ARCHITETTURA DEGLI ELABORATORI

APRILE 2020

PROF. TRAMONTANA
WWW.DMI.UNICT.IT/~TRAMONTA

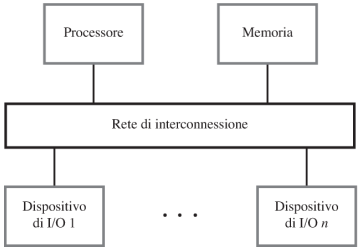
CAPITOLO 3
OPERAZIONI DI INGRESSO E USCITA

Obiettivi del capitolo

- Trasferimento di dati fra processore e dispositivi di I/O
- Trasferimenti di I/O dal punto di vista del programmatore
- I/O controllato da programma mediante scansione (polling)
- Uso delle interruzioni nei trasferimenti di I/O

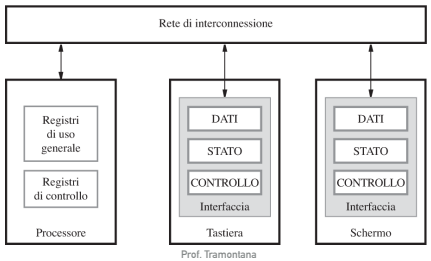
Accesso a dispositivi di I/O (S. 3.1)

- Il calcolatore scambia dati con dispositivi di ingresso e uscita (input/output, I/O), detti dispositivi di periferia o periferiche. Es. tastiera, video, e una varietà di sensori, e attuatori. I componenti di un calcolatore comunicano attraverso una **rete di interconnessione**
- Il processore accede a locazioni di memoria all'interno dello **spazio degli indirizzi**. `Load` e `Store` accedono alla memoria usando i modi di indirizzamento
- La tecnica di accedere tramite indirizzi a varie locazioni di memoria è estesa per accedere a dispositivi di I/O. Così ogni dispositivo appare al processore come un insieme di locazioni indirizzabili
- Alcuni indirizzi sono assegnati a locazioni di I/O anziché alla memoria. Questa disposizione si chiama **unificazione degli spazi di indirizzamento di memoria e di I/O** (**memory mapped I/O**)
- Sia `DATO_ING` l'indirizzo del registro buffer di ingresso della tastiera, allora `Load R2, DATO_ING` legge il carattere digitato sulla tastiera e lo scrive in `R2`, ovvero effettua la lettura dalla tastiera



Interfacce dei dispositivi di I/O

- Un dispositivo di I/O è collegato alla rete di interconnessione mediante un circuito detto *interfaccia del dispositivo*
- *L'interfaccia* include alcuni **registri** accessibili dal processore. Un registro può servire da buffer per i **dati**, un altro può tenere lo **stato** corrente del dispositivo, un altro per le informazioni che **controllano il modo di operare** del dispositivo
- L'accesso a questi registri avviene come se fossero locazioni di memoria

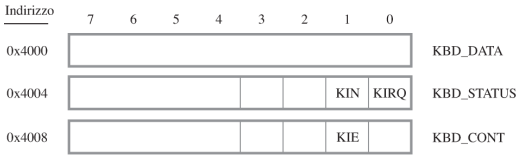


I/O controllati da programma

- Consideriamo l'acquisizione di dati dalla tastiera o la visualizzazione di dati sullo schermo. Entrambe le periferiche sono molto più lente del processore
- Si necessita di un meccanismo di sincronizzazione fra processore e periferiche. Per la tastiera
  - Il processore attende da parte della tastiera la notifica di dato pronto (tasto premuto), che viene inizialmente inserito in un registro buffer della tastiera. La notifica avviene attivando un segnale apposito. Quindi, il processore prende il dato
- Oppure, per il video
  - Il processore manda il carattere al video e attende che quest'ultimo notifichi di averlo ricevuto attivando un segnale apposito

Lettura di dati dalla tastiera

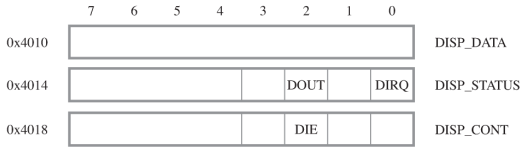
- La pressione di un tasto provoca l'introduzione di un codice a 8 bit nel registro buffer dati a 8 bit della tastiera, detto registro `KBD_DATA`
- La tastiera porta a `1` il bit di stato `KIN` che fa parte del registro di stato
- Il programma esamina periodicamente il bit `KIN` e quando vi trova `1` carica il dato da `KBD_DATA`
- La tastiera recepisce che la lettura è avvenuta e mette il bit `KIN` a `0`
- Alla tastiera è stato assegnato indirizzo di base `0x4000` e ciò rende i registri di I/O accessibili da un programma eseguito dal processore



(a) Interfaccia della tastiera

Scrittura di caratteri sul video

- Il video contiene il registro buffer dati `DISP_DATA` a 8 bit, e il bit di stato `DOUT` del registro di stato `DISP_STATUS`
- `DOUT` a `1` indica che il video è pronto a ricevere un altro carattere. Se è così, il processore scrive su `DISP_DATA` un carattere. L'operazione di scrittura segnala al video di mettere il bit `DOUT` a `0`



(b) Interfaccia dello schermo

Programma di lettura e eco di una linea di caratteri

	Move	R2, #LOC	Inizializza il registro puntatore R2 per puntare all'indirizzo della prima locazione nella memoria principale dove immagazzinare i caratteri
	MoveByte	R3, #CR	Carica in R3 il codice ASCII per il Ritorno Carrello
LEGGI:	LoadByte	R4, KBD_STATUS	Attendi l'immissione di un carattere
	And	R4, R4, #2	Controlla la condizione di stato KIN
	Branch_if_[R4]=0	LEGGI	
	LoadByte	R5, KBD_DATA	Leggi il carattere da KBD_DATA (ciò azzera KIN)
	StoreByte	R5, (R2)	Scrivi il carattere nella memoria principale e incrementa il puntatore alla memoria principale
	Add	R2, R2, #1	
ECO:	LoadByte	R4, DISP_STATUS	Attendi che lo schermo sia pronto
	And	R4, R4, #4	Controlla la condizione di stato DOUT
	Branch_if_[R4]=0	ECO	
	StoreByte	R5, DISP_DATA	Trasferisci il carattere appena letto al registro buffer dello schermo (ciò azzera DOUT)
	Branch_if_[R5]≠[R3]	LEGGI	Controlla se il carattere appena letto sia il Ritorno carrello. Se non lo è, reitera la lettura di caratteri

Lettura di un carattere dalla tastiera

`LoadByte` indica che l'operando è un byte
`And` controlla il bit di stato

Visualizzazione di un carattere sullo schermo

Esempio in stile CISC

- In stile CISC è possibile effettuare alcune operazioni aritmetiche e logiche direttamente su operandi in memoria. `TestBit destinazione, #k` controlla il bit b_k dell'operando destinazione e pone `Z` a `1` se b_k=`0`, altrimenti lo pone a `0`. `Compare destinazione, sorgente` effettua il controllo sottraendo e aggiorna i bit di esito in base al risultato, senza modificare il contenuto né di destinazione, né di sorgente

	Move	R2, #BLOCCO	Inizializza il registro R2 per puntare all'indirizzo della prima locazione nella memoria principale dove immagazzinare i caratteri
LEGGI	TestBit	KBD_STATUS, #1	Monitorando la condizione di stato KIN, attendi l'immissione di un carattere nel registro di I/O KBD_DATA
	Branch=0	LEGGI	
	MoveByte	(R2), KBD_DATA	Scrivi nel byte di memoria puntato da R2 il carattere contenuto nel registro di I/O KBD_DATA (ciò azzera KIN)
ECO	TestBit	DISP_STATUS, #2	Attendi che lo schermo sia pronto monitorandone la condizione di stato DOUT
	Branch=0	ECO	
	MoveByte	DISP_DATA, (R2)	Scrivi il carattere puntato da R2 nel registro di I/O DISP_DATA (ciò azzera DOUT)
	CompareByte	(R2)+, #CR	Verifica se il carattere appena letto da tastiera sia il Ritorno Carrello: se non lo è, reitera la lettura di caratteri; in ogni caso incrementa il registro puntatore R2
	Branch≠0	LEGGI	

Interruzioni (S. 3.2)

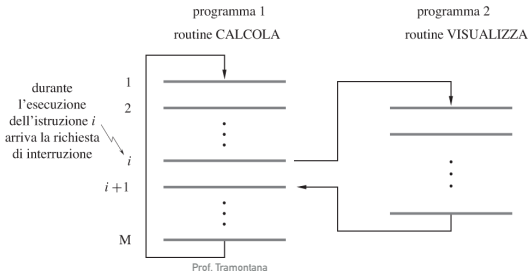
- Con la tecnica di I/O controllato da programma, il tempo del processore viene sprecato nei cicli di attesa (durante il controllo del bit che indica la disponibilità del dato o di ricevere il dato)
- La soluzione a questo problema sta nell'uso delle interruzioni
- La periferica manda un segnale apposito (di interruzione) su una linea apposita del bus di controllo, la linea di richiesta di interruzione `INT_REQ` (interrupt request)
- Grazie al meccanismo di interruzione il processore si dedica ad altri compiti e solo quando la periferica ne fa richiesta il processore legge il dato dal registro dati buffer dell'interfaccia (o scrive il dato). In questo modo non si spreca tempo del processore

Esempio

- Un programma deve fare dei calcoli e visualizzare i risultati ogni 10 secondi. Una soluzione è avere un cronometro che segnala una richiesta di interruzione ogni 10 secondi
- Il programma è costituito da due routine: `CALCOLA` e `VISUALIZZA`
- Quando il processore riceve una richiesta di interruzione, esso sospende l'esecuzione della routine `CALCOLA` ed esegue la routine `VISUALIZZA`. Al termine della routine `VISUALIZZA`, il processore riprende l'esecuzione della routine `CALCOLA` che aveva sospeso
- Il sottoprogramma attivato alla richiesta di interruzione è chiamato **routine di servizio di interruzione** (Interrupt Service Routine, ISR)

Routine di servizio di interruzione

- E' un meccanismo simile a quello di chiamata a sottoprogramma
- Supponendo che la richiesta avvenga durante l'esecuzione dell'istruzione i, l'esecuzione dell'istruzione i viene completata. Quindi in PC viene messo l'indirizzo della routine di servizio
- Alla conclusione dell'esecuzione delle routine di servizio, si ritorna con **return-from-interrupt** che recupera l'indirizzo dell'istruzione i+1. L'indirizzo di i+1 era nel PC durante l'esecuzione di i ed è stato salvato sulla pila o su un registro e recuperato da return-from-interrupt
- Il processore informa la periferica dell'avvenuta ricezione dell'interrupt tramite la linea `INT_ACK` nel bus di controllo



Servizio delle interruzioni

- Mentre la routine chiamata è nota al programma chiamante, la routine di servizio delle interruzioni (ISR) è del tutto estranea (di solito) alla logica del programma chiamante. Occorre salvare i dati su registri condivisi prima di cominciare a eseguire la ISR
- Tipicamente il processore salva in automatico solo PC, PS (`Program Status`). L'intervallo fra arrivo dell'interruzione e partenza della ISR si chiama **tempo di latenza dell'interruzione**: si allungherebbe se si salvano più cose
- La ISR deve salvare i registri prima di usarli, e ripristinarli prima di concludere e rientrare
- Alcuni processori salvano in automatico l'intero banco dei registri (se sono pochi), oppure (più spesso) la ISR usa un banco di registri ausiliario
- Il meccanismo di interruzione è fondamentale per regolare le interazioni fra SO e processi, e per i sistemi di controllo che reagiscono a eventi esterni in **real-time**

Controllo delle interruzioni

- Alcune sequenze di esecuzione non devono subire interruzioni
- Vi sono meccanismi per abilitare/disabilitare le richieste di interruzioni
  - Nel processore: il bit IE (Interrupt Enable) nel registro PS. Quando IE=1 le richieste di interruzione dai dispositivi di I/O sono accettate e servite dal processore
  - Nell'interfaccia di I/O: bit analogo del registro di controllo. Se questo bit è 1 il dispositivo di I/O è abilitato a segnalare le richieste di interruzioni
- Sequenza di eventi relativi a una richiesta di interruzione:
  - 1. la periferica attiva la richiesta di interruzione (IRQ va alto); 2. il processore interrompe il programma, salva i registri PC e PS; 3. le interruzioni vengono disabilitate ponendo il bit IE a 0; 4. la routine ISR va in esecuzione, la periferica viene informata e IRQ va basso; 5. il return-from-interrupt ripristina PC e PS, e riabilita le interruzioni, si rientra al programma interrotto