Gestione software
Il meccanismo delle interruzioni - 3

- Il valore emesso sul canale dal controllore delle interruzioni designa il servizio richiesto alla CPU
- Tale valore è un indice nel **vettore delle interruzioni**, contenente l'indirizzo della prima istruzione della procedura di servizio
- La base del vettore delle interruzioni è fissa e *nota* al S/O

Gestione software
Il meccanismo delle interruzioni - 4

- L'esecuzione della procedura di servizio *interrompe* l'attività corrente della CPU
- Per poter successivamente riprendere tale attività, occorre salvare l'informazione necessaria per ripristinare il corretto *contesto di esecuzione*
  - Cosa determina il contesto (minimo)
  - Dove salvarlo

Gestione software
Il meccanismo delle interruzioni - 5

- Minimizzazione del contesto da salvare
  - Per minimizzare il costo temporale di attivazione del servizio di interruzione e del successivo ripristino
    - Più lungo è il tempo di attivazione, più lungo è il periodo nel quale il controller di interruzioni *non* presta ascolto ai dispositivi controllati
  - Il contesto minimo comprende il valore corrente di IP e quello dei registri di configurazione del processo (p.es. base della tabella dei descrittori di segmento; registro di segmento)

Gestione software
Il meccanismo delle interruzioni - 6

- Localizzazione dell'area di salvataggio
  - Deve essere *sicura*
    - Una eccezione di accesso durante un servizio di interruzione sarebbe disastrosa
    - Meglio fidarsi del S/O piuttosto che del processo interrotto (di utente)
  - *Non* deve essere *costosa* da accedere
    - L'accesso a zone di memoria di S/O può richiedere l'aggiornamento dei registri di configurazione ed il relativo caricamento da memoria secondaria

Gestione software
Il meccanismo delle interruzioni - 7

- Quale è la relazione tra lo stato interno della CPU e l'arrivo (asincrono) di una interruzione?
  - L'interruzione dovrebbe idealmente avere effetto al confine tra due istruzioni successive
  - Le CPU con *pipeline* sovrappongono l'esecuzione di molte istruzioni
    - Confine meno chiaro
  - Le CPU super-scalari (*out-of-order execution*) modificano l'ordine di esecuzione specificato dal programma
    - Confine totalmente confuso

Gestione software
Il meccanismo delle interruzioni - 8

- Le **interruzioni precise** hanno 4 proprietà
  - L'IP viene salvato in un luogo noto e sicuro
  - Tutte le istruzioni precedenti ad IP sono state completate
  - Nessuna istruzione successiva ad IP è stata completata
    - Possono essere state emesse ma il loro effetto deve essere annullato
  - Lo stato di esecuzione dell'istruzione designata da IP è noto

Gestione software
Il meccanismo delle interruzioni - 9

- Modelli di interruzioni che non abbiano tali proprietà sono detti *imprecisi* e sono estremamente complessi da gestire per il S/O
- Architetture super-scalari come il Pentium Pro sono capaci di offrire interruzioni precise al prezzo di una logica assai complessa
  - Questo è uno dei costi della *backward compatibility*
- Alcune architetture consentono di disabilitare la modalità imprecisa ma le prestazioni crollano

Gestione software
Onere di interruzione (overhead)

F = frequenza di clock della CPU = 200 MHz

NP = numero cicli di clock per un servizio di base di interruzione = 300

= con trasferimento di 512 B (1 settore) = 2.860

NL = numero interruzioni al secondo =

- Mouse = 30
- Floppy = 50 kB/s ÷1 settore = 100
- Hard disk = 4 MB/s ÷ 1 settore ≈ 8.000

PO = onere di interruzione (*overhead*) = (**NL**×**NP**)/**F**

Mouse = 0,0045% **Floppy = 0,1%** **Hard disk = 11%**

Gestione software
Onere di DMA (overhead)

F = frequenza di clock della CPU = 200 MHz

NP = cicli di clock per inizializzazione di DMA = 800
= per trasferimento di 512 B (1 settore) = 1024
in *burst mode* (sottraendo cicli alla CPU)

NL = numero trasferimenti DMA al secondo =
- Floppy = 50 kB/s ÷1 settore = 100
- Hard disk = 4 MB/s ÷ 1 settore ≈ 8.000

PO = onere di DMA (*overhead*) = (**NL**×**NP**)/**F**
Floppy = 0,05% **Hard Disk = 4,2%**

Gestione software
Classificazione

- Il metodo più appropriato di gestione software dell'I/O dipende dal *tempo di risposta* del dispositivo
  - Dispositivi più lenti della CPU
  - Dispositivi veloci come la CPU
  - Dispositivi più veloci della CPU
  - Dispositivi a velocità variabile
- Occorre anche tenere conto della *banda passante*

Gestione software
Periferiche lente

- Occorre fornire una coda *ampia* all'ingresso del dispositivo e *disaccoppiare* l'attività di I/O dalla sua gestione
  - Dimensionamento della coda
  - Scelta progettuale tra
    - **Gestione programmata dell'I/O**
      - Con controllo esplicito a software **(*polling / busy waiting)***
    - **Gestione dell'I/O a domanda**
      - Con controllo software solo quando richiesto (*interrupt-driven*)

Gestione software
Periferiche veloci

- Non richiedono particolari accorgimenti in quanto i tempi di risposta sono confrontabili a quelli della CPU
  - Richiedono una coda *piccola* per disaccoppiare meglio l'operazione di I/O dalla sua gestione
  - Consentono l'uso sia della tecnica del *polling* che di quella dell'*interrupt*, con prestazioni analoghe

Gestione software
Periferiche molto veloci

- Richiedono una coda *ampia* in uscita dal dispositivo ed il trasferimento *diretto* tra dispositivo e memoria
  - Gestione dell'I/O mediante **DMA**
    - Minimo coinvolgimento software ad onere contenuto
  - La CPU perde il controllo assoluto dei canali di comunicazione (*bus*)
  - Occorrono canali molto ampi e veloci per il trasferimento diretto

Gestione software
Modalità polling (busy waiting)

- La CPU ciclicamente interroga il dispositivo per verificarne lo stato di libero e l'eventuale esito dell'operazione
  - Questa modalità forza *sincronizzazione* tra I/O e gestore software
  - La CPU ha controllo *totale* sull'I/O
  - La CPU dedica tempo di esecuzione al controllo dello stato e del progresso delle operazioni di I/O
    - Spesso a vuoto se il dispositivo è molto lento

Gestione software
Onere di polling (overhead)

F = frequenza di clock della CPU = 200 MHz

NP = numero cicli di clock per lettura (*poll*) = 200

NL = numero letture necessarie al secondo =

- Mouse = 30
- Floppy = 50 kB/s emessi su canale a 16 bit = 25.000
- Hard disk = 4 MB/s emessi su canale a 32 bit = 1.000.000

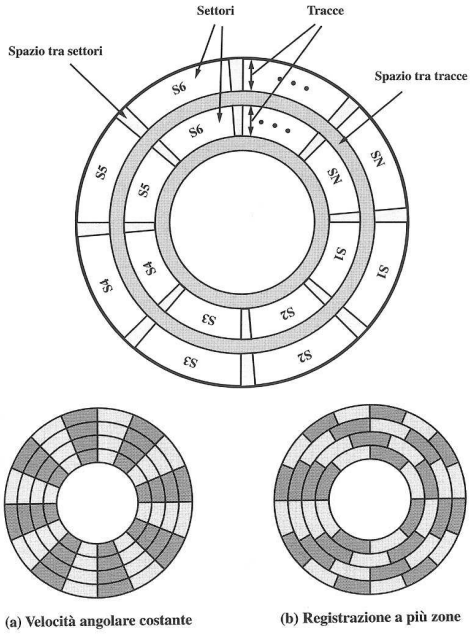
PO = onere di polling (*overhead)* = (**NL**×**NP**)/**F** [%]

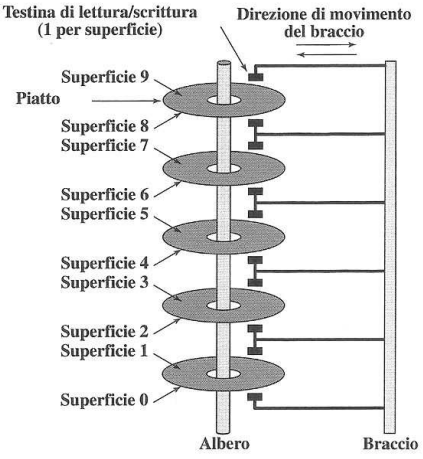
Mouse = 0,003% **Floppy = 2,5%** **Hard disk = 100%**

Supporti su disco
Dischi magnetici - 1 (*Cap. 6, Stallings*)

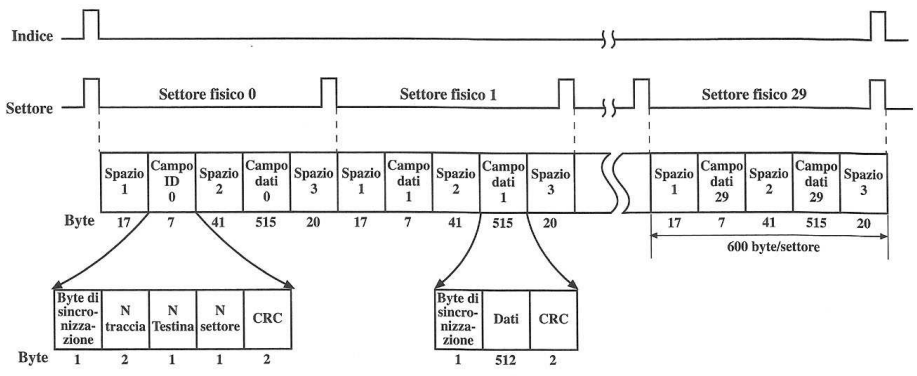
- Memoria secondaria o di livello superiore
- Adibiti alla memorizzazione *permanente*
- Grande capacità e relativa lentezza
  - *Floppy disk*
    - Di facile trasporto e di costo contenuto
  - *Hard disk*
    - Di maggiore capacità e velocità di accesso (più piatti in parallelo)

Supporti su disco:
Dischi magnetici - 2





Supporti su disco:
Dischi magnetici - 3



Supporti su disco:
Dischi magnetici - 4

Tabella 6.2 Parametri di un tipico drive di disco fisso.

Caratteristiche	Seagate Barracuda 36ES	Seagate Cheetah X15-36LP	Seagate Barracuda 36ES	Toshiba HDD1242	IBM Microdrive
Applicazione	Server ad alta capacità	Server ad alte prestazioni	Desktop di fascia bassa	Portatili	Palmari
Capacità	181,6 GB	36,7 GB	18,4 GB	5 GB	1 GB
Tempo minimo di posizionamento da traccia a traccia	0,8 ms	0,3 ms	1,0 ms	–	1,0 ms
Tempo medio di posizionamento	7,4 ms	3,6 ms	9,5 ms	15 ms	12 ms
Velocità dell'albero	7200 rpm	15K rpm	7200	4200 rpm	3600 rpm
Ritardo medio rotazionale	4,17 ms	2 ms	4,17 ms	7,14 ms	8,33 ms
Velocità massima di trasferimento	160 MB/s	522 to 709 MB/s	25 MB/s	66 MB/s	13,3 MB/s
Byte per settore	512	512	512	512	512
Settori per traccia	793	485	600	63	–
Tracce per cilindro (numero di superfici dei piatti)	24	8	2	2	2
Cilindri (numero di tracce su un lato del piatto)	24247	18479	29851	10350	–

Supporti su disco
Dischi magnetici - 5

T_A = Tempo medio di risposta (*access time*)

$$\mathbf{T_A = T_S + T_L}$$

T_S = Tempo di posizionamento delle testine sopra la traccia desiderata (*seek time*)

5-20 ms difficilmente riducibile

T_L = Arrivo del settore sotto la testina (*latency time*)

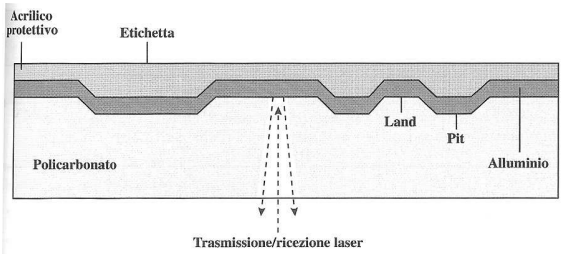
Dipende dalla velocità di rotazione

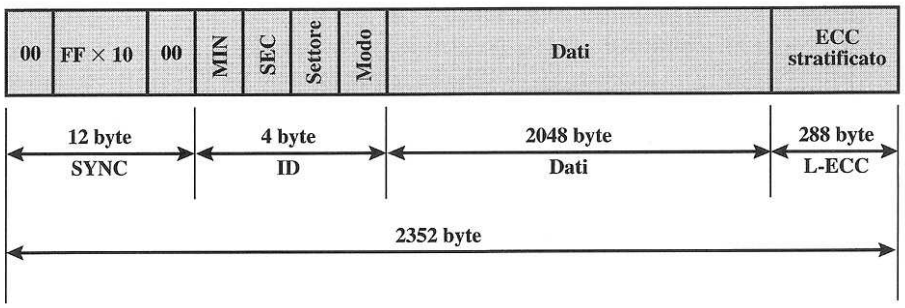
Esempio

RPM=3600 ⇒ RPS=60 ⇒ 1 rotazione ≈16.7ms ⇒ T_L=8.35ms

Supporti su disco:

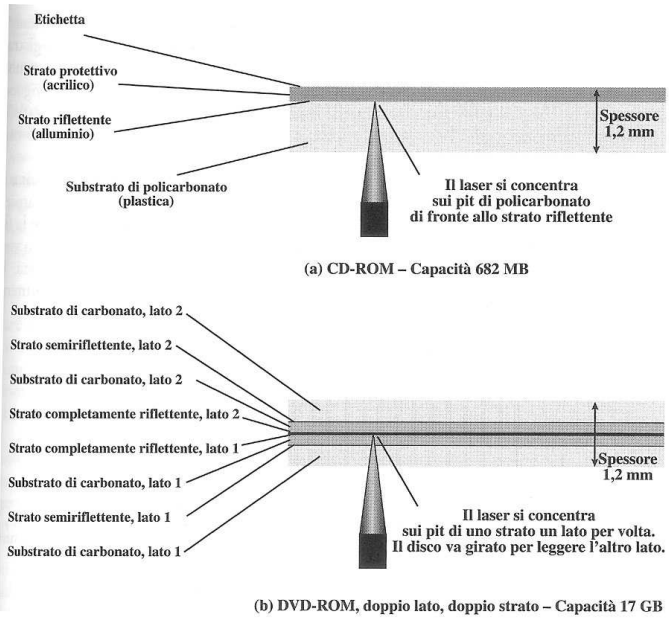
Dischi ottici - 1



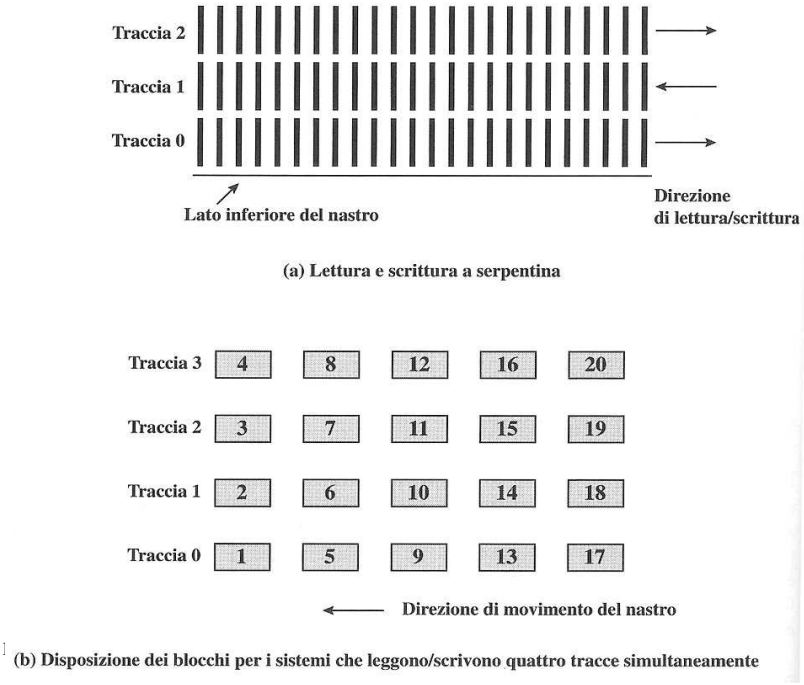


Supporti su disco:

Dischi ottici - 2



Supporti su nastro



Orologi - 1

(*Cap. 5, Tanenbaum*)

- 2 funzioni fondamentali
  - Misurano il trascorrere del tempo di sistema
  - Misurano la durata dell'utilizzo di risorse critiche da parte di processi
- Usano un *hardware* speciale, visto come dispositivo di ingresso
- Richiedono un *software* di gestione
  - Tipicamente posto sotto il controllo del S/O

Orologi - 2

- Visione *hardware*
  - Modello rudimentale
    - Dispositivo collegato alla linea di alimentazione
    - Una interruzione ad ogni ciclo di voltaggio (50 - 60 Hz)
  - Modello avanzato, programmabile

Orologi - 3

- Il segnale periodico emesso dal cristallo viene moltiplicato (per amplificazione) in modo da generare la frequenza desiderata
  - Segnale distribuito all'intero sistema a fini di sincronizzazione
- Ogni emissione amplificata di segnale causa un decremento nel registro contatore il cui valore iniziale è fissato dal registro programmatore

Orologi - 4

- Quando il registro contatore vale 0 si produce una interruzione verso la CPU
  - **Modalità non ripetitiva** (*one-shot*)
    - Inizializzazione → Decremento → Arresto → Riprogrammazione
  - **Modalità ciclica** (*square-wave*)
    - Inizializzazione → Decremento → Ricaricamento automatico
    - Produce una interruzione periodica detta *clock tick*
- Frequenza di interruzione controllata a *software*
  - Cristallo ad 1 GHz → 1 decremento ogni 1 *ns*
  - Registro contatore a 32 *bit* (*unsigned*) → interruzioni con periodo programmabile tra 1 *ns* e 8,6 *s*

Orologi - 5

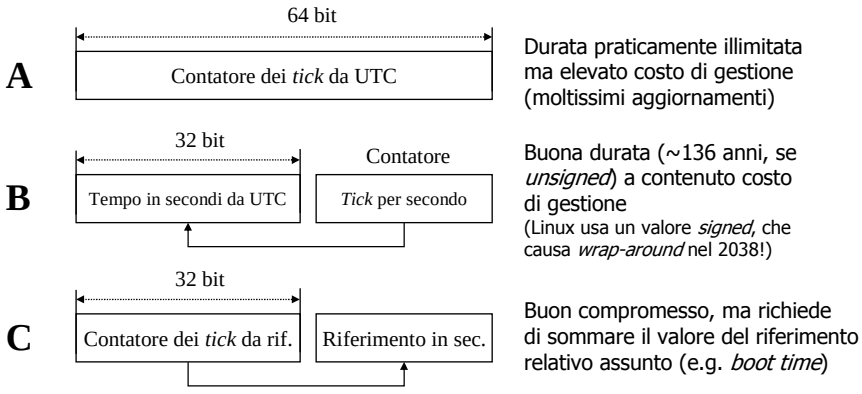
- Un singolo dispositivo può contenere svariati orologi programmabili con diverse opzioni
  - Contatore ad incremento; interruzioni disabilitate
- Un orologio di riserva (a basso consumo) alimentato a batterie mantiene il tempo attuale ad elaboratore spento
- Il valore del tempo corrente si misura come trascorso dall'*Universal Time Coordinated* (**UTC**)
  - **Linux** 12:00 AM del 01/01/70 (in sec.)
  - **Windows** 12:00 AM del 01/01/80

Orologi - 6

- Un gestore *software* specifico (*clock driver*) associa azioni significative alle interruzioni generate dall'orologio *hardware*
  - Avanzamento del tempo corrente (*real time*)
    - Gestione dell'*overflow* di registro
  - Controllo del tempo di esecuzione dei processi
    - Utile per ordinamento a quanti e prevenzione di eccessi
  - Verifica della situazione del sistema
    - Statistiche di gestione, monitoraggi periodici
  - Gestione di meccanismi di allarme (*watchdog*)

Orologi - 7

- Gestione dell'avanzamento del tempo corrente



Orologi - 8

- Controllo del tempo di esecuzione
  - Decremento del quanto
  - Orologio virtuale privato di ogni processo
    - Avanza per ogni *tick* che il processo è in esecuzione (accuratezza a costo non trascurabile)
- Gestione di meccanismi di allarme
  - Lista ordinata inversamente alla distanza delle richieste