, 19, 20, 21.

Codice di Hamming

bit 1	0	1	111	0000	101	0111	0
	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16	17 18 19 20	21	
bit 2	001		111	0000	101	0111	0
	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16	17 18 19 20	21	
bit 4	0010		111	0000	101	0111	0
	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16	17 18 19 20	21	
bit 8	0010	1110		0000	101	0111	0
	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16	17 18 19 20	21	
bit 16	0010	1110		0000	1011	0111	0
	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16	17 18 19 20	21	

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Correzione degli errori

- In caso di errore singolo, **uno o più bit** di parità **risultano errati**.
- Il **bit da correggere** si trova nella **posizione** indicata dalla somma dei **bit** di parità **errati**.

errore nel bit 5	0010	0110	0000	1011	0111	0																
bit 1	1 f	2 f	3 f	4	5 f	6 f	7 f	8	9 ↑	10 ↑	11 ↑	12	13 ↑	14 ↑	15 ↑	16 ↑	17 ↑	18 ↑	19 ↑	20 ↑	21 ↑	errore
bit 2		↑ t	↑ t			↑ t	↑ t															corretto
bit 4			↑ t			↑ t	↑ t	↑ t														errore
bit 8				↑ t		↑ t	↑ t	↑ t														corretto
bit 16									↑ t	↑ t	↑ t	↑ t	↑ t	↑ t	↑ t	↑ t	↑ t	↑ t	↑ t	↑ t	corretto	

I bit di parità errati sono il n. 1 e il n. 4, si corregge il bit $4+1=5$.

È l'unico bit compreso negli insiemi controllati dai bit 1 e 4 ed esterno a tutti gli altri insiemi.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

T I P I D I M E M O R I A D I M A S S A

Compact Disk

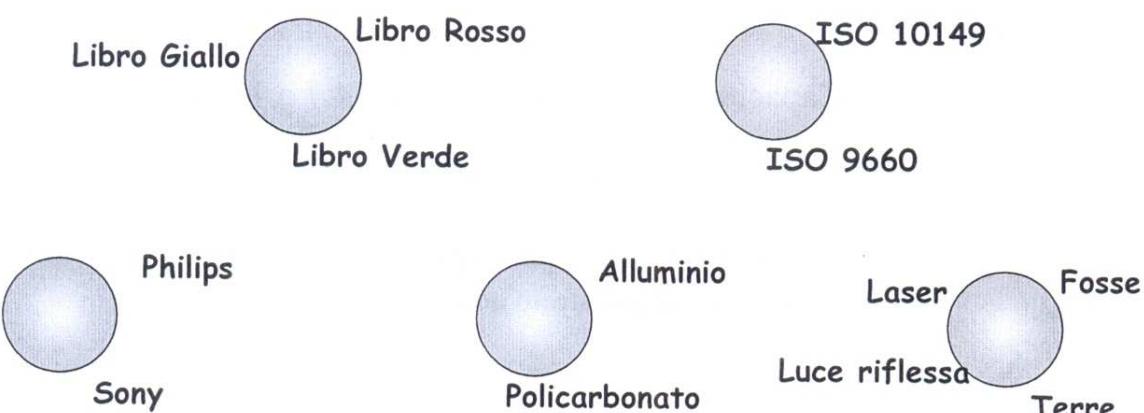
Contrariamente ai dischi magnetici i dischi ottici sono ricoperti da materiale riflettente, i valori 0/1 sono rappresentati dalla riflessione o dalla non riflessione di un fascio di luce (laser) che colpisce una microscopica parte della superficie.

I Compact Disk (CD) rappresentano, in questo momento, la massima diffusione commerciale di tale tecnologia, essi possono essere catalogati in base al numero di volte che è possibile effettuare una memorizzazione.

CD ROM	CD WORM	CD REWRITE
Read Only Memory in essi è possibile effettuare solamente operazioni di lettura	Write Once Read Many in cui è possibile scrivere una sola volta e leggere innumerevoli volte.	In cui è possibile effettuare operazioni di scrittura (<1000), consentendo di modificare le informazioni presenti sul CD
Capacità di memorizzazione		Da 650 a 700 MB

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I CD-ROM



La parte essenziale nelle caratteristiche dei CD-ROM è il file system che rende possibile il loro uso su computer con S.O. diversi.

Il file system per i CD-ROM è costituito da tre livelli:

- 1) **I file:** i loro nomi, nella definizione classica, devono essere inferiori o uguali a 8 caratteri con non più di tre caratteri come estensione, e la loro distribuzione sul disco deve essere contigua.
- Le **directory:** possono essere innestate in non più di 8 livelli e i loro nomi non devono contenere necessariamente estensioni.
- 2) I nomi dei file possono essere estesi sino a 32 caratteri.
- 3) Stesse caratteristiche del livello 2 ma con in più la caratteristica che i file possono non essere attigui.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

DVD: Digital Video Disk

I Digital Video Disk (DVD) sono fisicamente identici ai CD sia in termini di superficie che in termini di spessore, il supporto utilizza la stessa tecnologia con prestazioni superiori.

DIFFERNZE CON I CD:

- Utilizzo di entrambe le facce del disco.
- Capacità di memorizzazione su due strati per faccia.
- Maggiore densità di memorizzazione.
- Fosse più piccole (0,4 micron contro gli 0,8)
- Spirale più stretta (0,74 micron di distanza contro gli 1,6)
- Un Laser Rosso (0,65 micron contro gli 0,78)

DVD: Digital Video Disk

PREGI del DVD:

- Capacità di memorizzazione da 4.7 a 17GigaByte.

1. Lato singolo, strato singolo (4,7GB)
2. Lato singolo, strato doppio (8,5GB)
3. Lato doppio, strato singolo (9,4GB)
4. Lato doppio, strato doppio (17GB)

I dischi a Doppio strato: due strati uno riflettente ed un altro semi-riflettente, la messa a fuoco del laser farà in modo da far rimbalzare il segnale sul primo o sul secondo strato. Le fosse del secondo strato sono leggermente più grandi del primo per una maggiore garanzia di lettura, questo giustifica il fatto che la sua dimensione non è esattamente il doppio di quella a singolo strato.

I dischi a Lato doppio: hanno le stesse caratteristiche di dischi a doppio strato e sostanzialmente sono incollati schiena contro schiena.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Formattazione di un disco

Un disco fisso è composto da una pila di piatti in alluminio, lega, vetro; su di ogni piatto è depositato un sottile strato di ossido di metallo magnetizzabile.

Quindi il disco non contiene nessuna informazione; prima di poter scrivere o leggere qualcosa esso deve essere suddiviso in settori.

Questo processo si chiama processo di **formattazione fisica**, esso assegna al disco speciali strutture di dato (settori) costituite da:

- intestazione o preambolo
- area dati (512 byte)
- coda o ECC (es 16 bit per il test d'errore)

Nell'intestazione e nella coda sono presenti particolari informazioni, numero del settore, codice d'errore.

In particolare il codice di errore viene calcolato ogni volta che si effettua una operazione di lettura o scrittura, per individuare eventuali discrepanze.

La formattazione fisica è effettuata quasi sempre dal costruttore dei dischi.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Formattazione di un disco

Effettuata la formattazione fisica, il S.O. prima di poter scrivere sul disco deve ancora registrare la propria struttura di dati sul disco.

Partizione - permette di suddividere il disco in uno o più gruppi di cilindri, in tal senso il S.O. può trattare ogni gruppo di cilindri come se fosse una unità disco indipendente.

Formattazione Logica di ogni partizione. In tale fase si produce il File System, il S.O. memorizza sul disco le strutture di dato iniziali relative al file system.

Un elaboratore per funzionare necessita di un programma iniziale che inizializza il sistema in tutti i suoi aspetti, tale programma è chiamato **bootstrap**.

Bootstrap:

- 1) Inizializza registri della CPU, controller dei dispositivi
- 2) Avvia il S.O. attraverso il relativo kernel presente sul disco.

La ROM contiene un piccolo programma (bootstrap loader) il cui compito è quello di caricare da disco il programma di avvio completo residente in una partizione fissa del disco detta blocco di boot.

Un disco che contiene tale partizione è detto di sistema o di avvio.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

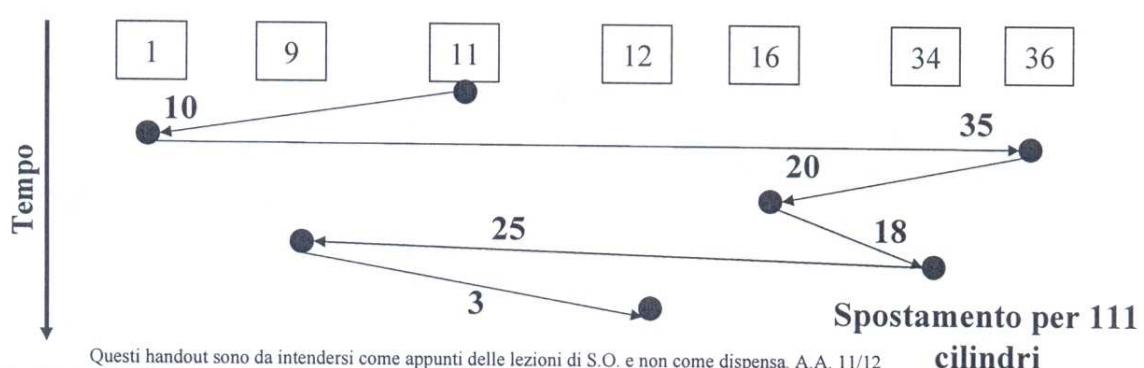
Algoritmi di schedulazione del braccio del disco_a

Problematica: Trovare la migliore strategia per spostare il braccio su cui è posta la testina di lettura/scrittura del disco in relazione alle richieste presentate al controllore del disco.

Esempio:

Al controllore del disco arriva la seguente sequenza di cilindri da leggere o scrivere: 1,36,16,34,9,12,....

La testina è in posizione 11 Con uno schedulatore First-Come First Served si avrebbe il seguente movimento del braccio:



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

329

Algoritmi di schedulazione del braccio del disco_b

- 1) Definiamo **tempo di posizionamento** o SEEK il tempo necessario per spostare la testina sul cilindro richiesto (movimento lineare del braccio).
- 2) Definiamo **tempo di rotazione** il tempo necessario affinché la rotazione del disco faccia passare il settore sotto la testina (movimento rotatorio del disco).
- 3) Definiamo il **tempo di trasferimento** dati il tempo necessario per il trasferimento dei dati.

Dopo aver definito tali quantità possiamo migliorare l'algoritmo FCFS con opportuni accorgimenti:

Durante il tempo necessario per effettuare la rotazione del disco e la traslazione delle testine possono arrivare ulteriori richieste al disco, tali richieste vengono memorizzate in un buffer e valutarle quando l'operazione di lettura o scrittura è terminata.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Algoritmi di schedulazione del braccio del disco_c

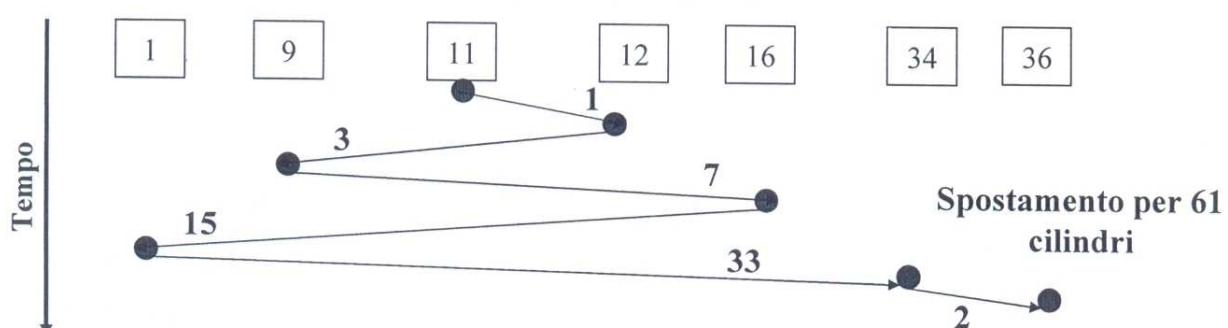
L'algoritmo **Shortest Seek First (SSF)** effettua una valutazione sui cilindri richiesti, ossia individua tra le richieste presenti nel buffer del disco quella che risulta più vicina alla posizione attuale (lo spostamento minimo da effettuare dalla posizione in cui si trova la testina).

Esempio:

Nel Buffer sono memorizzati i seguenti indirizzi di cilindri da leggere o scrivere:

12,9,16 ,1,34,36,....

Lo schedulatore SSF schedula seguendo la sequenza:



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Algoritmi di schedulazione del braccio del disco_d

Lo schedulatore SSF tende a privilegiare le posizioni centrali del disco, in questo modo le parti del disco molto esterne o interne (vedi cilindro 90) tenderanno ad aspettare molto tempo prima di essere servite.

Conflitto tra tempo minimo di risposta e equità di servizio

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Algoritmi di schedulazione del braccio del disco_e

L'algoritmo dell'ascensore si basa sul tenere traccia della direzione su cui si sta muovendo il braccio (UP, DOWN), esso mantiene la direzione corrente sino a quando non esistono valori superiori nel caso di direzione UP o valori inferiori nel caso di direzione DOWN.

In generale, l'algoritmo dell'ascensore peggiora leggermente il tempo di risposta ma migliora decisamente l'equità del servizio.

Esempio:

Supponiamo che nel Buffer sono memorizzati i seguenti indirizzi di cilindri da leggere o scrivere:

12,9,16 ,1,34,36,....

Lo schedulatore dell'ascensore schedula seguendo la sequenza:



Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Algoritmi di schedulazione del braccio del disco_f

Varianti agli algoritmi

Unica direzione: si suppone che l'ultimo cilindro sia collegato al primo.

Cache sul disco: i ritardi di lettura e scrittura possono essere eliminati inserendo una cache sul controller del dispositivo, in essa vengono caricate le informazioni dei cilindri attigui a quello letto.

Nota: La cache del controllore del disco non ha niente a che vedere con quella della memoria.

Cache multiple: Più dischi sullo stesso controllore impongono più cache per la bufferizzazione, una per disco.

Geometria: Si suppone che la geometria reale del disco sia uguale a quella virtuale.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

I CLOCK



I clock:

- Essenziali per qualsiasi S.O.;
- Mantengono data e ora;
- Evitano l'acquisizione perenne della CPU da parte di un processo (quanto di tempo);
- Dispongono di un driver per il loro funzionamento.

Il clock come componente Hardware:

- 1) Il clock legato all'alimentazione (110, 220volt).
- 2) Il clock legato ad un quarzo.

Il clock come componente Software:

- 1) Il drive del clock realizza quelle operazioni necessarie all'elaboratore per compiere le istruzioni impartite dai processi.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Il Clock come componente Hardware_a

- 1) **Il Clock basato sull'alimentazione** diffuso nei decenni scorsi si riferiva alla frequenza di alimentazione e più precisamente inviava una interruzione ogni ciclo di voltaggio (50,60 Hz).
- 2) **Il Clock basato sul quarzo** è costituito da tre componenti: un cristallo di quarzo tagliato opportunamente, un contatore e un registro di caricamento.

Il clock basato sul quarzo è il clock presente sugli attuali elaboratori, il suo compito è di fornire un segnale di sincronizzazione alle varie componenti del computer.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Il clock come componente Hardware_b

Il quarzo tagliato opportunamente, messo sotto tensione è in grado di generare un segnale estremamente preciso dell'ordine delle centinaia di Mhz che opportunamente moltiplicato con appositi componenti elettronici può raggiungere le migliaia di Mhz.

Il segnale emesso dal quarzo viene mandato a un contatore che decrementa il suo contenuto sino a raggiungere lo zero, causando quindi un'interruzione della CPU.

Clock così progettati possono essere programmati, si pensi per esempio alla possibilità di variare il valore del registro di caricamento....posso gestire il tempo tra un'interruzione e l'altra.

- 1) one-shot-mode
- 2) Square-wave-mode

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Il clock come componente Hardware_c

One-shot-mode

1. Allo start, il valore del registro di caricamento viene trasferito nel contatore e quindi comincia il suo decremento.
2. Quando il contatore raggiunge lo zero causa una interruzione e resta fermo fino a quando il software non lo fa ripartire.

Square-wave-mode

Dopo essere arrivato allo zero e aver causato l'interruzione, il valore del registro di caricamento è ricopiato nel contatore e nuovamente viene decrementato sino a al raggiungimento dello zero; il ciclo viene eseguito un numero infinito di volte.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Il clock come componente Software_a

L'Hardware del clock produce esclusivamente una o più interruzioni, la gestione del clock è compito del Software.

Il software del clock è identificato con il driver, i suoi compiti variano da sistema a sistema e un insieme di funzioni comuni a tutti i sistemi può essere riassunto come segue:

1. Mantenere la data e l'ora.
2. Evitare che un processo resti attivo più a lungo di quanto dovrebbe.
3. Tenere la contabilità dell'uso della CPU (CPU accounting).
4. Gestire le chiamate di sistema da parte dei processi utente.
5. Mettere a disposizione dei timer di controllo per parti del sistema operativo.
6. Monitoraggio e raccolta statistiche.

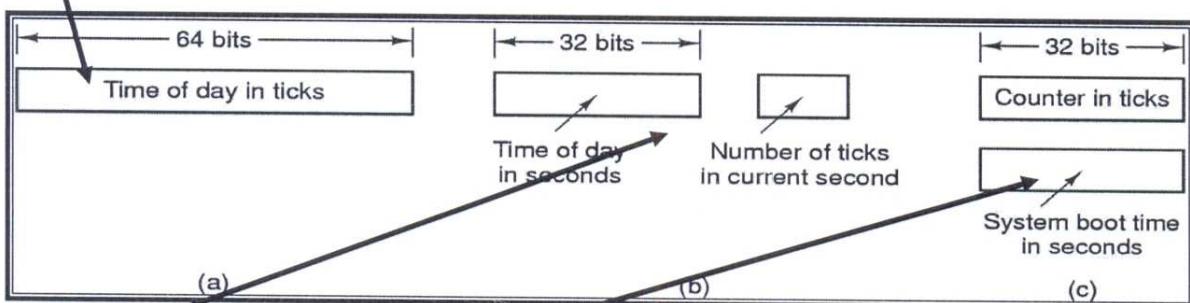
Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Il clock come componente Software_b

Esempi di come vengono trattati le funzioni

Prima Funzione(Mantenere la data e l'ora):

- a. Il tempo reale o la data e l'ora sono memorizzati in un registro e 64 bit in termini di tic (interruzioni periodiche o battiti di clock). Con questa strategia si possono gestire fino a due anni e quindi se lo starting point (es. il 1 gennaio 1970) non è sufficiente.



- b. Mantenere il tempo reale in secondi piuttosto che in tic, un contatore ausiliario mantiene i tic sino allo scadere del secondo.
- c. Mantenere i tic ma non contarli in relazione agli elementi esterni ma relazionarli alle accensioni del sistema. (Tempo di Boot) Quindi si conta il tempo relativo alla sessione di boot odierna e per ottenere il tempo assoluto si effettua come somma tra quello relativo e quello di backup.

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12

Il clock come componente Software_c

Seconda Funzione(Evitare che un processo resti attivo più a lungo di quanto dovrebbe):

Quando si fa partire un processo, lo scheduler inizializza un contatore con un valore corrispondente al quanto di tempo(in tic) che gli è stato assegnato. Ad ogni clock il driver decrementa questo contatore di una unità. Quando il contatore arriva a zero, il driver richiama lo scheduler, per far sì che venga mandato in esecuzione un altro processo.

Terza Funzione(CPU accounting):

Un secondo timer, diverso da quello del timer principale del sistema. Quando un processo viene caricato sulla CPU il contatore è azzerato e incrementato direttamente dal clock, quando il processo esce dalla CPU si può leggere il suo contenuto.

Indice degli Argomenti

342

- Inviare comandi ai dispositivi,
- Catturare le interruzioni,
- Trattare le condizioni di errore,
- Interfaccia tra i dispositivi fisici e il resto del sistema.
- Comunicazione Tra CPU e Registri del Controllore dell'I/O Mappato in Memoria
- DMA
- I/O programmato;
- I/O guidato dalle interruzioni del dispositivo;
- I/O tramite DMA
- Software del S.O. indipendente dal dispositivo
- I **R A I D**
- Codice di Hamming
- Algoritmi di schedulazione del braccio del disco
- I **C L O C K**

Questi handout sono da intendersi come appunti delle lezioni di S.O. e non come dispensa. A.A. 11/12