AVVISI IMPORTANTI

- Propedeuticita' per il corso di Sistemi Operativi
  - Matricole 566:
    - Programmazione mod.A e mod.B
    - Architetture degli elaboratori mod.A
  - Matricole N86
    - Programmazione I
    - Architetture degli elaboratori I

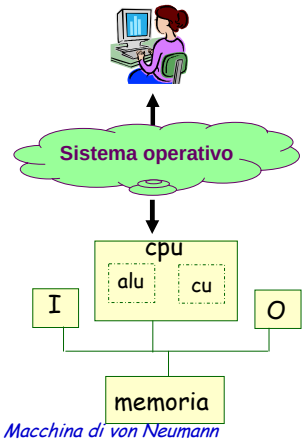
In nessun caso vengono "congelati" esami

Lezione 3:

Come interagiscono hardware e sistemi operativi

- hardware che deve essere gestito dal s.o.
- supporto hardware ai s.o.
- come il s.o. operativo protegge se stesso e i programmi
- l' I/O

Compiti del S.O.



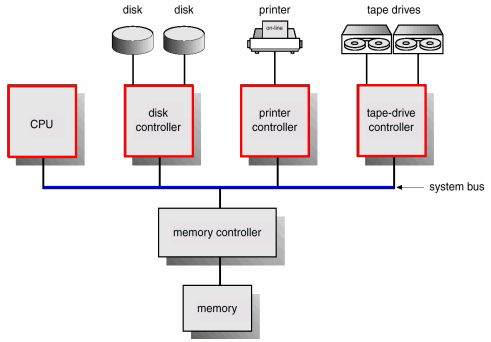
Macchina di von Neumann

Un S.O. ha il compito di rendere semplice (all'utente), l'utilizzo del calcolatore

Cosa deve fare un S.O. ?

Interazioni con:
- CPU
- Memoria
- Dispositivi di I/O

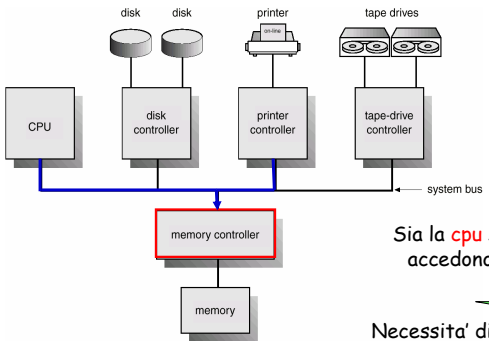
Architettura di un Sistema di Calcolo



Controller = dispositivo hw per la gestione delle periferiche

- la cpu e i controller operano in modo concorrente
- la cpu e i controller comunicano attraverso un bus

Architettura di un Sistema di Calcolo



Sia la cpu sia i controller accedono alla memoria

Necessita' di sincronizzazione degli accessi

Controller della memoria

la CPU

Compiti della CPU:
- recuperare una istruzione dalla memoria
- Decodificarla per determinare tipo e operandi
- eseguirla

Tutte le CPU hanno un insieme di registri:
- per contenere dati in transito da e per la memoria (accumulatori)
- program counter (indirizzo della prossima istruzione)
- stack pointer (indirizzo dello stack)
- program status word (insieme di bit di controllo)
- per delimitare lo spazio di indirizzamento del programma in esecuzione

La memoria centrale

La memoria centrale e' il supporto su cui sono conservati dati e istruzioni

E' vista dalla CPU come una sequenza lineare di locazioni con un indirizzo

Le uniche operazioni permesse sui dati residenti in memoria sono load e store

E' il solo dispositivo di memorizzazione di grandi dimensioni direttamente accessibile dalla CPU

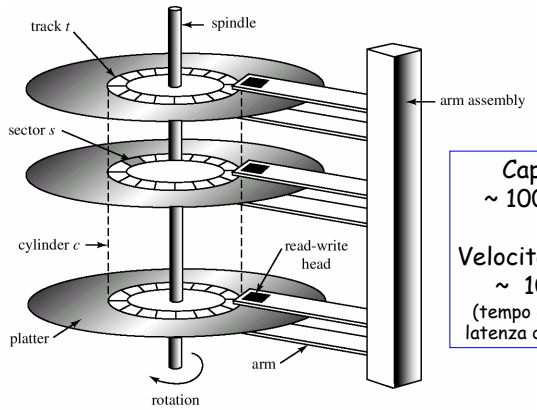
Memorie di massa

La memoria centrale e' un dispositivo volatile

Ha una capicita' dell'ordine dei 10^9 byte (Gbyte) e non e' sufficiente a contenere i modo permanente tutti i dati e i programmi di un sistema operativo

necessita' di dispositivi piu' capienti e non volatili
(memoria di massa / dischi magnetici)

Dischi magnetici



Capacita'
~ 100 Gbyte

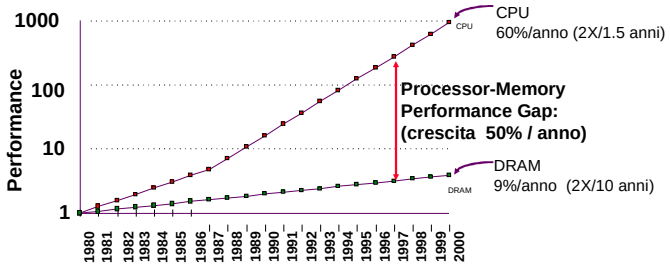
Velocita' accesso
~ 10^{-6}sec.
(tempo di ricerca + latenza di rotazione)

Gap tra memoria e CPU

Tempo per una operazione aritmetica
1 ciclo di clock (~ 10^{-9} sec)

Tempo di accesso alla memoria
4-5 cicli di clock (~ 10^{-8} sec)

Rallentamento della CPU



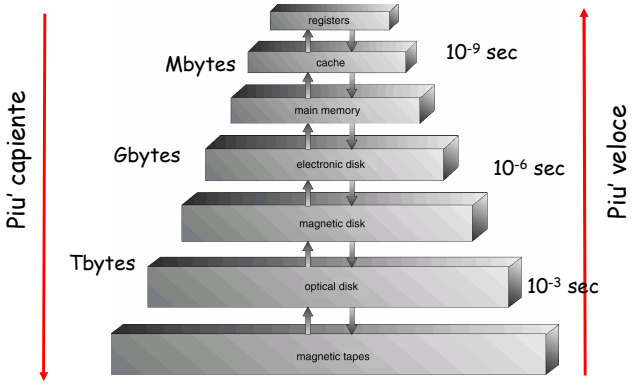
Registri e cache

Soluzione:
uso di dispositivi di memorizzazione 'on chip' capaci di supportare la velocita' operativa della CPU
(cache e registri)

Al momento del loro uso, blocchi di dati sono copiati dalla memoria nella cache

Caratteristiche:
- limitata capacita' (~10 Mbyte)
- alta velocita' di accesso (~ 10^{-9} sec)

La gerarchia delle Memorie



Caching

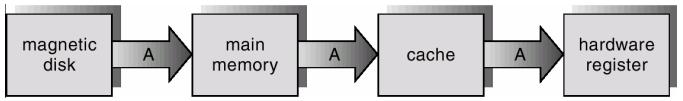
Caching – utilizzo di una memoria a piu' alta velocità per mantenere informazioni cui si accede più spesso.

- Richiede un politica di gestione della cache.
- Provoca una replicazione dei dati, quindi necessita di una politica di gestione che garantisca la *consistenza* dei dati (di solito gestita dall'hardware).

E' un concetto che puo' essere applicato a piu' livelli

- La memoria centrale puo' essere vista come una *cache* per i dischi magnetici
- I dischi possono essere visti come una cache per i supporti di backup

Migrazione di un dato dal disco ai registri



Problemi :

- dimensioni della cache (piu' grande = piu' costoso)
- criteri di aggiornamento ('prevedere' quali dati saranno utilizzati dalla cpu)

Gestione efficiente della cache ➔ 80% dei dati deve trovarsi nella cache quando servono

Protezione delle risorse

L'esistenza di risorse condivise richiede che il sistema operativo garantisca che un programma scorretto non possa effettuare operazioni non consentite.

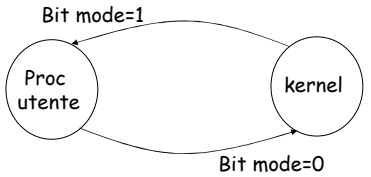
Le istruzioni possono essere eseguite in due modalità:

1. *User mode* – (un utente qualsiasi puo' eseguire un insieme ristretto di istruzioni).
2. kernel *mode* (anche *monitor mode*, *system mode* o *superuser mode*) – (il sistema operativo puo' eseguire tutte le istruzioni).

(dual mode operation)

Supporto hardware per la dual mode operation

- La CPU deve essere dotata di un *Mode bit* che indica lo stato corrente: system (0) or user (1).
- Quando giunge una interruzione o avviene un errore il sistema passa in modalità "sistema" e viene attivata la procedura di servizio.



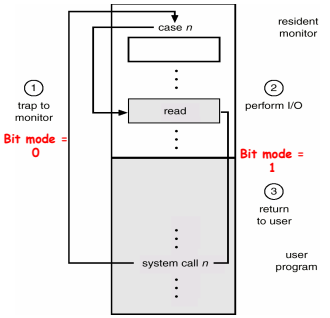
Esistono *istruzioni privilegiate* che possono essere eseguite solo in modalità superutente.

esempi

Sono istruzioni di tipo privilegiato tutte quelle che sono relative a moduli del kernel. Ad es.

- Chiamate di sistema
- Istruzioni di I/O

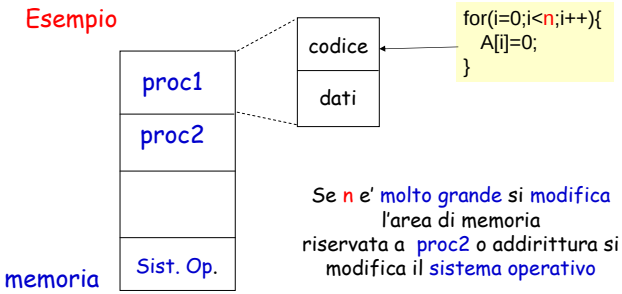
1. manda una interruzione alla CPU e pone il bit mode =0 (modalita' sistema)
2. esegue la procedura di servizio
3. ripone il bit mode=1 (modalita' utente) e restituisce il controllo del sistema al programma interrotto



problema

La condivisione delle risorse (memoria, cpu, I/O) da parte dei processi di un sistema operativo, comporta che un errore in un programma puo' alterare il funzionamento di tutto il sistema

Esempio

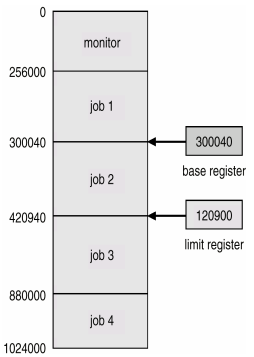



```
for(i=0;i<n;i++){
   A[i]=0;
}
```

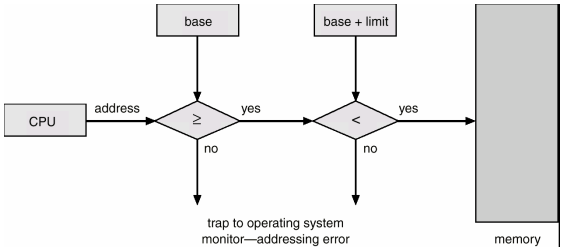
Se n e' molto grande si modifica l'area di memoria riservata a proc2 o addirittura si modifica il sistema operativo

Supporto hardware per la protezione della memoria

- E' necessario garantire protezione della memoria , come minimo per le informazioni che sitrovano nel vettore delle interruzioni e nelle ISR.
- La CPU deve possedere due registri per stabilire le locazioni di memoria cui ogni programma ha diritto ad accedere.:
  - **Base register** – memorizza il più piccolo indirizzo di memoria cui l'accesso è lecito.
  - **Limit register** – Contiene la dimensione massima di memoria ad accesso consentito
- La memoria al di là dell'intervallo indicato è *protetta*.



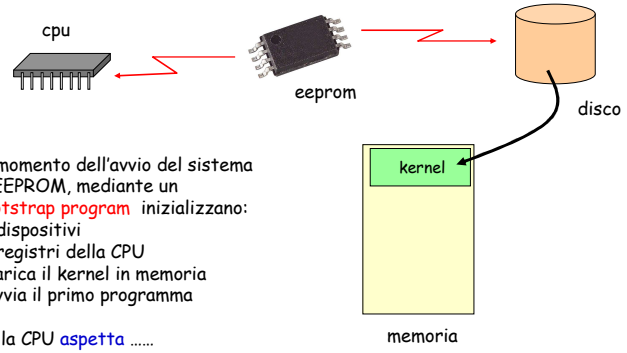
Protezione della memoria



- Quando una istruzione viene eseguita in modalià sistema ha accesso completo a TUTTA la memoria.
- Le istruzioni per caricare i valori dei registri *base* e *limit* sono di tipo privilegiato.

Le EEPROM

Electrically Erasable and Programmable Read Only Memory



Al momento dell'avvio del sistema le EEPROM, mediante un bootstrap program inizializzano:
- i dispositivi
- i registri della CPU
- carica il kernel in memoria
- avvia il primo programma

Poi la CPU aspetta .....

Gli eventi

Gli eventi (oppure segnali, oppure interruzioni) sono il principale meccanismo con cui si sincronizzano le azioni di un moderno sistema operativo

La ricezione di un evento avvisa la CPU che deve fare qualcosa

Se non ci sono processi, dispositivi da servire o utenti con cui interagire il sistema operativo rimane inattivo nell'attesa di un evento (i S.O. sono *events driven*)

eventi: dipendono dall'architettura

In generale due tipi di eventi

esempio: processori Intel

evento segnalato da una componente hardware (disp. di I/O, memoria, timer)

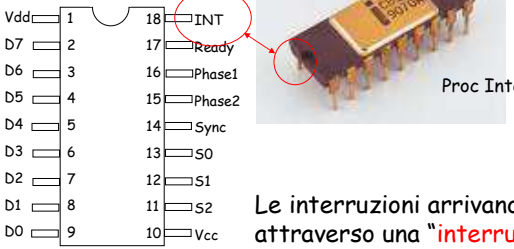
segnali di interruzione (interrupt)

evento segnalato da un programma in esecuzione

Segnali di eccezione (trap → es: div per 0
fault → es: overflow
abort → es: errore in un evento)

In generale si parla sempre di interruzioni

La interrupt line



Proc Intel 8008 (1974)

Le interruzioni arrivano alla CPU attraverso una "interrupt line"

Alla fine di ogni ciclo macchina la CPU controlla la presenza di un segnale sulla interrupt line

Come viene gestito una interruzione

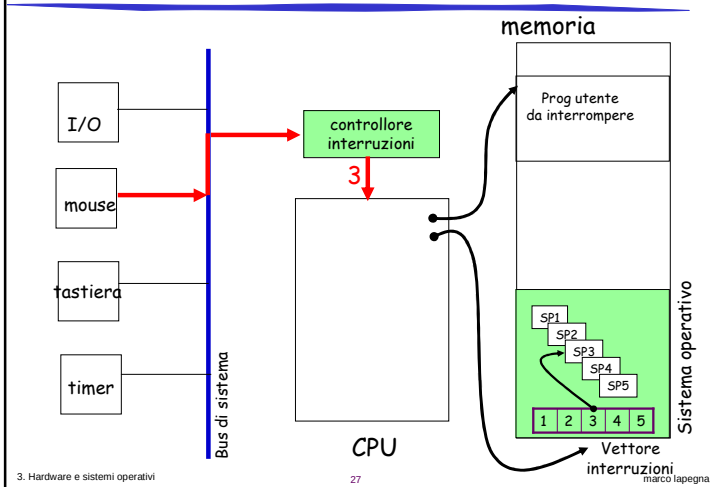
- Un dispositivo attiva una comunicazione elettrica con un controllore delle interruzioni della CPU
- Il controllore delle interruzioni informa la CPU codificando il tipo di interruzione
- La CPU interrompe l'esecuzione dell'istruzione corrente e salva lo stato del programma in esecuzione (dati e registri della CPU).
- trasferisce il controllo ad una *interrupt service routine* (ISR) il cui codice si trova in una prefissata zona della memoria
- Le ISR (Interrupt Service Routine) eseguono il codice specifico per gestire il segnale appena giunto (ad es. trasferisce il dato dal buffer locale del controller alla memoria)
- Al termine dell'esecuzione della ISR, il S.O. ripristina lo stato del programma e riprende l'esecuzione

La gestione delle interruzioni

L'efficienza di un Sistema Operativo dipende dall'efficienca con cui sono gestite le interruzioni

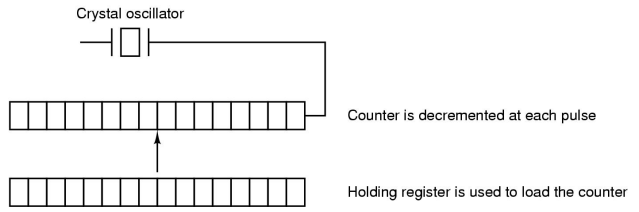
Per rendere efficiente la gestione delle interruzioni si usa di solito una sola ISR che gestisce una tabella di puntatori contenente gli indirizzi delle varie procedure di servizio (vettore delle interruzioni)

Gestione delle interruzioni



Un esempio: il clock (o timer)

- Ogni calcolatore ha un orologio interno chiamato clock o timer
- 3 componenti
  - Cristallo al quarzo che vibra ad una fissata frequenza
  - Contatore che viene decrementato da ogni impulso del cristallo
  - Registro di inizializzazione del contatore



clock

- Il contatore viene inizializzato dal registro di caricamento
- Quando il contatore si azzera viene emessa una interruzione

La frequenza delle interruzioni viene determinata dal software
(dal valore del registro di caricamento)

Esempio: con un cristallo da 500 MHz e registro di 32 bit. ➔ il contatore viene decrementato ogni 2 ns

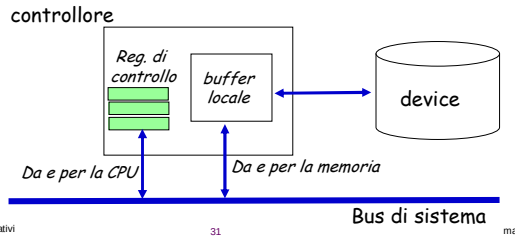
Interruzioni ogni 2 ns (registro di caricamento =1
Interruzioni ogni 8,6 sec (registro di caricamento = 2^{32})

Principali compiti del sw dei clock

- Mantenere ora e data in appositi registri di 32 o 64 bit anche quando si spegne il calcolatore
  - Contando i tic dal 1 gennaio 1970 (UNIX) o dal 1 gennaio 1980 (Windows)
- Evitare il monopolio della CPU da parte di un processo
- Tenere la contabilita' dell'uso delle risorse
  - Altri registri
- Ritardare segnali e interruzioni nel caso di processi critici

Controllori dei dispositivi di I/O

- Ogni dispositivo e' governato da un *controller*
- Ogni *controller* ha
  - un suo buffer locale.
  - Registri di controllo
- Le operazioni di I/O avvengono dal dispositivo verso i buffer locali e viceversa.
- La CPU sposta i dati tra la memoria e il buffer



Esempio

- 3 registri di controllo
  - status (read only) indica lo stato del dispositivo
    - 1 dispositivo occupato
    - 0 dispositivo pronto ad eseguire un comando
  - command (write) indica il comando da eseguire
    - Read / write
  - control (write) indica che un comando e' pronto nel registro command
    - 1 comando da eseguire presente nel registro command
    - 0 nessun comando da eseguire
- 2 buffer
  - Di input
  - Di output
- L'insieme delle regole per il coordinamento tra CPU e controller e' chiamato negoziazione (handshaking)

Esempio di negoziazione per l'output

1. La CPU interroga il registro status fino a che status==0
2. La CPU definisce il registro command = write e trasferisce i dati dalla memoria al buffer di output
3. La CPU pone il registro control=1
4. Il controller pone status=1
5. Il controller esamina il registro command e trasferisce i dati dal buffer al dispositivo
6. Il controller pone command=0 e status=0

ATTESA ATTIVA nel passo 1

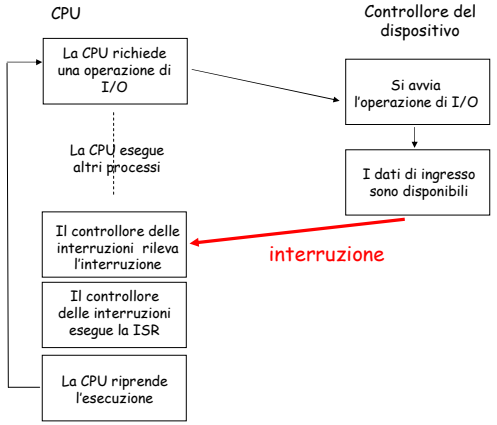
Attesa attiva (polling)

- Interrogare occasionalmente un dispositivo è in sé un'operazione poco costosa
  - Leggi il registro status
  - Test sul contenuto
  - Salto ad un altro punto del codice se status=0
- tale tecnica diviene però inefficiente se le ripetute interrogazioni trovano raramente un dispositivo pronto per il servizio mentre altre utili elaborazioni attendono la CPU.

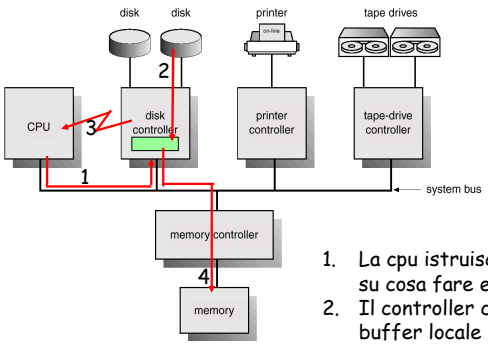
L'alternativa: le interruzioni

- I controllori dei dispositivi, gli errori e le chiamate di sistema generano segnali d'interruzione per comunicare alla CPU un evento asincrono
- Le interruzioni vengono generate dai dispositivi e recapitate alla CPU attraverso la linea delle richieste delle interruzioni
- Dipendono dall'architettura e possono avere nomi differenti
- Esempio: architettura IA32 (Intel Pentium)
  - Interrupt (se generati da dispositivi hardware)
  - Exceptions (se generati da errori o da programmi in esecuzione)

Ciclo di I/O basato sulle interruzioni

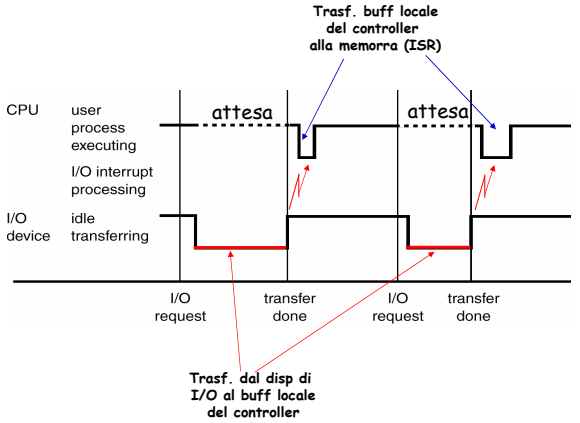


Esempio: lettura di un dato



1. La cpu istruisce il controller su cosa fare e attende
2. Il controller carica i dati nel buffer locale
3. Il controller manda un interrupt di fine I/O alla cpu
4. La cpu trasferisce il dato dal buffer locale alla memoria

Time Line per l'I/O di un singolo processo



Accesso diretto alla memoria

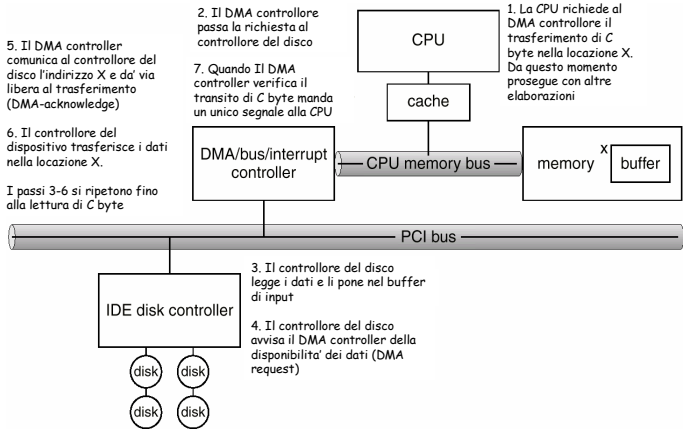
La dimensione di buffer dei controllori e' di pochi byte.

Quando devono essere trasferiti molti dati vengono generate molte interruzioni alla CPU

La gestione dell'I/O da e per la memoria e' di solito delegato ad una unita' separata dalla CPU chiamata
Controllore dell'Accesso Diretto alla Memoria (DMA controller)

Il controllore DMA agisce direttamente sul bus della memoria, esegue il trasferimento senza l'aiuto della CPU e genera una sola interruzione

Passi di un trasferimento DMA



Problema

Quando la CPU effettua una operazione di I/O, deve attendere il completamento del trasferimento dei dati dal dispositivo alla CPU (I/O sincrono)

- vantaggi:
  - una sola richiesta di I/O pendente alla volta
  - la CPU riconosce subito da quale dispositivo arriva il segnale di interruzione
- svantaggi:
  - la CPU rimane inattiva per tutta la durata dell'I/O
  - no a operazioni I/O in parallelo
  - no a sovrapposizione di I/O e calcolo

soluzione

Dopo l'avvio dell'I/O il controllo torna subito alla CPU, e un nuovo programma viene mandato in esecuzione (I/O asincrono)

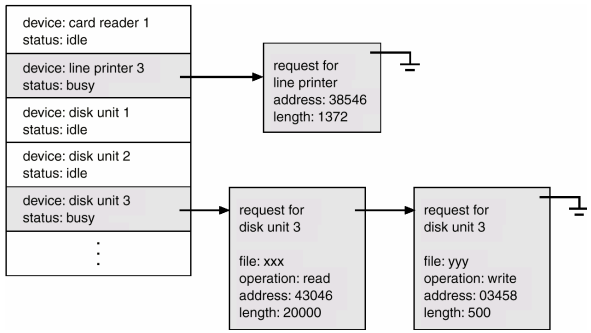
Le interruzioni sono alla base della multiprogrammazione

se c'e' un solo processo in esecuzione, tale processo deve comunque attendere la fine dell'operazione di I/O

Possibilita' di numerose richieste di I/O pendenti

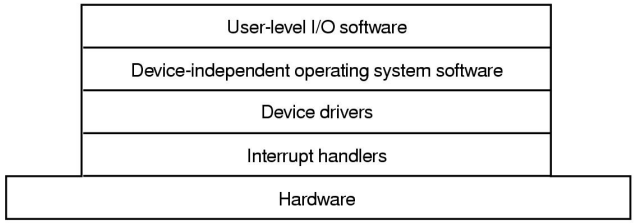
Device-Status Table

La presenza di piu' operazioni di I/O pendenti impone la presenza di una tabella di stato dei dispositivi



I livelli del software di I/O

Benche' l'organizzazione del software che gestisce l'I/O dipende dal sistema operativo e' in genere possibile individuare alcune componenti fondamentali



Gestore delle interruzioni

Problemi:

- Necessita' di differire le interruzioni nel caso di elaborazioni critiche
- Necessita' di un meccanismo efficiente e rapido per determinare quale dispositivo ha generato l'interruzione
- Necessita' di priorita' tra le varie interruzioni

Soluzione:

- La gestione e'affidata ad un dispositivo separato dalla CPU chiamato controllore delle interruzioni
- Per rendere efficiente la gestione delle interruzioni si usa di solito una sola Interrupt Service Routine (ISR) che gestisce una tabella di puntatori contenente gli indirizzi delle varie procedure di servizio (vettore delle interruzioni)

Compiti del gestore delle interruzioni

- Un dispositivo attiva una comunicazione elettrica con il controllore delle interruzioni della CPU
- La CPU interrompe l'esecuzione dell'istruzione corrente
- salva lo stato del programma in esecuzione (dati e registri della CPU).
- trasferisce il controllo ad una *interrupt service routine* (ISR) il cui codice si trova in una prefissata zona della memoria
- Le ISR (Interrupt Service Routine) eseguono il codice specifico per gestire il segnale appena giunto (ad es. trasferisce il dato dal buffer locale del controller alla memoria)
- Al termine dell'esecuzione della ISR, il S.O. ripristina lo stato del programma e riprende l'esecuzione

Vettore delle interruzioni della CPU intel Pentium

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19Ð31	(Intel reserved, do not use)
32Ð255	maskable interrupts

errori

dispositivi

Driver dei dispositivi

- Problema: differenze tra i vari controllori di dispositivo:
  - Es: i controller dei dischi possono essere molto diversi tra loro (# tracce, #settori,…)
- Il software che governa i dispositivi e rende uniforme l'accesso e' detto driver del dispositivo
- I driver hanno il compito di isolare il sottosistema di I/O del nucleo dai controllori dei dispositivi
  - Es: se si sostituisce un disco e' necessario sostituire solo il relativo driver senza modificare il resto del sistema operativo
- Poiche' ogni sistema operativo ha le sue convenzioni, ogni dispositivo necessita di un driver differente per ogni sistema operativo. Tale driver e' in genere scritto dal produttore del dispositivo

Driver dei dispositivi

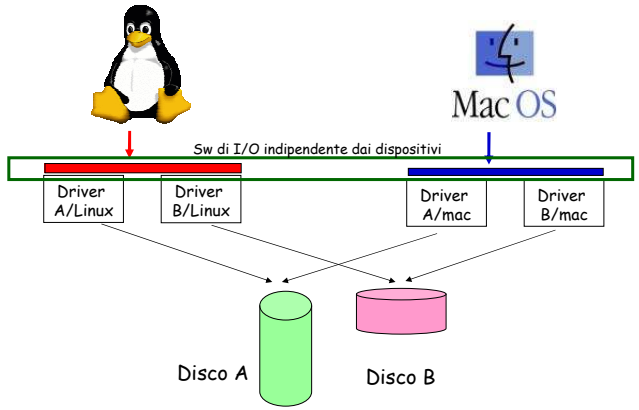
- 2 categorie di driver dei dispositivi
  - Dispositivi a blocco (ad es. dischi, fotocamere, cdrom)
  - Dispositivi a carattere (ad es. tastiera, terminali, stampanti, interfacce di rete)
- Funzioni principali dei driver
  - Accettare richieste di I/O dagli strati superiori del s.o.
  - Effettuare controlli sui dati
  - Gestire le differenti velocita' dei dispositivi (protocollo prod. cons.)
  - Garantire l'alimentazione, rilevare la presenza del dispositivo (es. hot swap dei dispositivi USB)

Sw di I/O indipendente dai dispositivi

- I driver dei dispositivi esportano le funzionalita' dei dispositivi di I/O verso il sistema operativo
- I driver sono strettamente legati al dispositivo, ma ogni s.o. utilizza un disco sempre allo stesso modo (ad es. differenti metodi di memorizzazione di un file)
- Problema: portabilita' delle applicazioni su calcolatori con dispositivi di I/O differenti:
  - Es: l'accesso ad un file dovrebbe essere sempre lo stesso indipendentemente dal tipo di disco su cui e' memorizzato

Necessita' di un livello software che renda standard l'accesso ai dispositivi

esempio



Compiti del sw di I/O indipendente dai dispositivi

- Interfacciamento uniforme
  - (rendere i dispositivi quanto piu' simili possibile)
- Bufferizzazione
  - (contribuire a gestire le differenti velocita' dei dispositivi)
- Gestire e riportare errori
  - (es. lettura da unita' di output e viceversa)
- Allocare e rilasciare risorse
  - (Garantire la mutua esclusione nell'accesso di alcuni dispositivi)

Sw di I/O a livello utente

- Librerie utente per l'I/O
  - (es. le chiamate di sistema scanf e printf)
- software per la gestione dei file speciali associati ai dispositivi
  - (es. spooling)

spooling

In molti s.o. (es. UNIX) i dispositivi sono associati a file speciali

Esempio: la directory /dev contiene tali file speciali

Problema : se un processo accede al file speciale associato ad una stampante e non lo rilascia per un lungo periodo, la stampante non puo' essere utilizzata da altri processi

Soluzione: si copia il file da stampare in una directory di spool che viene gestita da un apposito processo (demone) che e' l'unico ad avere diritti sul file speciale associato alla stampante

Riepilogo di una istruzione di I/O