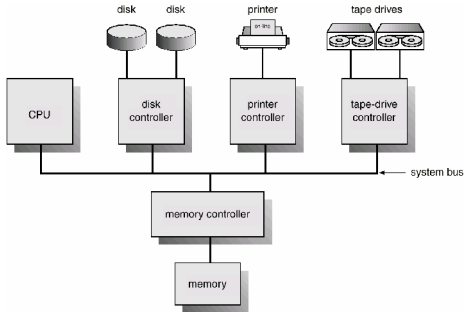
CAP. 2 – STRUTTURE DEI SISTEMI DI CALCOLO

Architettura di un sistema di calcolo

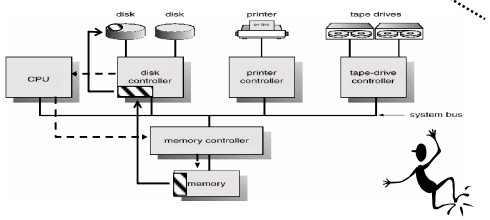


Caratteristiche generali…

- Dispositivi di I/O e CPU possono operare allo stesso tempo e competere per ***l'accesso a memoria*** (regolato dal controller di memoria, unico nel suo genere)
- I ***controller*** sono la parte ***intelligente*** di un dispositivo
- Ogni ***controller*** e' predisposto per un certo ***tipo di dispositivo***
- Esso ha ***registri*** e ***buffer locali*** per gestire le interazioni con le altre parti del sistema
- Il ***device driver*** e' la controparte del dispositivo in OS, e cioe' la ***parte di OS*** predisposta a interagire con un certo dispositivo

… e meccanismi di I/O

- La ***CPU*** sposta i dati da/a ***memoria principale*** a/da ***buffer del dispositivo***
  - ***I/O*** avviene da/a ***buffer del dispositivo*** a/da ***dispositivo***
  - Il ***controller*** informa la CPU, mediante un ***interrupt***, che ha ***finito*** l'operazione



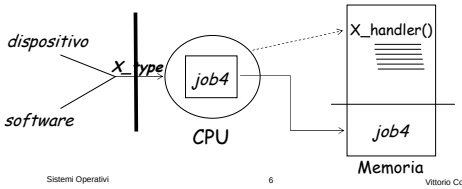
Mai sentito parlare di *Interrupt* ?!

… mi spiace, ma la risposta corretta era SI !

- OS e' ***interrupt driven***!
- ***Perche'?***
  - Interrupt regolano le interazioni tra OS e dispositivi e tra OS e programmi utente
- ***Esempi***
  - Un interrupt viene generato dal controller di un disco quando un'operazione di scrittura e' terminata
  - Una ***trap*** (interrupt generato via software) puo' essere causata da un errore (ex. divisione per 0) o dalla richiesta esplicita (da parte di un programma utente) di un servizio del OS (system call, ex. ***mkdir***)

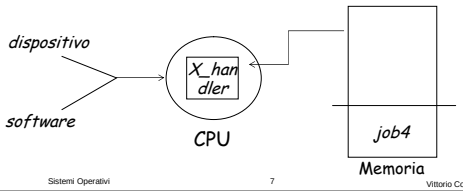
Gestione di un interrupt

- Salvataggio dell'indirizzo dell'istruzione corrente e di tutti i registri della CPU essenziali per il ***ripristino***
- Interrupt successivi sono ***disabilitati*** durante la gestione di un interrupt per evitare perdite
- Differenti segmenti di codice (routine di gestione dell'interrupt) determinano le azioni da intraprendere per ogni ***tipo di interrupt***



Gestione di un interrupt

- Salvataggio dell'indirizzo dell'istruzione corrente e di tutti i registri della CPU essenziali per il ***ripristino***
- Interrupt successivi sono ***disabilitati*** durante la gestione di un interrupt per evitare perdite
- Differenti segmenti di codice (routine di gestione dell'interrupt) determinano le azioni da intraprendere per ogni ***tipo di interrupt***



Determinare il tipo di interrupt

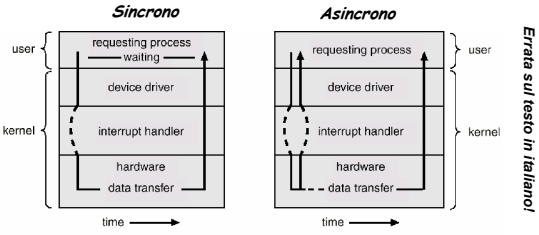
POLLING

La CPU riceve un ***anonimo interrupt*** e quindi ***interroga tutti i dispositivi*** per vedere chi ha da eseguire un'operazione di I/O e quindi esegue la routine appropriata

VECTORED INTERRUPT SYSTEM

La CPU riceve un ***interrupt di un certo tipo*** e, mediante l'interrupt vector, manda in esecuzione la routine appropriata.
Interrupt vector : mappa tipi di interrupt a indirizzi di routine di gestione interrupt

Due modi di fare Input/Output



Dopo che I/O inizia, il controllo ritorna al processo utente solo ***al completamento del I/O***

Dopo che I/O inizia, il controllo ritorna al processo utente ***senza attendere il completamento del I/O***

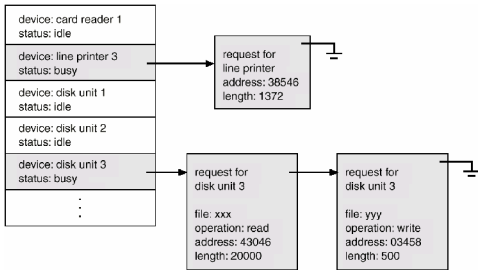
Errata sul testo in italiano!

… e loro caratteristiche

- ***Sincrono***
  - Al piu' ***una richiesta*** di I/O ***alla volta***

- ***Asincrono***
  - Anche un solo processo puo' originare piu' richieste contemporanee:
    - **Device-status table** *contiene una entry per ogni dispositivo di I/O che ne indica il tipo, l'indirizzo e lo stato*
    - **OS accede alla table** *per determinare lo stato di un dispositivo e per modificare la table entry in caso di interrupt*

Device-Status Table…



… e sua gestione

- ***Code di richieste*** in attesa su dispositivi
- Quando arriva un interrupt al OS per un certo dispositivo, OS va a ***modificare*** semplicemente ***la coda***
- ***In uscita*** da un'operazione di I/O il ***controllo*** puo' essere ***ridato*** al processo che stava in esecuzione prima dell'interrupt, a quello che stava in attesa di completamento di questa operazione, oppure a un altro processo

Qualche eccezione

Tastiera

Consente di ***inserire informazioni prima*** ancora che il programma cui sono diretti ne faccia ***richiesta***

Controller della Tastiera

- Segnala la ***presenza di informazioni*** e nel frattempo…
- … utilizza un ***buffer locale*** per conservare le informazioni inserite ***in attesa della corrispondente richiesta***

Direct Memory Access (DMA) Structure

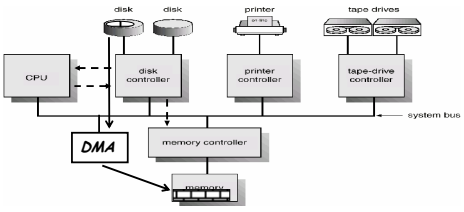
PROBLEMA : Device ***troppo veloci*** con frequenti interrupt prenderebbero ***troppa CPU***

IDEA : ***un unico interrupt per un insieme piu' ampio di informazioni***

Direct Memory Access (DMA) Structure

COME :

Dopo avere settato buffers, puntatori e counters, il controller spedisce un insieme di blocchi di dati dal suo buffer alla memoria (o viceversa) senza intervento ulteriore della CPU



Struttura astratta della *memoria*

- ***Memoria centrale***
  - il solo dispositivo di memoria al quale la CPU puo' ***accedere direttamente***
- ***Memoria secondaria*** (di massa)
  - Estensione della memoria centrale che fornisce grande capacita' di ***memoria non volatile***

Memoria centrale…

(Quasi) tutto cio' che "accade" in un sistema di calcolo passa per la memoria centrale

Load e ***store*** per trasferire da memoria a registri della CPU e viceversa

Memoria ***volatile***, ad ***accesso veloce***, dalla ***capacita' limitata***

Dal punto di vista OS, si tratta di gestire le possibili ***sequenze di indirizzi*** generate per l'accesso a memoria

… e rapporto con I/O

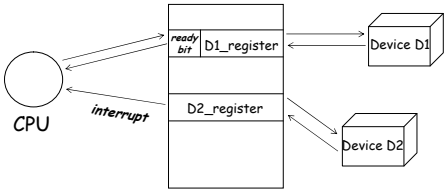
I/O mappato in memoria

Registri dei dispositivi mappati a indirizzi di memoria

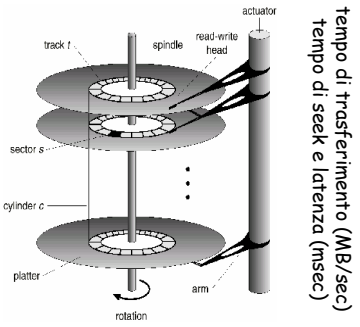
- per dispositivi con ***brevi tempi di risposta***, come un video controller (locazioni di memoria che corrispondono a posizioni sullo schermo)
- per ***porte*** seriali/parallele, in quanto e' facile mappare una porta ad un indirizzo di memoria

Modalita' di interazione in I/O mappato

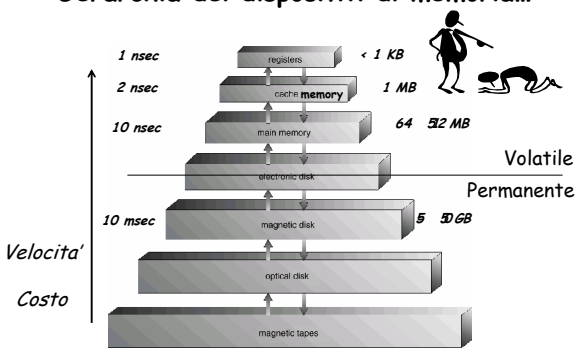
- la CPU fa polling su un ready bit (***programmed I/O***)
- la CPU riceve un interrupt quando il dispositivo e' pronto (***interrupt driven I/O***)



Esempio di *memoria di massa*: il disco



Gerarchia dei dispositivi di memoria…

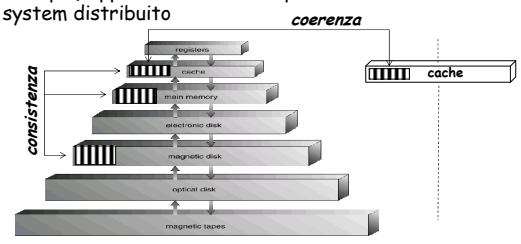


… qualche osservazione …

- ***Electronic disk*** e' accessibile come un file system (volatile o non volatile alimentato a batteria)
- ***Cache*** interamente gestite ***via hardware*** (ex., next instruction register)
- ***Movimento*** tra i vari livelli ***sia implicito che esplicito***
- In generale ***caching*** e' l'operazione di ***copia di informazioni in dispositivi di memoria piu' veloci*** (es. la memoria centrale puo' essere vista come una cache per la memoria secondaria)

… e qualche attributo

- ***Consistenza*** tra copie dello stesso item in diverse memorie
- ***Coerenza*** tra diverse cache (in un multiprocessor, per esempio) oppure tra diverse repliche di un file in un file system distribuito



Hardware Protection

1. Dual-Mode Operation
2. I/O Protection
3. Memory Protection
4. CPU Protection

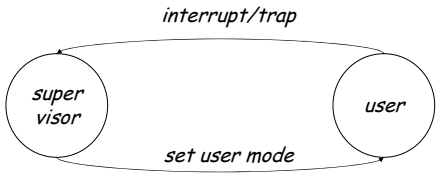
1. Dual-Mode Operation…

In un sistema a condivisione di risorse OS deve assicurare che ***un programma non possa danneggiare*** (non solo se' stesso ma anche) ***gli altri programmi***

Gestione "normale" di errore: emissione di una ***trap*** , dump del core

Supporto hardware per avere (almeno) due modi di operare:
Modo utente- da parte di un utente
Modo supervisor- da parte di OS

… e relative transizioni



Le ***istruzioni*** "pericolose" (***privilegiate***) possono essere eseguite ***solo in modo supervisor***

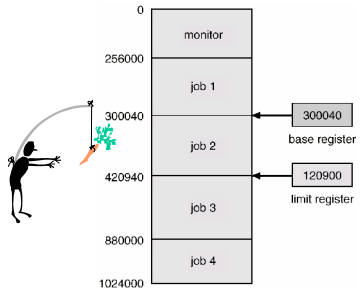
2. I/O Protection

- Regola: un programma ***utente*** non deve ***mai guadagnare*** controllo del computer in ***supervisor mode***
- Come potrebbe accadere:
  - ***scrivendo*** un ***indirizzo*** che appartiene al proprio spazio di indirizzi nell'interrupt vector
  - ***modificando*** il ***codice*** di un interrupt handler per deviarlo ad eseguire nel proprio spazio di indirizzi
- Soluzione: tutte le ***istruzioni di I/O sono privilegiate***

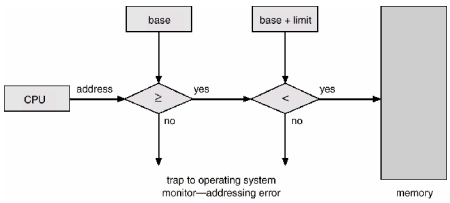
3. Memory Protection

- Proteggere interrupt vector e codice delle interrupt routines
- Due registri che determinano il range di indirizzi leciti di un processo:
  - **base register** – il piu' piccolo indirizzo fisico di memoria lecito del processo
  - **limit register** – taglia del range di indirizzi del processo
- La memoria al di fuori del range deve essere protetta

Spazio di indirizzi determinati da *base* e *limit*



Congegno di protezione (hardware) basato su *base* e *limit*



- In modo supervisor si ha accesso sia allo spazio user che quello supervisor
- Le ***istruzioni*** che manipolano ***i registri base e limit*** sono ***privilegiate***

4. CPU Protection

Regola : Evitare che un ***processo*** possa detenere per un ***tempo illimitato*** il controllo della ***CPU***

Soluzione : un timer che interrompe l'esecuzione dopo un certo tempo e rida' il controllo a OS (ex. quantum in time sharing)

Le istruzioni di ***manipolazione del timer*** sono ***privilegiate***

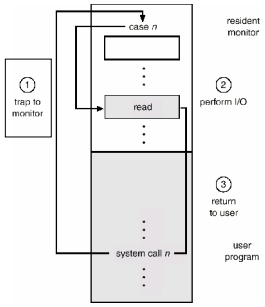
Meccanismo per l'esecuzione di istruzioni privilegiate

- OS deve sempre mantenere il controllo
- Come fa un processo utente ad accedere alle risorse se le istruzioni relative sono tutte privilegiate?

⇩

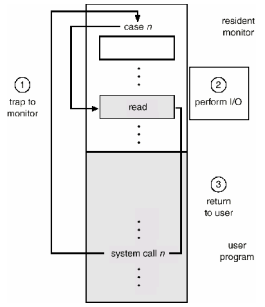
System call – il metodo usato da un processo per richiedere un'azione di OS

Uso di una system call per I/O



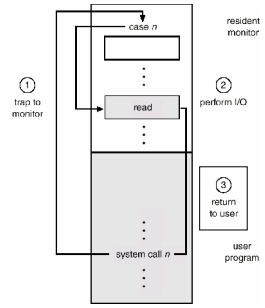
1- Usualmente prende la forma di una ***trap*** ad una specifica locazione dell'interrupt vector

Uso di una system call per I/O



2- Il controllo passa ***attraverso l'interrupt vector*** ad una ***routine di servizio di OS***, ed il ***mode*** viene settato ***a supervisor***

Uso di una system call per I/O



3- OS ***verifica*** che i ***parametri*** (caricati in qualche registro) sono ***corretti e leciti***, esegue la richiesta, e ***ritorna*** il controllo ***all'istruzione che seguiva la system call***

Esempio di moderni compiti di OS: una Wide Area Network