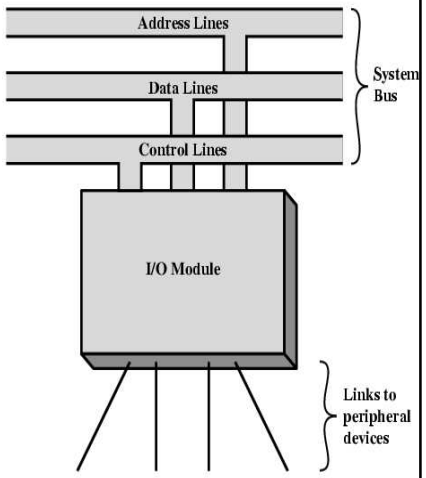
Input/Output
(*Cap. 7, Stallings*)

- Grande varietà di periferiche
  - gestiscono quantità di dati differenti
  - a velocità diverse
  - in formati diversi
- Tutti più lenti della CPU e della RAM
- Necessità di avere moduli di I/O

Moduli di Input/Output

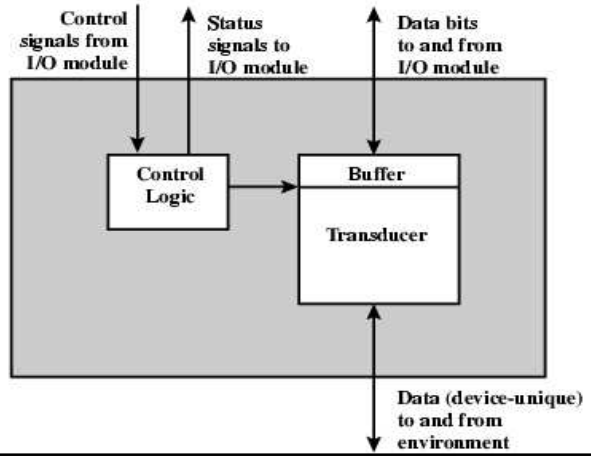
- Si interfacciano con
  - CPU e Memoria
  - Una o più periferiche



Dispositivi Esterni

- Comprensibili dall'uomo
  - video, stampante, tastiera
- Comprensibili dalla macchina
  - Monitoraggio e controllo
- Comunicazione
  - Modem
  - Rete [Network Interface Card (NIC)]

Schema di dispositivo esterno



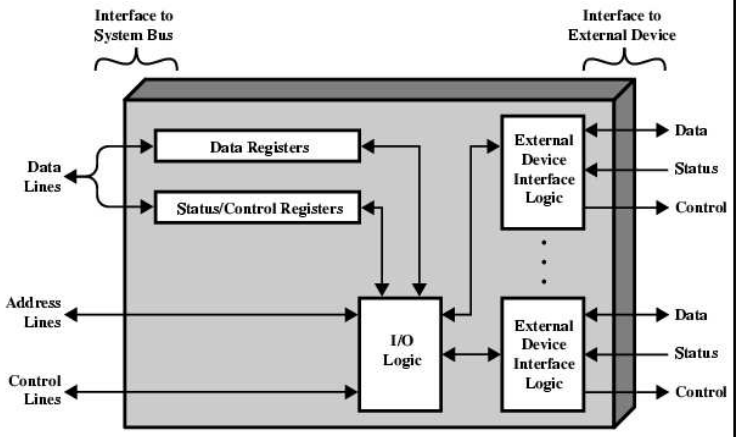
Funzioni del modulo I/O

- Controllo & Temporizzazione
- Comunicazione con CPU
- Comunicazione con i dispositivi
- *Buffering* dei dati
- Rilevazione degli errori

Passi di I/O (versione semplificata)

- CPU interroga il modulo I/O sullo stato del dispositivo connesso
- Il modulo I/O restituisce lo stato del dispositivo
- Se dispositivo pronto a trasmettere, CPU richiede il trasferimento dei dati, tramite comando a modulo I/O
- Il modulo I/O ottiene una unità di dati dal dispositivo esterno
- Il modulo I/O trasferisce i dati alla CPU

Diagramma modulo I/O



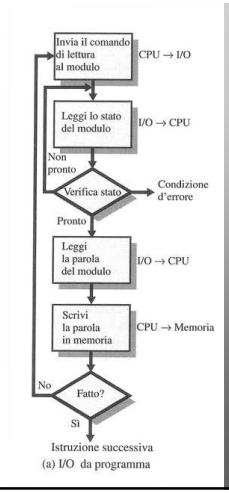
Caratteristiche modulo I/O

- Nascondere o rivelare le proprietà del dispositivo alla CPU
- Supportare dispositivi singoli o multipli
- Controllare le funzioni del dispositivo o lasciare il controllo alla CPU
- Caratteristiche Sistema Operativo
  - Ad esempio, Unix tratta tutto quello che può come se fosse un file

Tecniche di gestione Input/Output

- I/O da programma
  Programmed I/O
- I/O guidato da interrupt
  Interrupt Driven I/O
- Accesso Diretto alla Memoria
  Direct Memory Access (DMA)

Tre tecniche per l'input di un blocco di dati



(a) I/O da programma

I/O da programma

- CPU ha il controllo diretto sull' I/O
  - Controllo stato dispositivo
  - Comandi lettura/scrittura
  - Trasferimento dati
- CPU aspetta che il modulo I/O completi l'operazione
- Spreca tempo di CPU

I/O da programma- dettaglio

- CPU richiede operazione I/O
- Modulo I/O esegue operazione
- Modulo I/O setta bit di stato
- CPU controlla bit di stato periodicamente
- Modulo I/O non informa direttamente CPU
- Modulo I/O non interrompe CPU
- CPU può attendere o fare altro e controllare più tardi

Comandi I/O

- CPU invia indirizzo
  - che identifica modulo (& dispositivo se >1 per modulo)
- CPU invia commando
  - di controllo – dire al modulo come fare
    - ad esempio, dare velocità al disco
  - di test – controlla lo stato
    - ad esempio, alimentazione? errore?
  - di lettura/scrittura
    - il modulo trasferisce i dati tramite buffer dal/verso il dispositivo

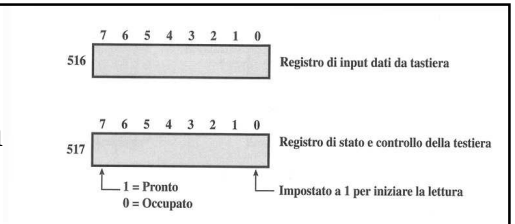
Indirizzamento dispositivi I/O

- Nell' I/O da programma il trasferimento dati è molto simile all'accesso alla memoria (dal punto di vista della CPU)
- Ad ogni dispositivo viene assegnato un identificatore unico
- I comandi di CPU riferiscono tale identificatore (indirizzo)

I/O Mapping

- I/O memory-mapped
  - Dispositivi e memoria condividono lo stesso spazio di indirizzamento
  - I/O sembra proprio come lettura/scrittura di memoria
  - Nessun comando speciale per I/O
    - Ampia varietà di comandi di accesso alla memoria disponibili
- I/O separato (isolated)
  - Spazi di indirizzamento separati
  - Necessita di linee di selezione fra I/O e memoria
  - Comandi speciali per I/O
    - Insieme limitato

Confronto fra I/O memory mapped e separato



INDIRIZZO	ISTRUZIONE	OPERANDO	COMMENTO
200	Load AC	"1"	Carica nell'accumulatore
	Store AC	517	Inizia la lettura della tastiera
202	Load AC	517	Legge il byte di stato
	Branch if Sign = 0	202	Sta in loop fino a quando è pronto
	Load AC	516	Carica un byte di dati

(a) I/O memory mapped

INDIRIZZO	ISTRUZIONE	OPERANDO	COMMENTO
200	Load I/O	5	Inizia la lettura della tastiera
201	Test I/O	5	Controlla il completamento
	Branch Not Ready	201	Sta in loop fino al completamento
	In	5	Carica un byte di dati

(b) I/O isolato

Confronto fra I/O memory mapped e separato

- La tecnica *memory-mapped I/O* ha diversi vantaggi
  - Non necessita di istruzioni speciali
    - Le istruzioni che accedono memoria 'normale' accedono anche le aree di I/O
    - Il software di controllo di dispositivo può essere scritto interamente in linguaggi ad alto livello
  - Consente una più agevole protezione
    - è sufficiente nascondere le aree di I/O allo spazio di indirizzamento di utente (*privilegi*)
    - Con la tecnica della memoria segmentata, *più* aree di I/O possono mappare sul *medesimo* spazio di indirizzamento fisico

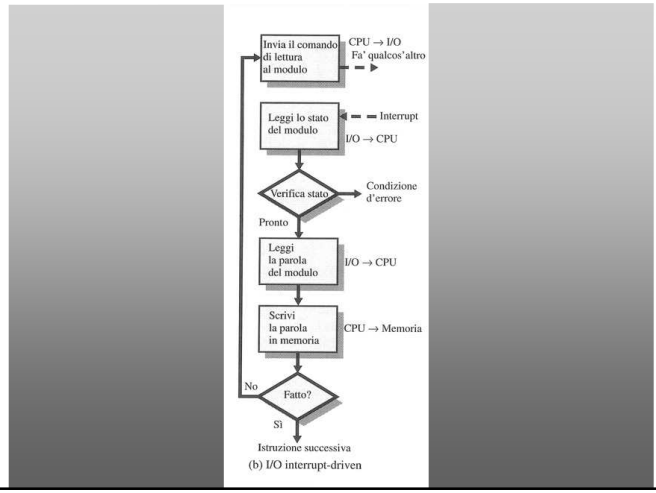
Confronto fra I/O memory mapped e separato

- La tecnica *memory-mapped I/O* presenta anche alcuni svantaggi
  - Non si presta all'uso di cache
    - Il dato rilevante è *sempre e solo* nella memoria del dispositivo
    - Occorre disabilitare selettivamente la cache
  - Non è compatibile con architetture a 'bus multipli'
    - I dispositivi di I/O non possono rispondere ad indirizzi emessi su bus non connessi
    - Occorre filtrare gli indirizzi emessi dalla CPU ed instradarli sul bus appropriato
      - Filtraggio a sorgente piuttosto che a destinazione

I/O Interrupt Driven

- Evita l'attesa da parte della CPU
- Nessun controllo ripetuto dello stato del dispositivo da parte della CPU
- Il modulo di I/O interrompe la CPU quando è pronto

Tre tecniche per l'input di un blocco di dati

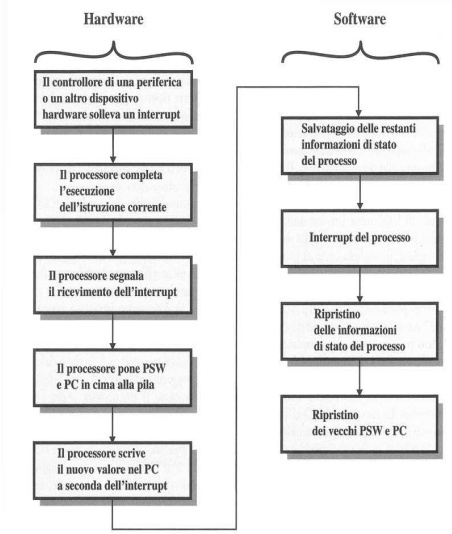


(b) I/O interrupt-driven

I/O Interrupt Driven Operazioni Base

- CPU rilascia comando di lettura
- Modulo I/O ottiene i dati dalla periferica mentre la CPU svolge altro lavoro
- Modulo I/O interrompe la CPU
- CPU richiede i dati al modulo I/O
- Modulo I/O trasferisce i dati

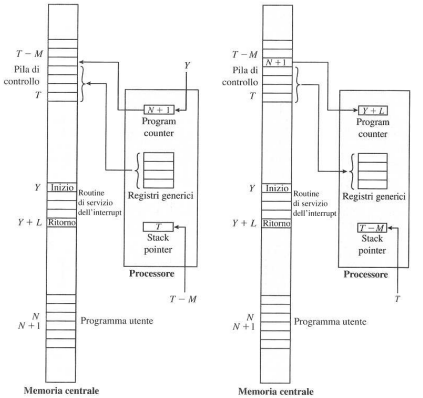
Semplice elaborazione delle interruzioni



Punto di Vista della CPU

- Rilascia comando di lettura
- Esegue altro lavoro
- Controlla se c'è interruzione alla fine di ogni ciclo di istruzione (ciclo fetch/execute con trattamento delle interruzioni)
- Se interruzione presente :-
  - Salva contesto (PC e registri)
  - Interruzione del processo corrente e elaborazione interrupt
    - Lettura dati da modulo I/O e scrittura in memoria

Cambiamento in Memoria e Registri per un Interrupt



Progettazione

- Come identificare il modulo che ha rilasciato l' interrupt?
- Come trattare interrupt multipli?
  - cioè l'interruzione di una procedure di gestione di una interruzione

Identificazione del modulo che genera l'interruzione (1)

- Linee differenti per ogni modulo
  - Non molte linee di bus per interrupt
  - Limita il numero di dispositivi
- Interrogazione software (software poll)
  - CPU interroga ogni modulo a turno
  - Lento

Identificazione del modulo che genera l'interruzione (2)

- Daisy Chain (anche detto hardware poll)
  - Interrupt di riconoscimento inviato su linea che connette tutti i moduli
  - Modulo responsabile dell'interruzione pone una parola (vettore) sul bus dati
  - CPU usa il vettore per identificare la giusta procedura di gestione dell'interruzione
- Arbitraggio del Bus
  - Modulo deve ottenere il possesso del bus prima di rilasciare una interruzione
  - p.e. PCI & SCSI

Interruzioni multiple

- Ogni linea di interruzione ha una priorità
- Linee a priorità più alta possono interrompere quelle a priorità più bassa
- Se c'è l'arbitraggio del bus, solo il modulo che possiede il controllo del bus può interrompere

Interruzioni e Pipeline

- 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

Interruzioni e Pipeline

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

Interruzioni e Pipeline

- 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