Corso di Architettura degli Elaboratori e Laboratorio (M-Z)

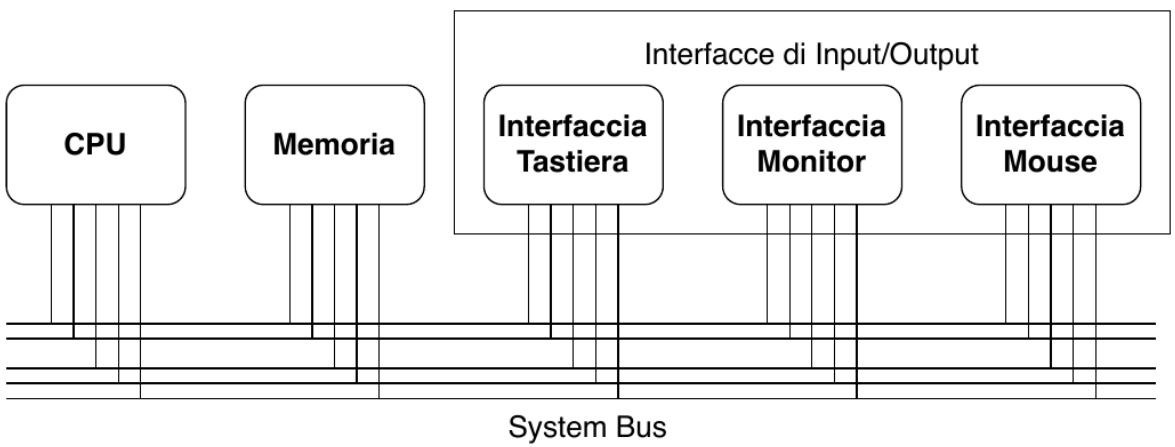
Operazioni di input e output

Nino Cauli

Dipartimento di Matematica e Informatica

Architettura base di un calcolatore elettronico

- **CPU**: esegue istruzioni elementari
- **MEMORIA**: contiene il programma (sequenza di istruzioni elementari) che la CPU deve eseguire e i dati necessari
- **INTERFACCE DI INPUT/OUTPUT**: circuiti elettronici che permettono di connettere la CPU al mondo esterno
- **BUS DI SISTEMA**: insieme di collegamenti elettrici che interconnettono I vari componenti di un calcolatore



Interfacce di Input/Output

- Un calcolatore ha necessita di comunicare con il mondo esterno

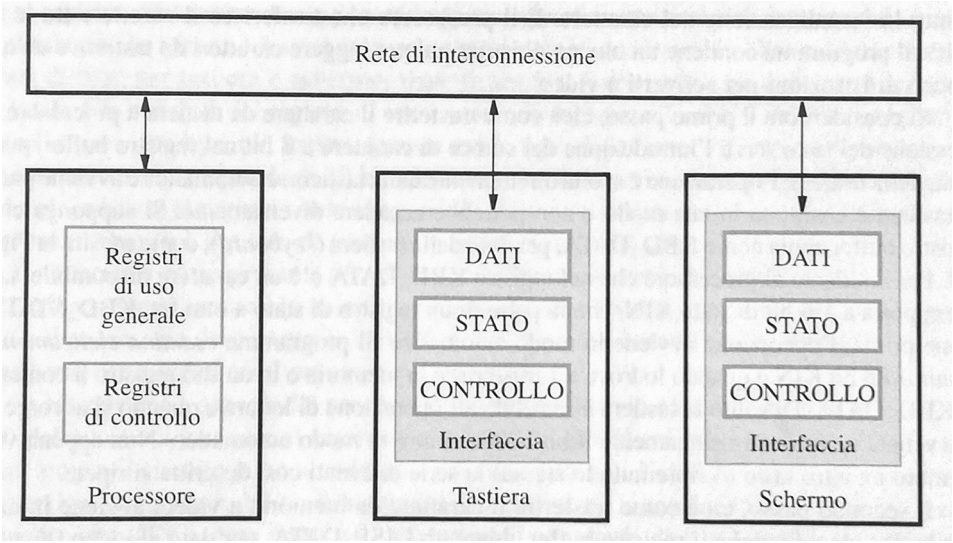
- Le interfacce di I/O sono tutti i circuiti elettronici che permettono alla CPU di interagire con l'utente:
  - Monitor
  - Tastiera
  - Mouse
  - Stampante
  - Connessioni di rete
  - ...

Interfacce di dispositivo

- Il collegamento tra le periferiche di I/O e il bus di sistema avviene tramite dei circuiti elettronici detti Interfacce di dispositivo

- L'interfaccia di ogni periferica contiene dei registri grazie ai quali è in grado di interagire con il processore attraverso il BUS di sistema

- Solitamente le interfacce di dispositivo contengono almeno i seguenti registri:
  - **Un registro DATI:** usato come buffer per il trasferimento dati
  - **Un registro STATO:** con informazioni sullo stato corrente del dispositivo
  - **Un registro CONTROLLO:** con informazioni per controllare il modo di operare del dispositivo



Spazio di indirizzamento di I/O

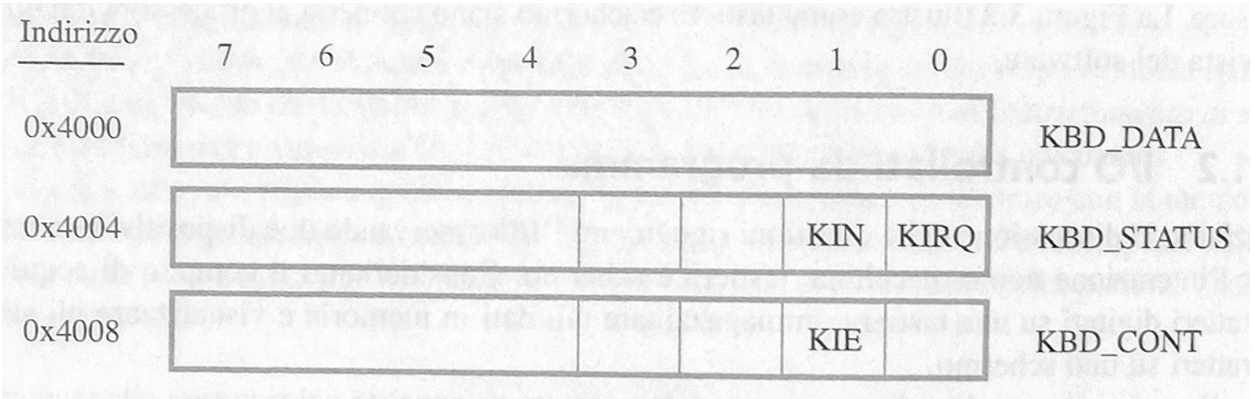
- I registri delle interfacce appaiono al processore come un insieme di locazioni indirizzabili (come le locazioni di memoria)
- Solitamente dispositivi di I/O e memoria condividono lo stesso spazio di indirizzi del processore
- Questa organizzazione è chiamata: **Memory Mapped I/O**
- In architetture di questo tipo, le funzioni di accesso alla memoria potranno accettare indirizzi di registri I/O o di memoria indifferentemente:

Load R2, DATO_ING (*i.e.* DATO_ING è il registro buffer della tastiera)

Store R2, DATO_USC (*i.e.* DATO_USC è il registro buffer dello schermo)

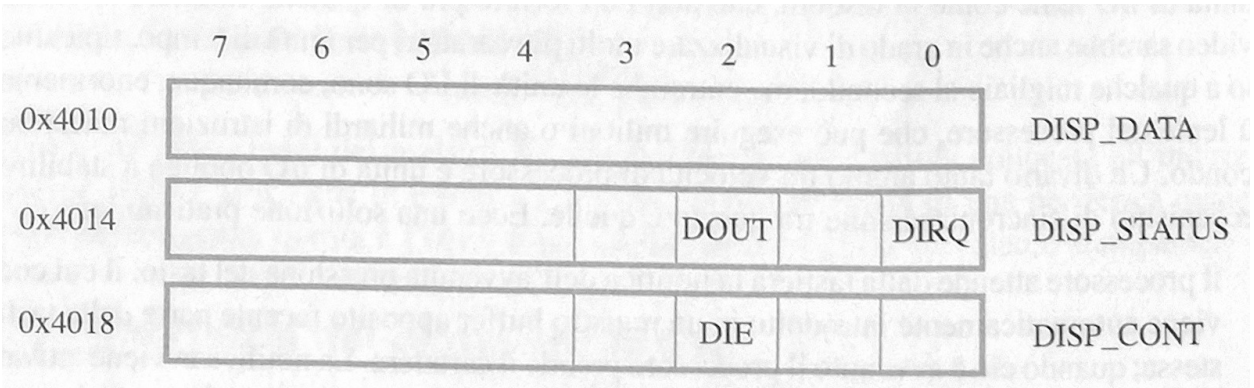
Interfaccia della tastiera

- La tastiera deve essere in grado di inviare al processore il carattere corrispondente al tasto premuto
- Assumiamo che l'interfaccia della tastiera possegga i seguenti registri a 8 bit:
  - **KBD_DATA**: registro Buffer in cui viene registrato il carattere una volta premuto il tasto
  - **KBD_STATUS**: registro di stato contenente i bit **KIN** (a 1 quando un nuovo carattere è pronto) e **KIRQ** (segnala una richiesta di interruzione)
  - **KBD_CONT**: registro di controllo contenente il bit **KIE** (bit di abilitazione di interruzione)



Interfaccia dello schermo

- Lo schermo deve essere in grado di mostrare dei dati (nel nostro caso caratteri) presenti in memoria a video
- Assumiamo che l'interfaccia dello schermo possegga i seguenti registri a 8 bit:
  - **DISP_DATA**: registro Buffer in cui viene registrato il dato da mostrare a video
  - **DISP_STATUS**: registro di stato contenente i bit **DOUT** (a 1 quando lo schermo è pronto a visualizzare un nuovo dato) e **DIRQ** (segnala una richiesta di interruzione)
  - **DISP_CONT**: registro di controllo contenente il bit **DIE** (bit di abilitazione di interruzione)



I/O controllati da programma

- Il modo più semplice di realizzare un programma in grado di gestire lo scambio di dati tra dispositivi I/O e processore è tramite la tecnica di **I/O controllati da programma**
- In questo caso il processore resta in attesa che la periferica I/O sia pronta prima di scambiare i dati
- Poco efficiente: Il processore rimane bloccato in attesa della risposta da parte della periferica I/O
- Facile da implementare: è sufficiente controllare i bit di stato delle periferiche per sapere quando iniziare lo scambio di dati
- Le periferiche più veloci dovranno rimanere in attesa di quelle più lente

Esempio programma tastiera/schermo

- Programma in grado di ricevere una stringa di caratteri da tastiera e visualizzarli man mano a video
- Il programma attenderà l'invio di ogni carattere da tastiera e lo visualizzerà su schermo appena ricevuto. Il programma terminerà appena ricevuto il carattere di Ritorno carrello da tastiera.
- Il programma sarà formato da due blocchi principali: lettura di carattere da tastiera e visualizzazione su schermo

LETTURA	**Legge la condizione di stato KIN**	**SCRITTURA**	**Legge la condizione di stato DOUT**
	Salta a LETTURA se KIN = 0		**Salta a SCRITTURA se DOUT = 0**
	Trasferisce i dati da KBD_DATA a Ri		**Trasferisce i dati da Ri a DISP_DATA**

Esempio programma tastiera/schermo RISC

	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

TestBit (CISC)

- L’istruzione TestBit è usata per controllare se un bit specifico di un registro o locazione di memoria sia uguale a 1 o meno

TestBit destinazione, #k

- Se il bit b_k dell’operando destinazione è uguale a 0 mette il bit di condizione Z a 0 altrimenti mette Z a 1

- Per valutare la condizione del bit di stato KIN possiamo eseguire l’istruzione:

TestBit KBD_STATUS, #1

Compare (CISC)

- L’istruzione Compare è usata per comparare il contenuto di due locazioni

Compare destinazione, sorgente

- L’istruzione compare sottrae il contenuto di sorgente al contenuto di destinazione e aggiorna i bit di condizione sulla base del risultato. Il risultato viene scartato:

- **CompareByte** esegue la stessa operazione ma su singoli byte

- La seguente istruzione controlla se l’indirizzo di memoria puntato da R2 contenga o meno il carattere di Ritorno Carrello:

CompareByte (R2), #CR

Esempio programma tastiera/schermo CISC

	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	

Tecnica di interruzione

- Con la tecnica di I/O controllati da programma il processore rimane bloccato finché la periferica da usare non è pronta
- Solitamente le periferiche sono in grado di allertare il processore quando sono pronte, lasciando il processore libero di eseguire altre istruzioni nel frattempo
- Questi “avvisi” vengono chiamati segnali di **INTERRUZIONE**
- Il bus di controllo contiene una linea dedicata a tale funzione: **Interrupt Request (INT_REQ)**
- Quando viene lanciato un segnale di interruzione il processore interrompe l’esecuzione del programma, salva il suo stato e salta all’esecuzione della **routine del servizio di interruzione**
- Terminata l’interruzione il processore riprende l’esecuzione del programma originario

Proprietà delle routine di servizio di interruzione

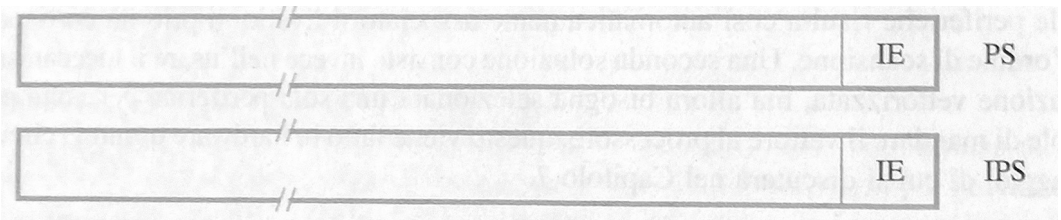
- La routine di servizio di interruzione è estranea al programma interrotto, a differenza del sottoprogramma chiamato

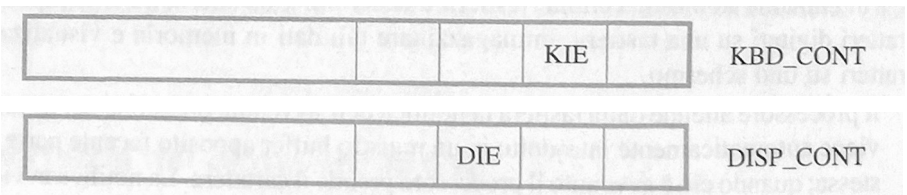
- Quando la routine viene chiamata bisogna salvare le informazioni necessarie a riprendere il programma principale:
  - Registro **PS (Program Status)**: registro contenente informazioni quali i bit di esito
  - Il contenuto dei registri temporanei usati dalla routine
  - Il contenuto di locazioni di memoria condivise dalla routine e dal programma interrotto
  - Il contenuto del registro **PC**

- Solitamente il processore salva automaticamente una copia di PS durante un interruzione mentre gli altri dati sono salvati dalla routine di servizio

Controllo dell'interruzione

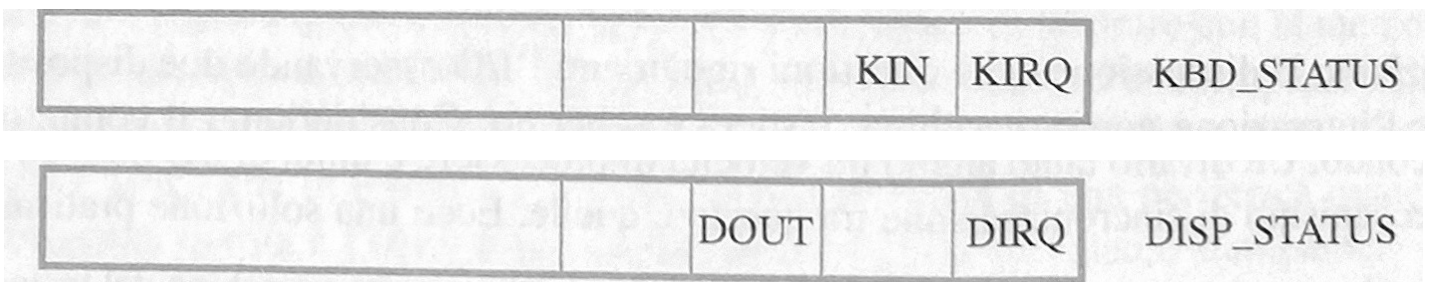
- Deve esistere un meccanismo che permetta sia dal lato processore che dal lato interfaccia di attivare o disattivare le interruzioni. I bit di attivazione di interruzione **IE** hanno questo ruolo
- Nel lato processore il bit **IE** si trova nel registro di stato **PS** e, quando è a 1, abilita le risposte alle interruzioni da parte del processore
- Nel lato interfaccia I/O il bit **IE** si trova nel registro di controllo e abilita le chiamate di interruzione da parte del dispositivo
- Nel registro **IPS** viene salvata automaticamente una copia di PS al momento della risposta ad un interruzione da parte del processore





Gestione di dispositivi multipli

- Per rispondere a richieste di interruzione da parte di più dispositivi il processore deve essere in grado di riconoscere il dispositivo richiedente l'interruzione
- Le interfaccie I/O presentano un bit di richiesta di interruzione **IRQ** nel loro registro di stato
- Un modo semplice per rispondere ad un segnale di interruzione è scansionare i registri di stato di tutti i dispositivi e lanciare la routine di servizio del dispositivo con il bir **IRQ** a 1
- Seppur semplice, questo metodo consuma parecchio tempo

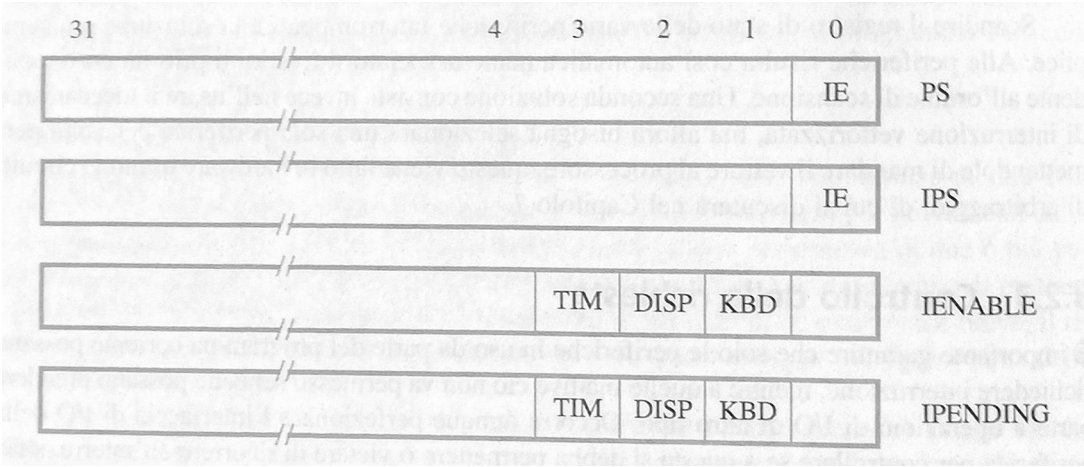


Interruzione vettorizzata

- Un modo più efficiente di gestire le interruzioni è attraverso l'**interruzione vettorizzata**
- La periferica interrompente manda nel BUS un **codice univoco di identificazione**
- Il codice di pochi bit (4-8) è un indice della **tabella dei vettori d'interruzione**
- Ogni campo della tabella contiene l'indirizzo iniziale della routine di servizio di un dispositivo
- Quando il processore riceve il codice di interruzione, carica sul registro PC l'indirizzo corrispondente nella tabella dei vettori, facendo partire la routine di servizio corretta

Registri di controllo del processore

- **PS**: contiene informazioni sullo stato del programma (bit di esito, bit di attivazione di interruzione, bit di priorità, etc.)
- **IPS**: copia del registro PS salvata prima di servire un'interruzione
- **IENABLE**: contiene un bit per ogni dispositivo I/O. Ciascun bit è messo a 1 se il processore intende rispondere alle richieste di interruzione del dispositivo
- **IPENDING**: contiene un bit per ogni dispositivo I/O. Ciascun bit è messo a 1 se il dispositivo ha una richiesta di interruzione in attesa di risposta



Accesso ai registri di controllo

- Non si può accedere ai registri di controllo tramite le istruzioni aritmetiche e logiche
- Per leggere e scrivere sui registri di stato si usa l'istruzione speciale **MoveControl**

Esempi:

- Per caricare il contenuto di PS nel registro R2:

MoveControl R2, PS

- Per copiare il contenuto di R3 nel registro IENABLE:

MoveControl IENABLE, R3

Annidamento interruzioni

- Alcuni tipi di periferica non possono attendere troppo a lungo prima di essere serviti
- Si ha bisogno di un metodo per gestire interruzioni annidate
- Una soluzione consiste nel usare un sistema di priorità:

1) Si assegna al processore un livello di priorità corrente, si usano dei bit nel registro PS
2) Se un dispositivo con priorità maggiore lancia un'interruzione viene servito e i bit di priorità in PS vengono aggiornati con il nuovo livello di priorità
3) Quando si rientra dall'interruzione il processore riassume il livello di priorità precedente contenuto nella copia di PS ricaricata