Dispositivi per l'ingresso-uscita (I/O)

Roberto Negrini Giuseppe Pozzi

DEI, PoliMI, Milano

roberto.negrini,giuseppe.pozzi@polimi.it

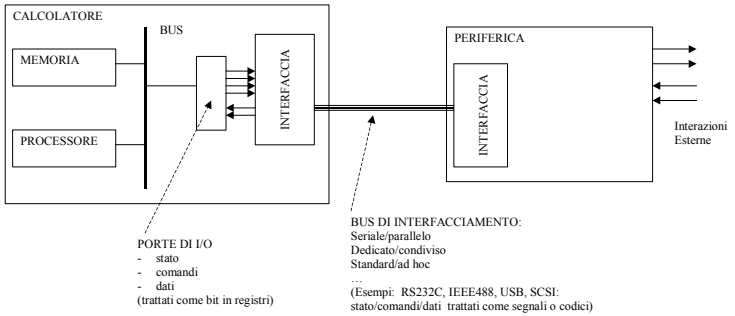
- versione dell'11 aprile 2003 -

L'ingresso-uscita

Bibliografia:

- Tanenbaum A. S., Goodman J. R, "Architettura dei computer - Un approccio strutturato", Prentice Hall, 2000.
- Hamacher V. C., Vranesic Z. G., Zaky S. G., "Introduzione all'architettura dei calcolatori", McGraw-Hill, 1997 (capitolo 4).

Collegamento di periferica a calcolatore

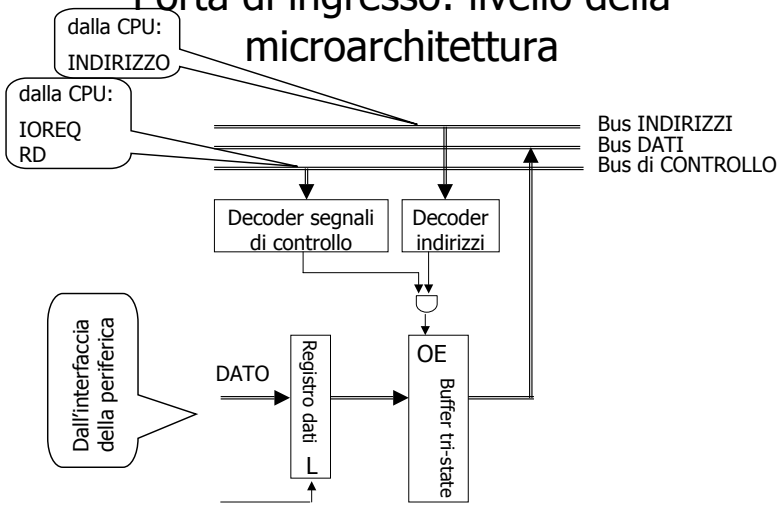


Buffer tri-state

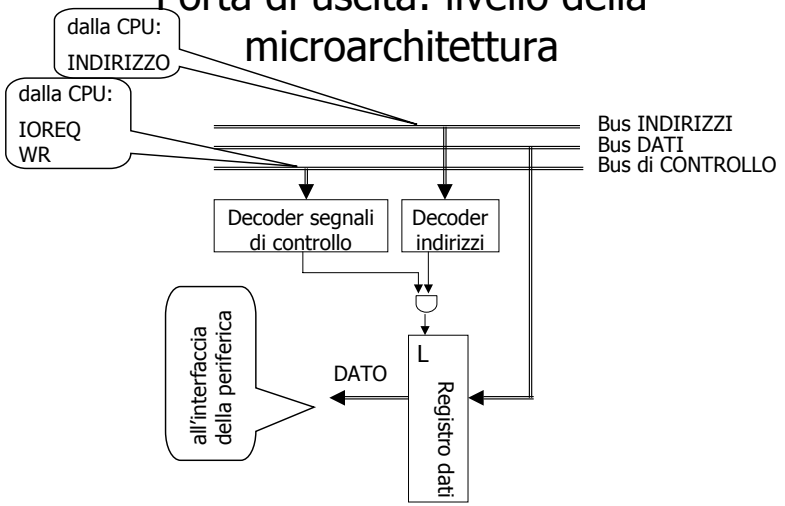
E' il circuito elementare modellabile come un contatto a tre posizioni:

- in stato di bassa impedenza consente di avere in uscita o il **livello alto** (1) o il **livello basso** (0)
- in stato di alta impedenza (Z) **isola elettricamente** l'uscita
- l'uscita tri-state viene gestita da un apposito ingresso di controllo (**O**uput **E**nable) che, se non attivo, forza lo stato di alta impedenza

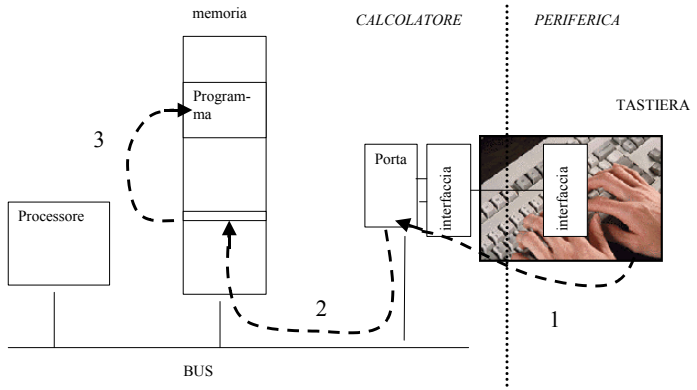
Porta di ingresso: livello della microarchitettura



Porta di uscita: livello della microarchitettura



Esempio di ingresso

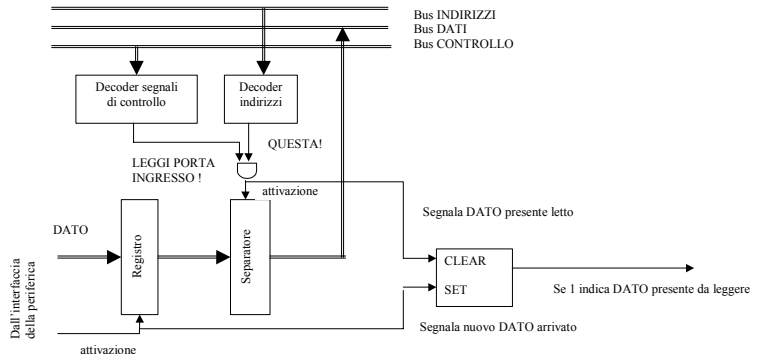


Esigenze

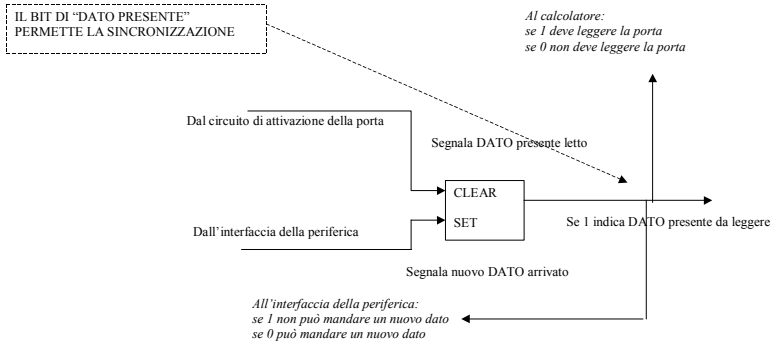
- Evitare perdite o duplicazioni di dati.
- Consentire comunicazioni asincrone.
- Nel caso di lettura da tastiera, le comunicazioni sono:
  - da tastiera a porta;
  - da porta a cella di memoria;
  - da cella di memoria a programma che utilizza il dato;

  indipendentemente dalle modalità di gestione di I/O adottata.

Da tastiera a porta



Da periferica a porta



Da porta a cella di memoria

- Se nessun ciclo di bus legge la porta, il dato rimane presente all'infinito e non entra mai nel calcolatore.
- Il ciclo di bus è gestito a livello di microarchitettura.
- A livello ISA, la acquisizione del dato avviene:
  - a) a controllo di programma;
  - b) a interruzione;
  - c) ad accesso diretto alla memoria (DMA: direct memory access).

A controllo di programma

- Durante la sua normale esecuzione un programma esegue una istruzione di lettura della porta:
  es Intel: `IN R0, INDIRIZZOPORTA`
  es Motorola: `MOV INDIRIZZOPORTA, R0`
- Nella fase di esecuzione di questa istruzione il processore esegue il ciclo di bus di lettura della porta.
- Il programmatore ha deciso dove, nel programma, inserire questa istruzione.
- Il flusso dell'esecuzione del programma stabilirà **quando** l'istruzione verrà eseguita.

A interruzione

- La parte di programma che legge la porta (ad es. con la istruzione `IN R0,INDIRIZZOPORTA`) **NON** è nel programma ma è silente in memoria in una locazione convenuta.
- Quando l'interfaccia della periferica porta il dato alla porta di ingresso, con un segnale allerta il processore.
- Il processore interrompe l'esecuzione del programma in corso e salta automaticamente a eseguire la parte di programma che legge la porta. La lettura avviene come nel caso precedente.
- Al termine di questo, il processore riprende il programma interrotto.
- In pratica, la periferica ha deciso **quando** l'istruzione di lettura della porta deve essere eseguita.

DMA

- Quando l'interfaccia della periferica porta il dato alla porta di ingresso, manda un segnale al processore, imponendogli di lasciare libero il bus.
- Appositi circuiti generano un ciclo di bus che forza l'attivazione della porta, genera l'indirizzo in memoria dove deve finire il dato, comanda la memoria alla scrittura.
- Intanto, il processore non utilizza il bus.
- Terminato il ciclo, l'interfaccia della periferica manda un altro segnale al processore, lasciandolo libero di proseguire.
- In pratica, alcuni circuiti di I/O hanno scritto il dato in memoria, pochi nanosecondi dopo il suo arrivo.

Dalla cella al programma

- Il programma che aspetta il dato deve essere allertato.
- Usualmente questa fase è realizzata con le funzionalità del sistema operativo, che è deputato alla gestione corretta dell'I/O per conto dei programmi e allo scambio di dati tra programmi.

Dalla cella al programma

- Esempio:
  - Il programma che richiede un carattere alla tastiera esegue una chiamata al sistema operativo.
  - Il sistema operativo:
    - sospende il programma;
    - gestisce l'ingresso del carattere dalla porta a una cella di memoria (per es. a interruzione);
    - riprende poi l'esecuzione del programma, passandogli l'indirizzo della cella in cui ora si trova il carattere.

Dalla cella al programma

- Nel caso di sistemi operativi a multiprogrammazione/time-sharing, in attesa del carattere il sistema operativo può far procedere l'esecuzione di altri programmi che non usino la tastiera.

(Nota: Nel caso più semplice, è il programma stesso che esegue l'ingresso a controllo di programma e acquisisce il tasto.)

Analizziamo le 3 modalità

- Controllo da programma;
- interruzione;
- DMA.

L'ingresso-uscita (Input/Output)

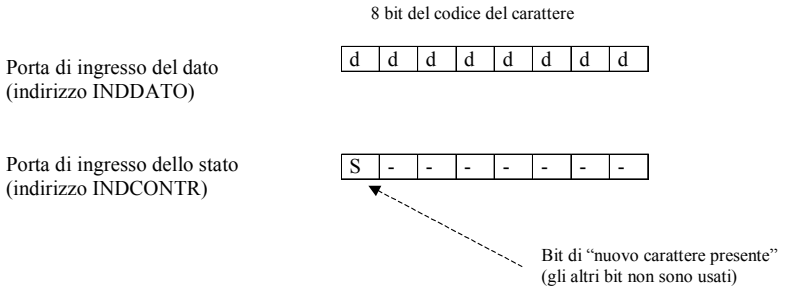
(parte 1: controllo da programma)

A controllo da programma (ingresso)

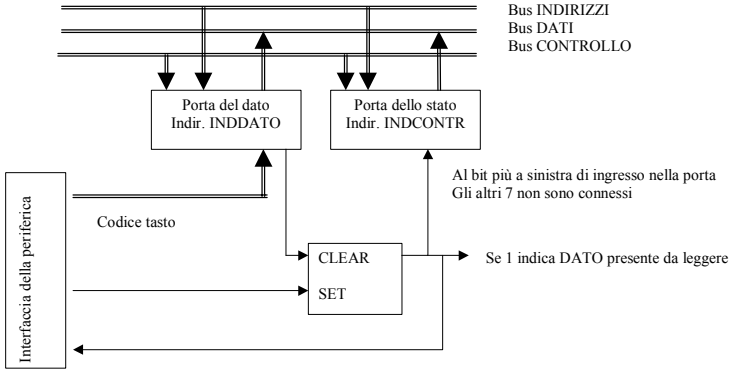
- Esiste un apposito programma che:
  - legge le porte a cui si presentano i bit di stato ed i dati dell'interfacia della periferica;
  - scrive le porte che forniscono bit di comando all'interfaccia di periferica.

In questo caso, il programma che acquisisce il dato, lo memorizza e lo usa è unico.

La tastiera: controllo da programma



Le porte della tastiera



A controllo da programma

LEGGITASTO	IN R0,INDCONTR	input dello stato della tastiera in registro R0
	AND R0,MASKCA	in R0 rimane solo il bit di *codice arrivato* (MASKCA = 10000000)
	JZ LEGGITASTO	se il bit di *codice arrivato* è 0, ripeti lettura finché non lo trovi a 1
	IN R0,INDDATO	*codice di tasto* è arrivato e qui viene letto dalla porta e scritto in R0
	MOV TASTO,R0	copia *codice di tasto* in cella di memoria TASTO
	…	Istruzioni successive (hanno a disposizione nella cella TASTO il *codice di tasto* arrivato)

A controllo di programma (ingresso)

- Ideale se:
  - il ciclo di istruzioni per la lettura è inserito nel programma utente;
  - quando il programma utente è in attesa che venga premuto un tasto, non ha null'altro da fare;
  - non c'è nessun altro programma utente da eseguire;
  - quindi la soluzione di ripetere all'infinito il ciclo di istruzioni per la lettura non ha controindicazioni.

A controllo di programma

- La soluzione a controllo di programma è molto veloce:
  - viene eseguito un ciclo di tre istruzioni per sentire se il codice è arrivato;
  - vengono eseguite due istruzioni per acquisire il codice e scriverlo in memoria.

A controllo di programma

- Nel caso più fortunato, il codice arriva proprio prima dell'esecuzione della istruzione `IN R0,INDCONTR` e quindi con 5 istruzioni il codice è in memoria.
- Nel caso più sfortunato, il codice arriva proprio dopo l'esecuzione della istruzione `IN R0,INDCONTR` e quindi si eseguono le due istruzioni successive e poi si ricicla, stavolta leggendo il codice: con 7 istruzioni il codice è in memoria.

A controllo di programma (uscita)

- Stampa di un carattere.
  - L'interfaccia della stampante tramite un bit comunica che è pronta ad accettare un nuovo codice di carattere da stampare.
  - Occorrono circuiti analoghi a quelli visti per la tastiera:
    - una porta di ingresso, per leggere il bit di pronta ad accettare (indirizzo `INDCONTRST`);
    - una porta di uscita, su cui il programma scrive il codice del carattere (indirizzo `INDDATOST`);
    - il circuito che genera il bit di pronto ad accettare (analogo a quello che generava il bit di codice arrivato).

A controllo di programma

STAMPAC	IN R0,INDCONTRST	input dello stato della stampante in registro R0
	AND R0,MASKPA	in R0 rimane solo il bit di *pronta ad accettare* (MASKPA = 10000000)
	JZ STAMPACAR	se il bit di *pronta ad accettare* è 0, ripeti lettura finché non lo trovi a 1
	MOV R0,CARATT	copia *codice di carattere* da cella di memoria CARATT in R0
	OUT INDDATOST,R0	*codice di carattere* da R0 è scritto sulla porta di uscita
	...	Istruzioni successive

L'ingresso-uscita (Input/Output)

(parte 2: gestione a interruzione)

Gestione a interruzione

- Ricordando che se nessun ciclo di bus legge la porta, il dato continua a rimanere disponibile all'infinito e non entra mai nel calcolatore, la gestione a interruzione permette il passaggio di un dato:
  - da una porta di ingresso a una cella di memoria (ingresso);
  - da una cella di memoria a una porta di uscita (uscita).

Funzionamento a interruzione

- La parte di programma che legge la porta (ad es. con l'istruzione `IN R0,INDIRIZZOPORTA` la quale nella sua esecuzione provoca il ciclo di bus che abilita la porta) **NON** è nel programma ma è silente in memoria, isolata, in un luogo di memoria convenuto.
- Quando l'interfaccia della periferica pone il dato alla porta di ingresso, con un segnale allerta il processore.
- Il processore interrompe l'esecuzione del programma in corso e salta automaticamente a eseguire la parte di programma che legge la porta. La lettura avviene come nel caso precedente.

Funzionamento a interruzione

- Al termine di questo, il processore riprende il programma interrotto.
- In pratica, la periferica ha deciso quando l'istruzione di lettura della porta deve essere eseguita.
- L'interruzione "scatena l'esecuzione del programma di risposta all'interruzione" (o "routine di servizio").

La gestione a interuzione

- Richiede componenti hardware e software addizionali a livello di:
  - sistema;
  - ISA;
  - microarchitettura

sia nel calcolatore che nella interfaccia della periferica.

Ampliamenti nel calcolatore per gestire l'interruzione

- A livello di sistema:
  - convenzioni per l'identificazione della periferica interrompente e della posizione in memoria del programma di risposta corrispondente;
  - convenzioni di attribuzione di priorità differenti ai diversi tipi di interruzione e ai diversi tipi di periferica per gestire in modo corretto le interruzioni contemporanee;
  - convenzioni di interrompibilità (priorità) a livelli diversi delle parti del sistema operativo e dei programmi utenti.

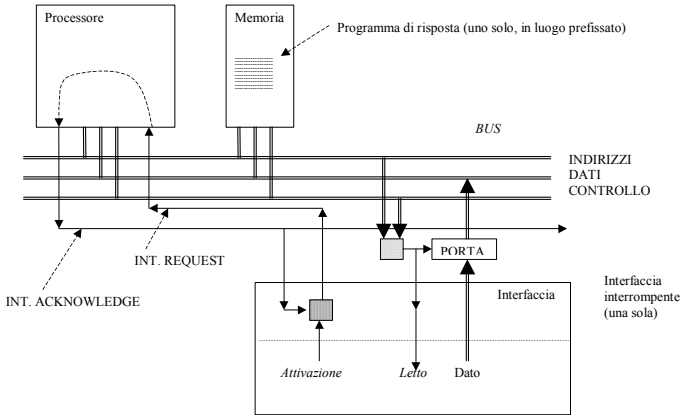
Ampliamenti nel calcolatore per gestire l'interruzione

- A livello ISA:
  - istruzioni per la gestione dei livelli di interrompibilità (priorità) e per il rientro dalle risposte alle interruzioni;
- A livello di microarchitettura:
  - hardware per gestire le richieste e le accettazioni di interruzione, e per identificare l'interrompente;
  - hardware per gestire le priorità di interruzione;
  - hardware per il salvataggio dello stato di esecuzione (es: PC, PSW), e per il successivo ripristino;
  - hardware per forzare l'esecuzione della risposta all'interruzione.

Ampliamenti nella periferica per gestire l'interruzione

- Hardware per gestire le richieste e le accettazioni delle interruzioni (anche di tipo diverso, se diverse sono le richieste di attenzione che la periferica può generare), anche in presenza di più periferiche interrompenti.

Una sola periferica ed un solo livello di interruzione



Fasi dell'interruzione

- Condizioni esemplificatrici:
  - una sola periferica che esegue interruzione;
  - un solo programma in esecuzione;
  - una sola azione da compiere in risposta all'interuzione;
  - un calcolatore senza sistema operativo

Fasi dell'interruzione

- E' importante tenere presente che in questa descrizione delle fasi si mescolano azioni a due livelli ben diversi:
  - al livello della microarchitettura: AZIONI HARDWARE.
    - sono azioni compiute dallo hardware del processore durante l'esecuzione di una istruzione e durano tipicamente qualche nanosecondo. Hanno durata superiore, se comprendono cicli di bus per la lettura o scrittura di una cella di memoria o di una porta.
  - al livello ISA: AZIONI SOFTWARE.
    - sono azioni definite al livello delle istruzioni nei programmi. La loro durata dipende dal numero delle istruzioni eseguite.

Fase 0: abilitazione e avvio

- Al livello ISA, il programma che è in esecuzione e ha bisogno di un dato dalla porta interrompente:
  - scrive 0 nella variabile `ARRIVATO` (una cella di memoria). Il programma di risposta scriverà 1 per segnalare l'avvenuto arrivo in memoria del dato. Per il dato è predisposta la cella `INPUTDATO`;
  - con apposite istruzioni abilita l'interruzione;
  - e continua eseguendo parti che non hanno bisogno del dato (ci devono essere, altrimenti era meglio ricorrere al controllo a programma... che è più semplice e molto più veloce).

Fase 1: interruzione

- La periferica ha il dato pronto. A livello di microarchitettura:
  - l'interfaccia pone il dato alla porta;
  - l'interfaccia attiva il segnale `ATTIVAZIONE`;
  - i segnale `INTAKNOWLEDGE` non è attivo; l'interfaccia attiva il segnale `INTREQUEST`;
  - il segnale `INTREQUEST` arriva al processore;
  - il processore è interrompibile, e attiva il segnale `INTACKNOWLEDGE`;
  - l'interfaccia riceve questo segnale, e disattiva il segnale `INTREQUEST`;
  - il processore termina l'esecuzione dell'istruzione in corso;
  - il processore salva lo stato (per esempio, copia nella pila in memoria i valori di PC, PSW,...): ciò può richiedere alcuni cicli di bus per scrivere in memoria;
  - il processore forza nel PC l'indirizzo di inizio (noto per convenzione) del programma di risposta;
  - il processore inizia il fetch della prossima istruzione (la prima della risposta).

Fase 2: esecuzione della risposta

- A livello ISA:
  - il processore esegue il programma di risposta, che contiene decine di istruzioni, e tra queste c'è l'istruzione di lettura della porta.
- A livello della microarchitettura:
  - durante la fase di esecuzione dell'istruzione di lettura della porta, un ciclo di bus abilita la porta;
  - il segnale `LETTO` di abilitazione della porta arriva all'interfaccia;
  - l'interfaccia disabilita il segnale `ATTIVAZIONE`;
  - l'interfaccia ritiene consegnato il `DATO`, e può iniziare la fornitura del dato successivo, se c'è già.

Fase 2: esecuzione della risposta

- A livello ISA:
  - il processore continua l'esecuzione del programma di risposta;
  - una istruzione copia il dato nella cella di memoria `INPUTDATO`;
  - una istruzione scrive 1 nella cella di memoria `ARRIVATO`;
  - il processore esegue l'ultima istruzione (`RETURNFROMINTERRUPT`).

Fase 3: il rientro dall'interruzione

- Al livello della microarchitettura:
  - durante la fase di esecuzione dell'istruzione `RETURNFROMINTERRUPT`, il processore ripristina lo stato salvato (per es. rimette in PC, PSW, ... i valori che avevano prima dell'interruzione, togliendoli dalla pila in memoria): ciò può richiedere alcuni cicli di bus di lettura in memoria;
  - automaticamente, il processore come prossima istruzione preleverà quella che avrebbe dovuto prelevare se non fosse arrivata l'interruzione. ---

Fase 3: il rientro dall'interruzione

- Al livello ISA:
  - il programma che era stato interrotto riprende in modo corretto;
  - continua l'esecuzione delle parti che non hanno bisogno del dato;
  - arriva a eseguire la parte che ha bisogno del dato;
  - controlla la variabile `ARRIVATO`;
  - la trova a 1;
  - quindi il dato è in memoria, nella cella `INPUTDATO`;
  - il programma continua, usandolo.
  - (se `ARRIVATO` vale ancora 0, allora l'interruzione non c'è ancora stata e il dato non è arrivato: se non ha altro da fare , il programma può solo ciclare leggendo `ARRIVATO` finché non lo trova a 1);
  - con apposite istruzioni, disabilita l'interruzione, che non gli serve più.

Commenti

- Al livello ISA:
  - Il programma di risposta non deve interferire con le variabili del programma interrotto, e non deve toccare lo stato salvato.
  - P.es. se PC e PSW sono salvati in cima alla pila, il programma di risposta può usare la pila ma poi al termine in cima alla pila ci devono essere ancora i valori salvati di PC e PSW, altrimenti l'istruzione `RETURNFROMINTERRUPT` non ripristina correttamente PC e PSW.
  - Quindi il programma di risposta può contenere istruzioni che usano la pila come `PUSH` e `POP`, `CALL` e `RETURN` di procedure, ma alla fine devono essere bilanciate (tante `PUSH` e `CALL` quante `POP` e `RETURN`) prima della `RETURNFROMINTERRUPT` finale.

Commenti

- Al livello ISA:
  - Nell'esempio illustrato il programma di risposta è molto semplice (deve solo leggere una porta e segnalare al programma utente che l'ha letta).
  - In altri casi l'interruzione può essere usata da una periferica per costringere il processore a eseguire azioni molto più complesse (p. es. il programma che gestisce una situazione di allarme in un impianto).

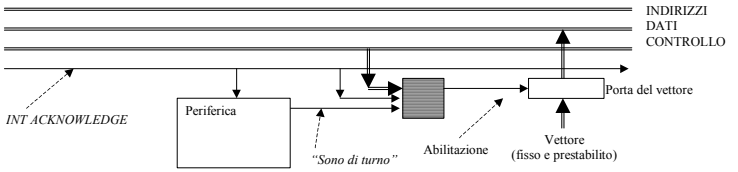
Casi reali

- Più periferiche, più livelli di interruzione:
  - modi per consentire al processore di identificare la periferica interrompente durante la fase hardware di accettazione dell'interruzione, per scegliere la procedura di risposta giusta;
  - modi per attribuire priorità differenti alle varie periferiche, magari raggruppandole su differenti livelli di priorità;
  - modi per consentire al processore di non accettare interruzioni al sotto di un certo livello di priorità;
  - modi per gestire interruzioni annidate (un programma di risposta è a sua volta interrotto, prima che sia terminato, da una richiesta di interruzione a livello di priorità superiore);
  - arbitri hardware per gestire le richieste di interruzione che nascono contemporaneamente, per rispondere alla più prioritaria e lasciare pendenti le altre.

Vettore di interruzione

- A livello microarchitettura:
  - durante la fase di accettazione di una interruzione, il processore attiva il segnale `INTACKNOWLEDGE` e genera un ciclo di bus di lettura senza indirizzo;
  - questo ciclo è identificato da appositi segnali di controllo;
  - solo la periferica interrompente che è di turno (tra tutte le contemporaneamente interrompenti è quella più prioritaria) partecipa a questo ciclo di bus, ponendo sul bus dati un numero che la identifica (il **vettore dell'interruzione**);
  - il processore usa il vettore per determinare l'indirizzo in memoria della procedura di risposta corrispondente.

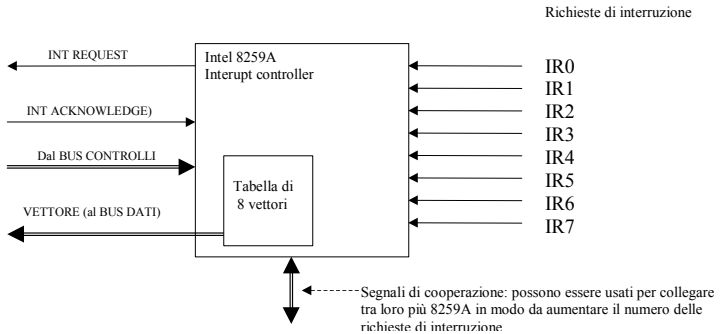
Esempio



Un controllore di interruzioni

- Intel 8259A:
  - riceve 8 richieste separate (una per interfaccia di periferica);
  - ha l'ordine di priorità, ha 8 vettori di interruzione (uno per periferica);
  - gestisce la richiesta/ accettazione di interruzione col processore e passa al processore il vettore della periferica di turno.

Intel 8259A



L'ingresso-uscita (Input/Output)

(parte 3: gestione a DMA)

Gestione a DMA

- La gestione a DMA (accesso diretto alla memoria) del passaggio di un dato:
  - da una porta di ingresso a una cella di memoria (ingresso);
  - da una cella di memoria a una porta di uscita (uscita).
- Consideriamo ancora il nostro esempio di ingresso.

Gestione a DMA

- Quando l'interfaccia della periferica porta il dato alla porta di ingresso, manda un segnale al processore, imponendogli di lasciare libero il bus.
- Non appena il bus è libero, appositi circuiti dell'interfaccia di periferica generano **un ciclo di bus** che:
  - forza l'attivazione della porta, che pone il dato sul bus dati;
  - genera l'indirizzo della cella di memoria in cui il dato deve essere scritto;
  - comanda la memoria alla scrittura (dal bus dati alla cella di memoria).

Gestione a DMA

- Intanto, il processore è fermo, bloccato, isolato dal bus, non deve fare nulla (non si salva lo stato).
- Terminato il ciclo, l'interfaccia della periferica disattiva il segnale di richiesta al processore, lasciandolo libero di riattivarsi.
- In pratica, dei circuiti di I/O hanno scritto il dato in memoria **pochi nanosecondi dopo il suo arrivo** (il tempo richiesto per eseguire un ciclo di bus).
- Il DMA "ruba" un ciclo di bus al processore, che **non partecipa** al passaggio del dato dalla periferica alla memoria. Il trasferimento avviene tutto a livello di microarchitettura.

Osservazioni sul DMA

- Il vantaggio principale del controllo a DMA consiste nei tempi molto rapidi di trasferimento, e quindi permette "raffiche" di trasferimenti molto rapide e intense di blocchi di dati (utili in molte periferiche: ad es. nei controllori dei dischi una raffica può trasferire molto velocemente un settore intero);
- i circuiti necessari nelle interfacce sono complessi, ma che aumentandone di poco la complessità si ottengono circuiti in grado di controllare sequenze di trasferimenti (una "raffica", appunto), perché basta usare un registro (ad autoincremento) contenente l'indirizzo di memoria dove deporre il singolo dato della raffica, e un registro (ad autodecremento) per contare il numero di dati che rimangono da trasferire nella raffica;
- di conseguenza, nelle architetture solite, le interfacce a DMA gestiscono raffiche di trasferimenti (ogni raffica, un blocco di dati: DMA a blocchi).

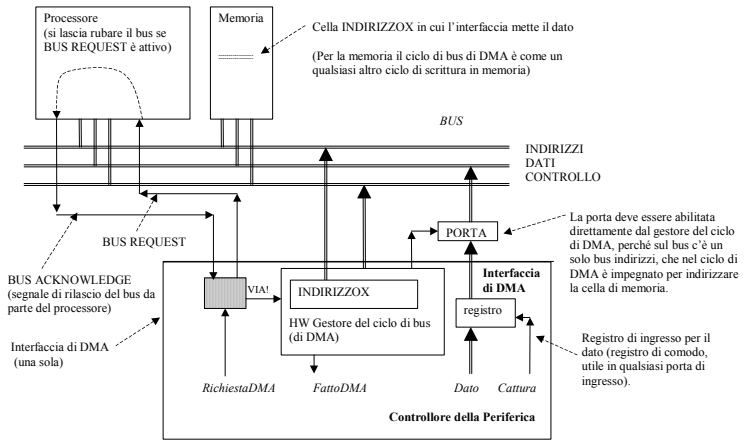
Ampliamenti nel calcolatore per gestire il DMA

- A livello di sistema:
  - convenzioni per l'identificazione della posizione in memoria della cella che riceve il dato;
- A livello ISA:
  - nulla;

Ampliamenti nel calcolatore per gestire il DMA

- A livello di microarchitettura:
  - hardware per gestire le richieste e le accettazioni di DMA (nel processore e nell'interfaccia della periferica);
  - hardware per gestire un ciclo di bus (nell'interfaccia della periferica);
  - hardware per impostare e abilitare il trasferimento.

DMA



Ampliamenti nel calcolatore per gestire più DMA

- A livello di sistema:
  - convenzioni di attribuzione di priorità differenti alle diverse periferiche per gestire in modo corretto le possibili richieste di DMA contemporanee;
- A livello ISA:
  - nulla per il trasferimento effettivo;
  - programmi per l'avvio e l'abilitazione del DMA;

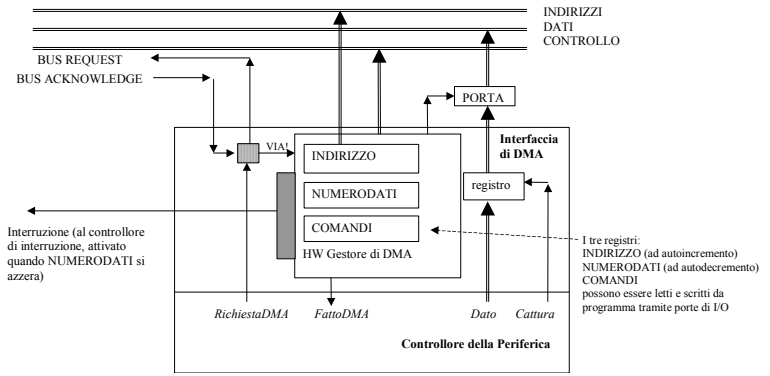
Ampliamenti nel calcolatore per gestire più DMA

- A livello di microarchitettura:
  - hardware per gestire le priorità di richiesta del bus, per gestire richieste contemporanee;
  - hardware per impostare e abilitare il trasferimento (quanti dati, e dove in memoria).

Osservazioni sul DMA

- Il furto di bus è molto breve;
- il ciclo di bus non può essere spezzato in parti;
- quindi per semplicità richieste contemporanee di DMA vengono accettate ma eseguite solo in successione. Padrona del bus è sempre la CPU, la periferica che ha chiesto e ottenuto il bus non è in grado di accettare richieste di DMA più prioritarie e di cedere temporaneamente a queste l'uso del bus, per poi riprenderselo quando questi DMA più prioritari sono finiti.

DMA per trasferimento a blocchi



DMA: Fasi per ingresso di blocco dati

- *1. Predisposizione.* Una procedura del Sistema Operativo scrive nei registri dell'interfaccia (accessibili come porte di uscita) quanti dati trasferire e l'indirizzo iniziale in memoria del buffer in cui scrivere il blocco di dati (fase a livello ISA, svolta da un programma).
- *2. Attivazione.* La procedura del Sistema Operativo scrive nel registro di controllo dell'interfaccia i bit che la attivano in scrittura (fase a livello ISA, svolta da un programma).

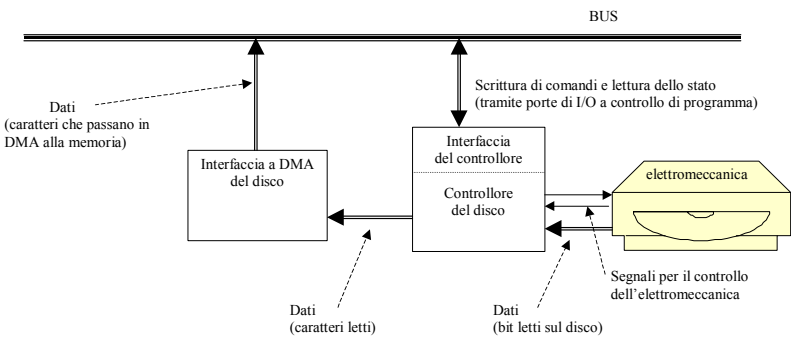
DMA: Fasi per ingresso di blocco dati

- *3. Trasferimento di un blocco di dati.* Ogni volta che dalla periferica arriva un dato all'interfaccia, l'interfaccia ruba al processore un ciclo di bus e in DMA scrive il dato in memoria; incrementa l'indirizzo in memoria, decrementa il numero di dati ancora attesi e, se non sono arrivati tutti, aspetta il prossimo dato (fase a livello microarchitettura, svolta da circuiti).
- *4. Fine (tipicamente a interruzione).* Se l'ultimo dato è arrivato, l'interfaccia segnala al processore che la raffica di DMA si è conclusa, e che il buffer in memoria è pieno coi nuovi dati. Nelle architetture usuali questo avviene tramite un segnale di interruzione (fase a livello microarchitettur, svolta da circuiti).

DMA: Fasi per ingresso di blocco dati

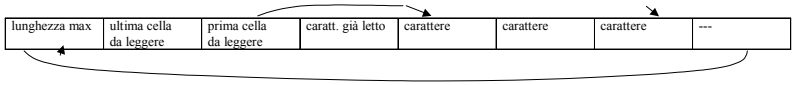
- *5. Chiusura (passaggio dei dati all'utente e disattivazione).* Una procedura del Sistema Operativo, attivata in risposta all'interruzione, gestisce il passaggio dei dati al programma utente che li attendeva, e gestisce la disattivazione del DMA, scrivendo nel registro di comando dell'interfaccia i bit che la disattivano (fase a livello ISA, svolta da un programma).

Passaggio dati al programma utente



Azioni del programma utente e del S.O.

- Il programma (p.es. un programma di gestione testi come Word) preleva i caratteri da un **buffer** in memoria, in cui li ha scritti l'interfaccia a DMA.



- Per esempio, il buffer contiene la copia di un settore di un file sul disco.

Azioni del programma utente e del S.O.

- Il sistema operativo mette a disposizione del programma (e del programma di risposta) una propria **procedura di gestione del buffer** (lettura ordinata di caratteri dal buffer, gestione dei puntatori).
- La procedura è eseguita come se fosse una procedura del programma e torna il carattere.
- Per prelevare il prossimo carattere dal buffer, il programma esegue una chiamata alla procedura di gestione del buffer.
- Se il buffer **non è vuoto** la procedura rende subito il carattere (il primo non ancora letto nel buffer) e il programma prosegue.

Azioni del programma utente e del S.O.

- Se il buffer **è vuoto**:
  - la procedura fa una chiamata al sistema operativo, che sospende l'esecuzione del programma e lo mette tra i programmi in attesa di evento esterno (marcandolo come in attesa di carattere da quel particolare buffer);
  - tramite le procedure e i dati del file system, il sistema operativo identifica il settore successivo del file sul disco, che quindi è quello che va letto, e scopre quanti caratteri contiene (potrebbe non essere tutto pieno) (i settori sono tutti di uguale lunghezza, e il buffer è lungo quanto un settore);
  - il sistema operativo attiva l'interfaccia di DMA con il numero di caratteri del settore da leggere e la posizione in memoria della prima cella del buffer da riempire coi caratteri in arrivo. Il sistema operativo aggiorna i puntatori nel buffer in modo che risulteranno giusti quando il DMA avrà messo i dati nel buffer;

Azioni del programma utente e del S.O.

- Se il buffer **è vuoto**:
  - attiva poi il controllore delle interruzioni, che dovrà gestire l'interruzione di fine DMA: l'interruzione da fine DMA (e il relativo programma di risposta) sono quindi pronti ad agire;
  - il sistema operativo comunica al controllore della periferica quale è il settore da leggere sul disco (numero di traccia/ numero di settore), e lo attiva.
- Comunque vada, il sistema operativo sceglie poi un altro programma (tra quelli pronti) e lo manda in esecuzione.

Esempio di funzionamento del DMA

- Il controllore del disco muove la testina posizionandola sulla traccia voluta. Il disco ruota, e il controllore del disco legge i vari settori. Finalmente sotto la testina passa il settore voluto, che viene letto dal controllore. Quando il singolo carattere è disponibile, il controllore del disco lo passa all'interfaccia di DMA e le chiede un ciclo di DMA;
- l'interfaccia di DMA scrive il carattere in memoria, nella cella di turno del buffer, incrementa l'indirizzo memoria, pronto per il successivo carattere, decrementa il conta caratteri;
- dopo aver letto l'ultimo carattere del settore, il controllore del disco smette di agire;
- ricevendo l'ultimo carattere, quando l'interfaccia di DMA riceve l'ultimo carattere del settore, lo scrive in memoria. Ora il suo conta caratteri si è azzerato;
- l'interfaccia di DMA genera un segnale di interruzione (interruzione di fine DMA).

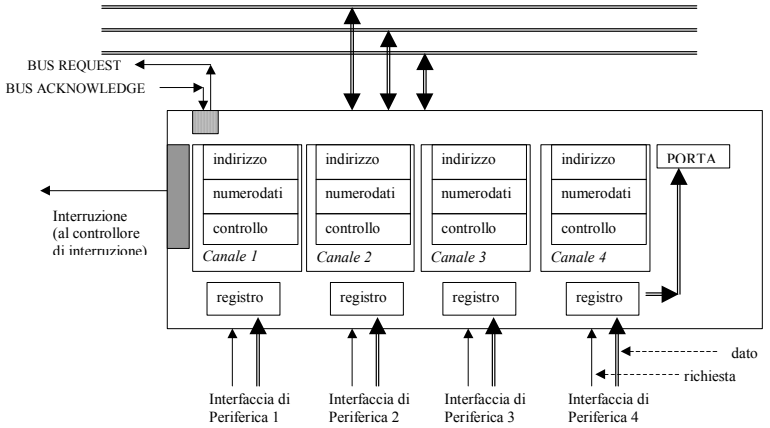
Azioni del programma di risposta a interruzione di fine DMA

- Il programma di risposta all'interruzione di fine DMA trova il buffer ormai pieno della copia del settore voluto.
- Il programma di risposta disabilita l'interfaccia di DMA sia al DMA sia all'interruzione di fine DMA, scrivendo gli appositi bit di controllo nel registro di controllo dell'interfaccia.
- Il programma di risposta chiama la parte di sistema operativo che toglie il programma da quelli in attesa di un evento esterno e lo mette tra quelli pronti per essere eseguiti, e poi torna: il programma di risposta viene concluso. Il programma era certamente in attesa, altrimenti non si sarebbe scatenata la sequenza di DMA.

Osservazione

- Per diminuire ancora il tempo tra due cicli successivi rubati, spesso il DMA viene fatto senza restituire al processore il bus durante tutta la raffica dei trasferimenti di un blocco di dati (DMA Block Transfer).

Un controllore DMA completo



Confronti tra tecniche di I/O

	Tempo minimo tra due trasferimenti successivi	Costo circuitale tipico	Esecutore del trasferimento del singolo dato	Periferiche tipiche
Controllo a programma	Medio	Molto basso	Programma	Periferica che fornisce/accetta il dato quando richiesto dal programma, *sempre e velocemente,* in modo certo e con temporizzazioni note; inoltre, la perdita o la duplicazione di un dato è irrilevante. *Esempio: termometro da forno*
Interruzione	Alto (lento, per la necessità di salvataggio e ripristino dello stato)	Medio (uso di un controllore d'interruzione)	Programma, a esecuzione imposta dall'interfaccia	Periferica che di tanto in tanto, in tempi imprevedibili dai programmi, scambia un dato che non deve essere perso o duplicato, oppure richiede attenzione e reazione. *Esempi: tastiera, sensore di allarme*
DMA	Bassissimo (molto veloce, vicino al limite massimo consentito da memoria e bus)	Alto (uso di un controllore di DMA)	Circuito controllore di DMA, su richiesta dei circuiti di interfaccia della periferica.	Periferica che di tanto in tanto scambia raffiche di dati velocissime; i dati non devono essere persi o duplicati. *Esempio: disco magnetico*