Calcolatori Elettronici
Sistema di Ingresso Uscita

Ing. Gestionale e delle Telecomunicazioni

A.A. 2008/09

Gabriele Cecchetti

Sistema di Ingresso Uscita

- Sommario:
  - Generalità
  - Interfaccia di I/O
  - Controllo di programma
  - Interruzione
  - Gestione di Interruzione nei sistemi M68000 e Pentium
  - Accesso diretto alla memoria (DMA)
  - Arbitraggio del bus
  - Funzionamento del bus
  - Sistema di Ingresso-Uscita
- Riferimenti
  - C. Hamacher, “Introduzione all’architettura del Calcolatore”, cap. 8.

Alcune nozioni fondamentali
Interfaccia di periferica
Gestione di I/O a controllo di programma

GENERALITÀ

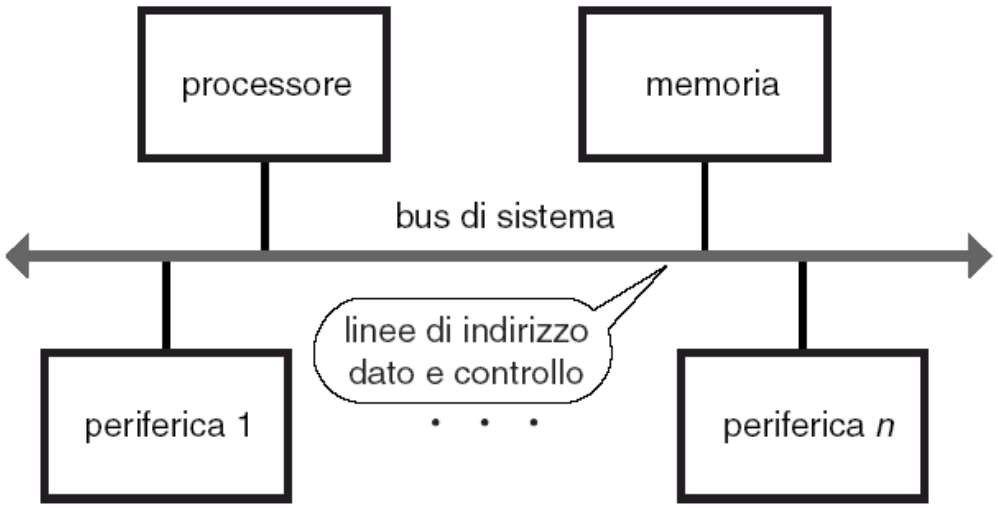
Operazione di I/O

- Le unità di periferica (periferiche), o dispositivi di I/O, servono al calcolatore per scambiare dati con l'ambiente.
- La periferica può essere:
  - letta: il processore ottiene da essa un dato
  - scritta: il processore le invia un dato
- L'operazione di I/O non è sostanzialmente diversa da quella di memoria (lettura e scrittura di parola).
- Si possono scambiare singoli byte (o parole), o blocchi di byte (o di parole), con la periferica.
- L'operazione di scambio di un blocco di byte viene scomposta in una sequenza di operazioni di byte.

Struttura del Bus

- Le unità di periferia del calcolatore sono collegate al processore tramite il bus.
- Il bus è un fascio di collegamenti (fili) e contiene tre gruppi di linee:
  - di indirizzo
  - di dato
  - e di controllo (linee in numero vario)
- Se il bus è unico, viene spesso chiamato *bus di sistema* (*system bus*).

Schema di Collegamento



periferiche collegate al bus di sistema

Funzionamento del Bus

- In linea generale, l'operazione di I/O tra processore e periferica (cioè interfaccia) si svolge nel modo seguente:
  - il processore emette l'indirizzo della periferica da leggere o scrivere tramite il bus indirizzi, e il comando (di lettura o scrittura) tramite il bus di controllo; tutte le periferiche vedono entrambi
  - le periferica interessata rileva l'indirizzo come proprio, si attiva e interpreta il comando, poi:
    - se è lettura, invia il dato al processore tramite il bus dei dati
    - se è scrittura, riceve il dato dal processore tramite il bus dei dati (e si suppone che il processore lo invii, un po' dopo l'indirizzo)
- Questo andamento dell'operazione è generico e minimale; ci possono essere altri segnali di controllo scambiati, per sincronizzare processore e periferica.

Interfaccia di Ingresso-Uscita

- Più precisamente, la periferica non è collegata direttamente al bus, bensì all'interfaccia (o porta) di ingresso-uscita (o di I/O), la quale a sua volta è collegata direttamente al bus del calcolatore.
- Ogni interfaccia è associata a un indirizzo (o a un intervallo di indirizzi consecutivi), che la identificano univocamente (tra tutte le altre interfacce).
- L'interfaccia risponde all'indirizzo inviato dal processore tramite le linee di indirizzo, e scambia dati con il processore tramite le linee di dato.
- Il bus di controllo serve per scambiare comandi e risposte (tra interfaccia e processore), e segnali di sincronizzazione di varia natura.

Schema di Indirizzamento

- Si possono definire gli indirizzi di interfaccia di I/O in due modi differenti.
- Spazio di indirizzamento di I/O unificato con quello di memoria:
  - alcuni indirizzi di memoria sono assegnati alle interfacce (e non sono più usabili come memoria)
- Spazio di indirizzamento di I/O separato da quello di memoria:
  - gli indirizzi di I/O sono indipendenti da quelli di memoria, e in aggiunta a questi
- Per lo più, i calcolatori attuali hanno spazi di indirizzamento di memoria e di I/O unificati.

Schema di Indirizzamento

- Se gli spazi di indirizzamento di I/O e memoria sono unificati, le istruzioni macchina che lavorano in memoria (caricamento e memorizzazione) possono anche leggere e scrivere da e verso le interfacce di I/O:
  - gestione unificata di memoria e I/O
- Altrimenti, il processore deve disporre di istruzioni macchina specifiche per le operazioni di I/O (istruzioni IN e OUT):
  - gestione separata di memoria e I/O

Schema di Indirizzamento e SO

- Se il calcolatore è dotato di sistema operativo con meccanismo di memoria virtuale e paginazione, il SO stesso può gestire una o più pagine come se fossero assegnate a periferiche.
- In tale modo lo spazio di memoria virtuale risulta suddiviso naturalmente in indirizzi memoria vera e propria e in indirizzi di periferica.
- Naturalmente il sistema deve contenere un'unità di gestione della memoria (MMU) capace di realizzare tale funzionalità di virtualizzazione.

Struttura dell'interfaccia
Componenti interni
Esempio di interfaccia di terminale
Tecniche di ingresso-uscita

INTERFACCIA DI I/O

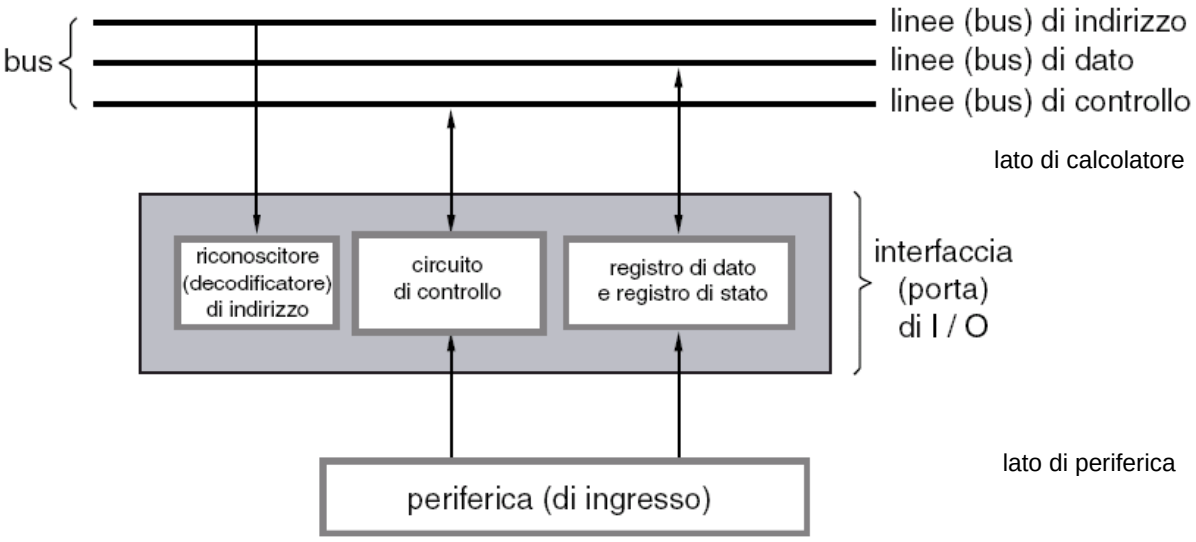
Struttura dell'Interfaccia di I/O

- L'interfaccia di I/O ha struttura varia, ma si possono distinguere quattro parti fondamentali.
- Riconoscitore di indirizzo:
  - esamina l'indirizzo corrente sulle linee di indirizzo e segnala se sia quello assegnato all'interfaccia (o uno di quelli), nel quale caso l'interfaccia si attiva
  - è specifico per l'interfaccia, o quanto meno programmabile
- Registro di stato:
  - contiene alcuni bit che indicano lo stato di funzionamento della periferica:
    - dato pronto per il processore, attesa di nuovo dato da parte del processore, errore, ecc
  - serve per scambiare tra interfaccia e processore varie informazioni di stato e controllo

Struttura dell'Interfaccia di I/O

- Registro di dato:
  - serve per scambiare il dato con il processore:
    - operazione di lettura: la periferica deposita il dato nel registro, da dove il processore lo va a leggere
    - operazione di scrittura: il processore deposita il dato nel registro, da dove la periferica lo va ad acquisire
  - se l'interfaccia ha più indirizzi assegnati, può avere altrettanti registri di dato e stato
- Circuito di controllo:
  - serve per controllare, coordinare e sincronizzare il funzionamento dell'interfaccia
  - secondo l'interfaccia, può essere semplice o molto complesso e ricco di funzionalità

Schema di Interfaccia



componenti fondamentali dell'interfaccia di I/O (qui sola lettura)

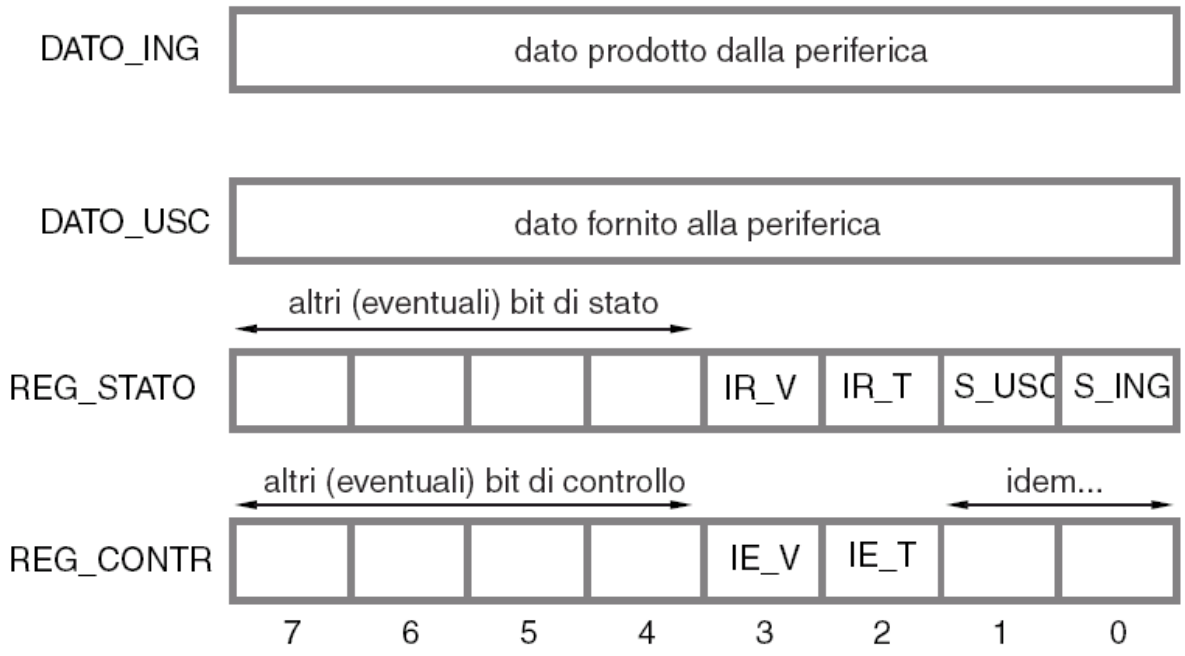
Esempio – Porta di Terminale

- Il terminale è l'insieme di:
  - tastiera: dispositivo di ingresso di caratteri
  - schermo (monitor): dispositivo di uscita
    - a carattere (finestra di terminale)
    - grafico (più o meno sofisticato)
- Qui si considera il terminale con uscita a caratteri, la versione più semplice.
- Ha un'interfaccia di lettura e scrittura:
  - la tastiera manda un carattere (codice ASCII del tasto premuto) al processore
  - il processore manda un carattere (ASCII) allo schermo, da visualizzare alla posizione corrente

Registri di Dato e Stato

- La porta ha quattro indirizzi assegnati:
  - DATO_ING: indirizzo del registro di dato dove la tastiera mette il codice ASCII del tasto premuto, per il processore
  - DATO_USC: indirizzo del registro di dato dove il processore mette il codice ASCII del carattere da visualizzare, per lo schermo
  - REG_STATO: indirizzo del registro di stato tramite cui processore e terminale si scambiano informazioni di funzionamento (dato pronto o no, e simili)
  - REG_CONTR: indirizzo del registro di stato contenente i bit di controllo del meccanismo di interruzione (vedi di seguito)
- I registri di dato qui sono da 8 bit ciascuno (un byte).
- I registri di stato e controllo sono da 8 bit, ma solo alcuni bit di essi sono realmente utilizzati.
- I registri potrebbero essere anche da 16, 32, ecc, bit.

Terminale – Dato e Stato



Terminale – Bit di Stato

- I bit del registro di stato REG_STATO hanno i significati seguenti:
  - S_ING o stato di ingresso:
    - va automaticamente a 1 non appena la tastiera ha depositato nel registro DATO_ING il codice ASCII di tasto premuto
    - va automaticamente a 0 non appena il processore ha letto il contenuto del registro DATO_ING
  - S_USC o stato di uscita:
    - va automaticamente a 1 non appena il processore ha scritto nel registro DATO_USC il carattere da visualizzare
    - va automaticamente a 0 non appena lo schermo ha acquisito il contenuto del registro DATO_USC
- Il circuito di controllo della porta aggiorna in modo automatico i due bit S_ING e S_USC.
- I bit rimanenti (IR_V e IR_T) servono per l'interruzione.

Bit di Controllo di Interruzione

- Alcuni bit (nel registro di stato e di controllo) servono per gestire ordinatamente il meccanismo di interruzione:
  - IR_T (interrupt request di tastiera), va automaticamente a 1 quando la tastiere chiede interruzione (a seguito di pressione di tasto), e in seguito viene riportato a 0 dal processore
  - IR_V, idem, ma vale per lo schermo
  - IE_T (interrupt enable di tastiera), se il processore lo pone a 1 la tastiera ha il permesso di chiedere interruzione, se lo pone a 0 non ha il permesso (serve per ammettere o inibire l'interruzione)
  - IE_V, idem, ma vale per lo schermo
- Questi bit sono usati, in modo vario, quando il terminale è gestito in modo di interruzione (vedi di seguito).

Tecniche di Gestione di I/O

- Controllo di programma:
  - l'operazione di lettura o scrittura di periferica è interamente gestita dal processore
  - è una tecnica puramente SW
- Interruzione (o interrupt):
  - parte dell'operazione è assegnata direttamente all'interfaccia
  - è una tecnica mista HW/SW
- Accesso diretto alla memoria (direct memory access o DMA):
  - quasi tutta l'operazione è svolta in HW (tranne la fase iniziale)
  - necessita di un apposito controllore di DMA, un dispositivo HW specializzato (integrato nell'interfaccia, o nel processore, o separato)

Concetto di controllo di programma

Ciclo di controllo

CONTROLLO DI PROGRAMMA

Controllo di Programma

- Il programma esamina (si dice “monitora”) periodicamente o con continuità l’interfaccia, controllandone i bit di stato.
- Se le condizioni indicate dai bit di stato sono opportune, legge o scrive i registri di dato, secondo quanto occorre fare.
- Eventualmente aggiorna i bit di stato (e di controllo), secondo l’esito dell’operazione.
- Il programma è pertanto ciclico e non ha altro da fare (o quasi) se non gestire la periferica.

Controllo di Terminale

- Il programma esamina ciclicamente (monitora) lo stato della tastiera.
- Se questa ha pronto un codice di tasto, il programma agisce come segue:
  - lo legge e deposita in un tampone (buffer) di memoria predisposto allo scopo
  - lo scrive nel registro di dato dello schermo affinché venga visualizzato
  - verifica se il carattere sia di fine linea, e se sì termina invocando una routine di elaborazione della linea
  - altrimenti, torna a monitorare la tastiera …
- Con qualche dettaglio aggiunto, vedi la codifica.

Programma di Controllo

	move	#LINEA, R_0	prepara il puntatore al buffer
ATT_TAS	test bit	#0, REG_STATO	esamina il bit S_ING
	branch = 0	ATT_TAS	attendi la pressione del tasto
	move byte	DATO_ING, R_1	leggi da tastiera il carattere
ATT_VID	test bit	#1, REG_STATO	esamina il bit S_USC
	branch = 0	ATT_VID	attendi che il video sia pronto
	move byte	R_1, DATO_USC	scrivi a video il carattere
	move byte	R_1, (R_0) +	scrivi il carattere nel buffer
	compare	#0x0D, R_1	controlla se sia *riporta cursore*
	branch ≠ 0	ATT_TAS	se no salta ad ATT_TAS
	move byte	#0x0A, DATO_USC	se sì scrivi a video *scendi linea*
	call	ELAB_LINEA	routine per elaborare la linea

Istruzione Macchina di TEST

- L'istruzione macchina seguente:
  - TEST BIT pos_bit, sorgente

  esamina il bit, nella posizione specificata dall'operando "pos_bit" (di solito una costante), della parola (byte, ecc) contenuta nell'operando "sorgente" (di solito un registro di processore o una parola di memoria indirizzata in modo assoluto, cioè dandone direttamente l'indirizzo).
- Secondo il valore del bit da testare (0 o 1), il bit di esito Z del processore viene aggiornato (a 0 o 1).
- L'istruzione di salto condizionato successiva esamina il bit di esito Z e si comporta di conseguenza.
- In conclusione, l'istruzione TEST BIT serve per esaminare selettivamente singoli bit in una parola.

Vantaggi e Svantaggi

- Vantaggi del controllo di programma:
  - semplice da realizzare e collaudare
  - economico (solo SW, niente HW aggiuntivo)
- Svantaggi del controllo di programma:
  - impegna costantemente o quasi il processore
  - non concede tempo ad altre attività
  - presuppone che lo stato della periferica sia sempre noto e ben rappresentato nella porta, e che non muti tanto rapidamente da non essere monitorabile periodicamente o con continuità
  - per ogni dato scambiato esegue diverse istruzioni macchina e pertanto è un metodo di gestione relativamente lento
- È adatto a un insieme di periferiche semplici da gestire, che abbiano pochi dati da scambiate e caratterizzate da comportamento prevedibile e regolare.

Concetto di interruzione
Richiesta – identificazione - salto a routine di servizio - priorità

INTERRUZIONE

Interruzione (1/2)

- La tecnica di interruzione prevede che:
  - la periferica possa richiedere autonomamente un servizio (di ingresso-uscita) al processore
  - il processore reagisca sospendendo l'attività corrente e passi a eseguire il servizio richiesto dalla periferica
  - al termine del servizio, il processore riprenda la sua attività precedentemente interrotta
- La periferica (o piuttosto la sua interfaccia) richiede il servizio mandando al processore un segnale di interruzione (interrupt request).

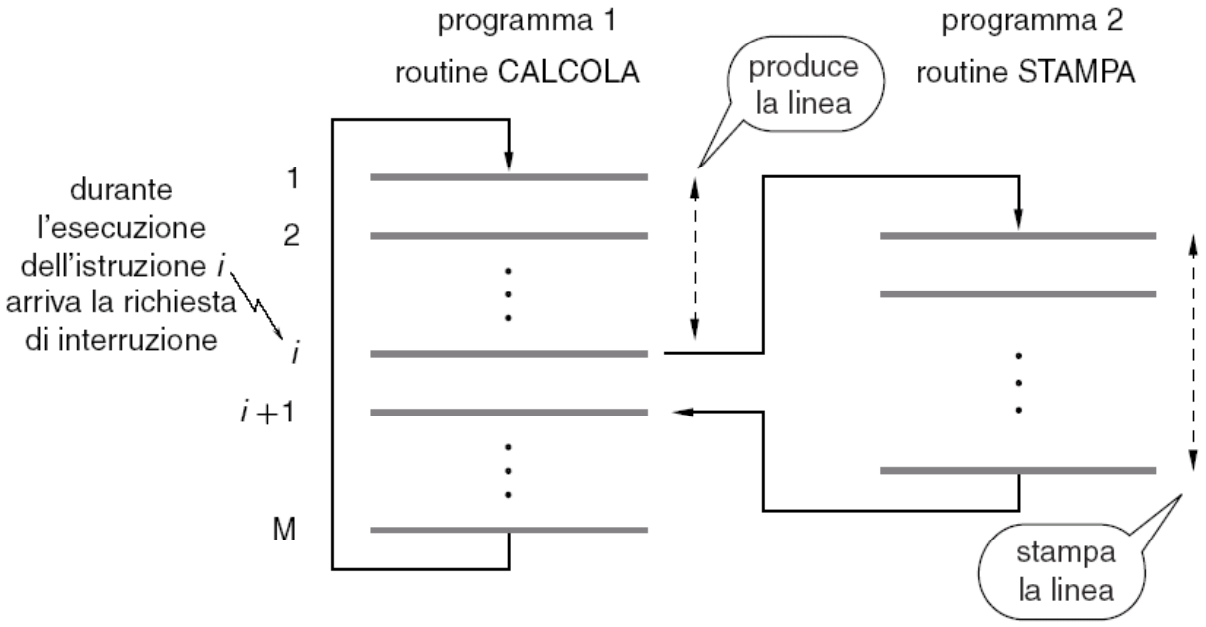
Interruzione (2/2)

- Il servizio richiesto dalla periferica tramite interruzione si chiama routine di servizio di interruzione (interrupt service routine).
- La routine di servizio di interruzione somiglia a una routine ordinaria (procedura o funzione), ma il meccanismo di attivazione è del tutto diverso.
- La routine di servizio può essere chiamata e attivata in un punto qualunque del programma corrente, in modo imprevedibile e senza preavviso.
- Invece, la routine ordinaria viene chiamata e attivata da istruzioni macchina apposite (chiamata a routine o call), collocate in punti definiti del programma e noti a priori.

Esempio - Stampa

- Il programma ciclico CALCOLA (esso stesso una routine, di tipo ordinario) produce una serie di linee di testo (caratteri) da stampare.
- Ogniqualvolta la stampante ha finito di emettere la linea corrente, manda al processore una richiesta di interruzione per avere nuovi dati.
- La richiesta chiama e attiva la routine di servizio di interruzione STAMPA, la quale trasferisce la nuova riga di caratteri alla stampante, e poi termina.
- Terminata la routine STAMPA, la routine CALCOLA riprende da dove si era sospesa.

Interruzione e Servizio



relazione tra programma corrente e routine di servizio

Particolarità

- L'arrivo del segnale di richiesta di interruzione è imprevedibile:
  - l'istruzione macchina raggiunta dal segnale di richiesta viene comunque portata a termine
  - subito dopo, l'esecuzione viene passata alla routine di servizio, che svolge le sue attività di comunicazione con la periferica
  - quando la routine di servizio termina, il programma interrotto riprende dall'istruzione macchina consecutiva a quella che era stata interessata dall'interruzione (già portata a termine prima)
- Nota bene: l'interruzione vera e propria è il segnale di richiesta; il trasferimento di caratteri alla stampante avviene tramite una porta di ingresso-uscita (nell'esempio di sola uscita) del tutto ordinaria.

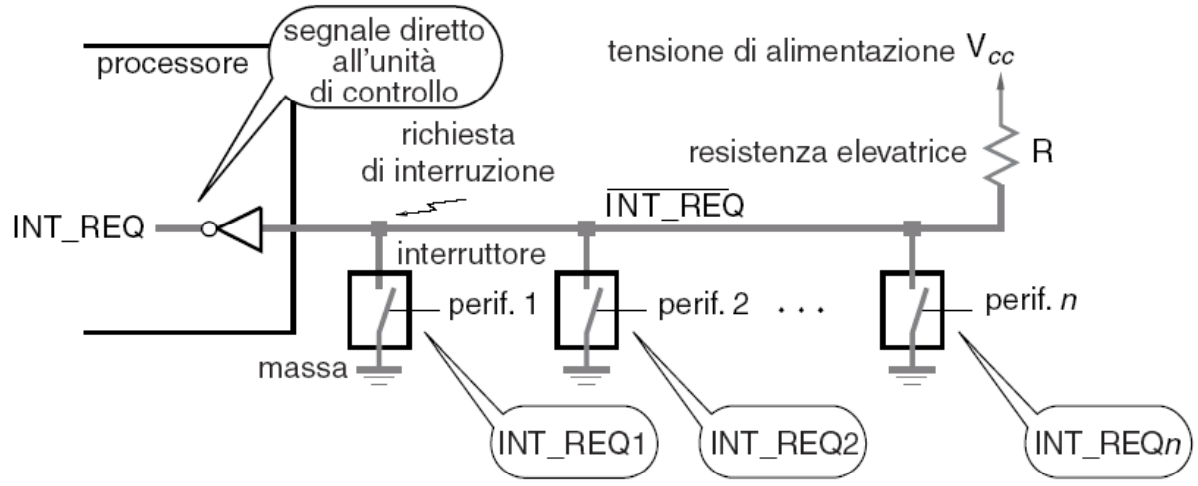
Rientro da Interruzione

- Per gestire il meccanismo di rientro al programma corrente interrotto, si usa la pila (*stack*), come nel caso di routine ordinaria.
- Quando l'esecuzione passa alla routine di servizio di interruzione, l'indirizzo di rientro al programma corrente (nell'esempio l'indirizzo $i + 1$), viene impilato.
- La routine di servizio termina con un'istruzione macchina apposita, chiamata RETI (return from interrupt) o similmente, la quale spila l'indirizzo e lo rimette nel contatore di programma.

Segnale di Interruzione

- Il bus di controllo del calcolatore contiene un segnale di richiesta di interruzione:
  - interrupt request (INT_REQ)
- Tutte le periferiche sono collegate alla linea di INT_REQ e possono attivarla (per esempio portandola a 0) per richiedere il servizio.
- Tale meccanismo di segnalazione funziona correttamente se c'è una sola richiesta per volta.
- Di seguito si mostrerà come rilassare tale vincolo, di per sé troppo restrittivo.

Segnale di Interruzione



meccanismo di richiesta di interruzione (versione semplice)

Problematiche di Interruzione

- In generale, al calcolatore sono collegate due o più periferiche, e tutte o molte potrebbero servirsi del meccanismo di interruzione.
- Inoltre, una stessa periferica potrebbe avere bisogno di due o più routine di servizio, con compiti differenziati secondo la richiesta.
- Infine, le periferiche sono indipendenti e in linea di principio non sincronizzate, e potrebbero richiedere interruzione simultaneamente o a distanza di tempo molto ravvicinata.

Quattro Domande

- Si può abilitare e disabilitare il meccanismo di interruzione e rendere il programma non interrompibile ?
- È ammissibile che una richiesta di interruzione interrompa una routine di interruzione già in corso (problema di servizio annidato) ?
- Come fa il processore a distinguere tra richieste provenienti da periferiche differenti (o anche dalla stessa periferica, ma per motivi diversi) e ad attivare la routine di servizio corrispondente ?
- Che cosa succede se due (o più) richieste di interruzione vengono mandate simultaneamente o molto ravvicinate da due (o più) periferiche ?

Abilitazione e Disabilitazione

- Si può abilitare e disabilitare il meccanismo di interruzione, agendo dal lato di processore oppure di periferica.
- Lato di processore:
  - un bit di stato (nel registro di stato SR) permette di rendere il processore sensibile o insensibile alla richiesta (se è in stato di insensibilità, il processore la ignora)
  - talvolta ci sono istruzioni macchina apposite per modificare tale bit: DI ed EI (disable / enable interrupt), o con nomi simili
- Lato di periferica:
  - su può premettere o vietare alla periferica l'uso del meccanismo di interruzione, mediante bit appositi situati nel registro di controllo dell'interfaccia di I/O della periferica stessa
  - per esempio, vedi i bit IE_T e IE_V dell'interfaccia di terminale, nel registro di controllo REG_CONTR (esempio precedente)
  - tali bit vanno modificati da parte del processore, con istruzioni macchina di ingresso-uscita scrivendo nel registro di controllo

Annidamento di Servizio

- Usando la pila come struttura dati per gestire gli indirizzi di rientro da routine di servizio, si possono avere servizi annidati.
- Spetta al programmatore decidere se una routine di servizio vada eseguita con interruzione disabilitata (dal lato di processore) oppure abilitata.
- Se la routine di servizio non deve essere interrompibile, basta disabilitare il meccanismo di interruzione all'inizio della routine (con un'istruzione DI) e riabilitarlo subito prima del rientro (con un'istruzione EI).
- Spesso, il meccanismo di interruzione viene disabilitato in modo automatico non appena parte la routine (ciò è attinente soprattutto al rapporto con il sistema operativo)

Identificazione della Periferica (1/2)

- Di solito il calcolatore deve erogare uno tra diversi servizi di interruzione possibili, secondo quale sia la periferica che ha richiesto l'interruzione e che va servita.
- Ciascun servizio può essere svolto da una routine di servizio differente.
- Oppure più servizi possono essere raggruppati in una sola routine, che ne esegue in modo selettivo uno solo a ogni attivazione.

Identificazione della Periferica (2/2)

- Per decidere quale servizio svolgere a seguito della richiesta, ci sono tre metodi fondamentali:
  - una sola linea INT_REQ e salto a indirizzo di memoria fisso di routine di servizio (la quale è unica)
  - due o più linee INT_REQ distinte, ciascuna associata a una specifica routine di servizio collocata a un indirizzo di memoria specifico
  - una sola linea INT_REQ e meccanismo di interruzione vettorizzata (*vectored interrupt*)
- Tali metodi non sono sempre esclusivi, e non pochi processori li possono usare tutti e tre.

Salto a Indirizzo Fisso

- Accettando la richiesta di interruzione, il processore salta a un indirizzo di memoria prefissato, dove inizia la routine di servizio (che è unica).
- La routine scandisce (polling) le porte di I/O di tutte le periferiche (o di un gruppo), esaminando i bit di stato che segnalano quale di esse abbia fatto la richiesta.
- Nota bene: si suppone che la periferica mandi la richiesta sulla linea INT_REQ e simultaneamente attivi un bit apposito nel registro di stato della sua interfaccia.
- Identificata la periferica interrompente, la routine ne disattiva il bit di stato, esegue il servizio specifico per la periferica in questione e poi termina.
- Nell'esempio di prima (terminale), tali bit di stato sono IR_T e IR_V (per tastiera e schermo), in REG_STATO.

Linee Multiple di Richiesta

- Ciascuna periferica dispone di una sua linea di richiesta INT_REQ specifica ed eventualmente di una linea di conferma INT_ACK.
- Ogni linea è associata a un indirizzo di memoria prefissato di salto, dove si trova la routine di servizio corrispondente.
- Naturalmente il numero di periferiche è limitato dal numero di linee, che di solito è relativamente modesto, ≤ 8 o circa.

Interruzione Vettorizzata

- A ogni periferica è associato un codice identificativo univoco (di solito di pochi bit, ≤ 8).
- La periferica manda la richiesta su INT_REQ.
- All'arrivo della conferma su INT_ACK (nota che la richiesta potrebbe essere rimasta pendente a lungo), la periferica manda al processore (via bus dei dati) il codice di identificazione.
- Il processore riceve il codice e lo usa per derivare l'indirizzo di memoria effettivo dove è collocata la routine di servizio corrispondente.

Vettore di Interruzione

- Il metodo più comune per derivare l'indirizzo effettivo di routine di servizio a partire dal codice di identificazione di periferica, è il seguente:
  - gli indirizzi iniziali di tutte le routine di servizio sono elencati in una tabella (un array) di indirizzi (di fatto, sono numeri interi)
  - il codice identificativo funziona come indice della tabella, e punta all'elemento contenente l'indirizzo di routine da usare.
- Gli indirizzi di routine si chiamano vettori di interruzione.
- Naturalmente la tabella dei vettori sta in una regione di memoria a ciò designata (di solito essa parte proprio dalla cella iniziale di memoria), e va inizializzata correttamente (di solito se ne occupa il S.O.).
- Insieme al vettore di interruzione, in certi casi può anche essere associata una parola di stato.

Confronti

- Saltare a indirizzo fisso ha costo limitato, ma scandire le periferiche può dare luogo a ritardi.
- Associare un indirizzo a ogni linea di richiesta è più flessibile, ma occorrono due o più linee e tale numero non può essere molto elevato.
- Il meccanismo di vettorizzazione è il metodo più raffinato e costoso, ma in definitiva non troppo, ed è una tecnica versatile e facilmente riconfigurabile anche in corso di esecuzione.
- Di fatto il metodo di vettorizzazione è molto diffuso, tranne che in processori assai semplici.

Schema di Priorità

- Quando ci sono numerose periferiche collegate in interruzione, per gestire in modo ordinato i permessi di interruzione annidati e la simultaneità di richiesta occorre prevedere uno schema di priorità.
- Secondo lo schema di priorità, il processore ascolta e serve determinate richieste di interruzione con preferenza rispetto ad altre.
- Ci sono varie soluzioni, secondo il numero di linee di controllo da aggiungere al bus.

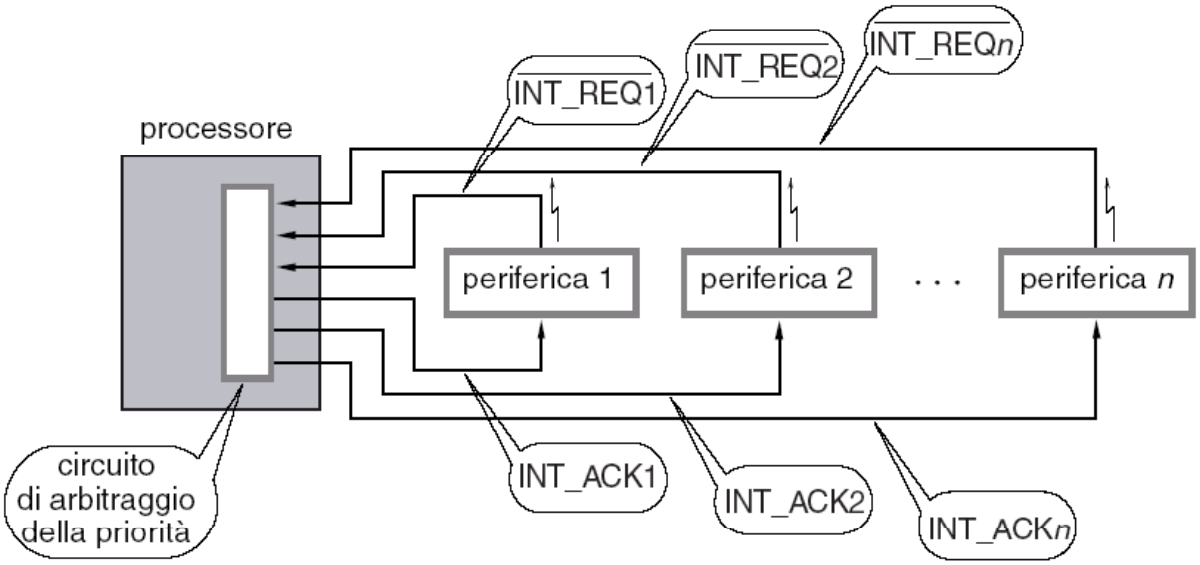
Conferma di Interruzione

- Per gestire la priorità, occorre comunque un meccanismo di interruzione basato su una coppia di segnali (linee nel bus di controllo):
  - INT_REQ, per richiedere il servizio (come si fa senza priorità)
  - INT_ACK (interrupt acknowledge o conferma di interruzione, emessa dal processore), per dare conferma dell'avvenuto inizio del servizio
- Nell'intervallo di tempo tra l'invio della richiesta di interruzione e l'arrivo della conferma (che potrebbe tardare anche a lungo), si dice che l'interruzione è pendente (pending interrupt).

Controllore di Interruzione

- Ogni periferica ha una coppia di fili (linee):
  - INT_REQ, per richiedere il servizio di interruzione
  - INT_ACK, per confermare l'inizio del servizio
- Richieste e conferme confluiscono nel circuito di arbitraggio di priorità (o controllore di interruzione), che le ordina in priorità secondo uno schema fisso oppure programmabile.
- Il processore programma all'inizio il controllore, il quale in seguito opera in modo autonomo.
- Spesso il controllore è integrato nel processore.

Schema di Priorità - Controllore



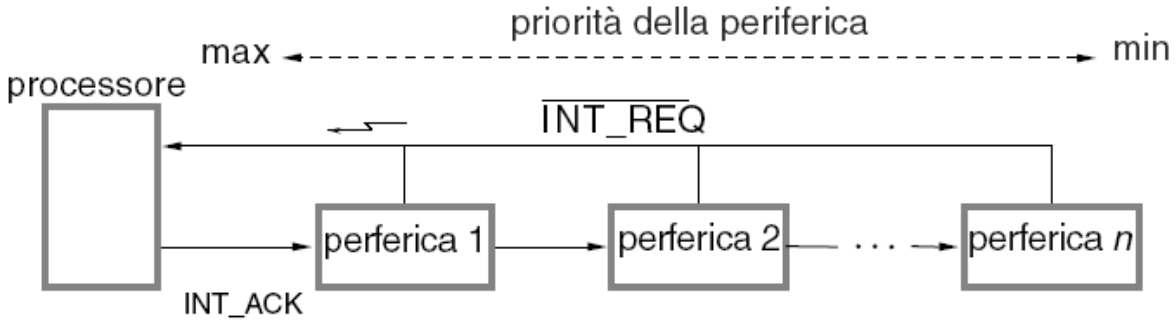
soluzione con coppia di fili INT_REQ e INT_ACK per ciascuna periferica e con circuito di arbitraggio della priorità (controllore di interruzione)

Collegamento a Festone

- Le periferiche sono collegate a festone (*daisy chain*), e l'ordine lineare di collegamento lungo il festone dà luogo alla priorità.
- La linea di richiesta INT_REQ è unica e comune a tutte le periferiche. Ognuna di esse può attivarla.
- La linea di conferma INT_ACK è unica ma attraversa le periferiche, le quali possono decidere se inoltrare la conferma alla periferica successiva o intercettarla e tenerla per sé.
- Di norma il segnale di conferma si propaga lungo la catena di periferiche.
- Quando una periferica richiede, attende la conferma, e quando la riceve la intercetta.

Schema di Priorità - Festone

collegamento a festone (daisy chain)



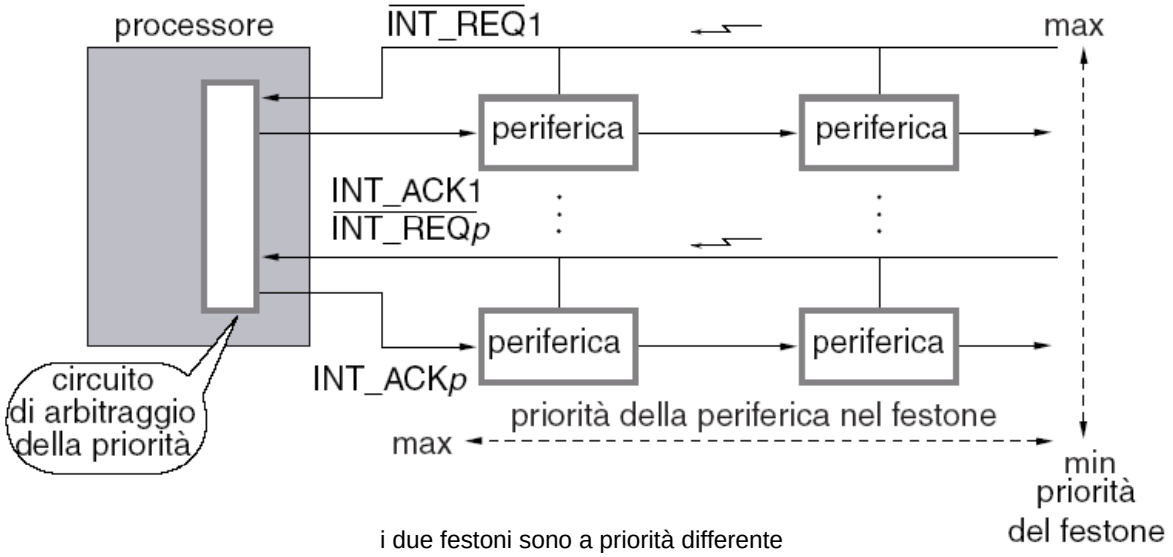
la linea di richiesta (INT_REQ) è unica: il processore emette il segnale
di conferma (INT_ACK) non appena riceve una richiesta, il segnale si propaga lungo la catena di periferiche e la periferica interrompente lo intercetta,
senza passarlo alla periferica successiva

Festone Multiplo

- Il festone singolo dà luogo a uno schema di priorità puramente lineare.
- Per avere flessibilità maggiore si possono prevedere due o più festoni, a priorità differente.
- La priorità della periferica dipende in primo luogo dal festone cui essa è collegata, e in secondo luogo dalla posizione che essa occupa nel festone.
- Si può collegare una stessa periferica a più festoni, per darle modo di mandare richieste differenziate secondo il livello di priorità.

Priorità - Festone Multiplo

collegamento a festone multiplo



i due festoni sono a priorità differente
lo schema di priorità complessivo e più complesso di quello lineare

Priorità in Software

- Il processore mantiene (nel registro di stato) un livello di priorità, cioè un numero (per esempio da 0 a 7, estremi inclusi).
- Ogni richiesta di interruzione è associata a un livello di priorità ben definito (due o più richieste possono avere lo stesso livello).
- La richiesta di interruzione viene accettata e servita da parte del processore se e solo se il livello di priorità della richiesta è maggiore o uguale a quello corrente del processore.
- Quando il processore accetta la richiesta di interruzione, il suo livello di priorità si equalizza a quello della richiesta accettata.
- Quando la routine di servizio di interruzione termina, il livello di priorità del processore ritorna al valore originale, cioè a quello che valeva subito prima di accettare la richiesta.
- Il livello di priorità della richiesta è registrato nella tabella di interruzione insieme al vettore di interruzione corrispondente (il livello è un campo di bit contenuto nella parola di stato).
- Per salvare e ripristinare il livello di priorità corrente, il processore si serve della pila, esattamente come fa per l'indirizzo di rientro.

Esempio di Interruzione

- Esempio con tastiera gestita in interruzione (con uno qualsiasi dei tre metodi di identificazione di periferica).
- Il programma principale prepara in memoria un tampone (buffer) per i caratteri (tasti) in arrivo dalla tastiera, predispone il meccanismo di interruzione e lo attiva.
- Ogniqualvolta viene premuto un tasto, la tastiera ne registra il codice ASCII nel registro di dato della sua interfaccia di I/O e richiede l'interruzione.
- La routine di interruzione legge dalla porta di tastiera il codice ASCII del tasto premuto e lo scrive nel tampone di memoria predisposto dal processore.
- Quando viene premuto il tasto di fine linea, la routine disabilita il meccanismo di interruzione dal lato di tastiera (bit IE_T nel registro di controllo dell'interfaccia).

Programma Corrente

Programma principale

move	#LINEA, PUNTA	predisponi il puntatore al buffer di memoria
clear	FINE_LINEA	azzera l'indicatore di fine linea
set bit	#2, REG_CONTR	abilita interruzione dal lato di tastiera (IE_T = 1)
set bit	#9, SR	abilita interruzione dal lato di processore (IE = 1)
. . .		

attiva il meccanismo di interruzione dal lato di processore e di tastiera

Routine di servizio di interruzione da tastiera

routine di servizio	SERVI_T	move block	R_0-R_1, – (SP)	impila i registri R_0 e R_1 per conservarne il contenuto
		move	PUNTA, R_0	carica in R_0 il puntatore al buffer di memoria
		move byte	DATO_ING, R_1	leggi da tastiera il carattere scrivendolo nel registro R_1
		move byte	R_1, (R_0) +	e poi scrivilo nel buffer di memoria
		move	R_0, PUNTA	aggiorna il puntatore al buffer di memoria
		compare byte	#0x0D, R_1	verifica se il carattere preso da tastiera sia di fine linea
		branch ≠ 0	RIENTRA	se no rientra al programma chiamante
		move	#1, FINE_LINEA	se sì attiva l'indicatore di fine linea
		clear bit	#2, REG_CONTR	disabilita interruzione dal lato di tastiera (IE_T = 0)
	RIENTRA	move block	(SP) +, R_0-R_1	spila i registri R_0 e R_1 ripristinandone il contenuto
		return from interrupt		rientra da interruzione al programma chiamante

Vantaggi e Svantaggi

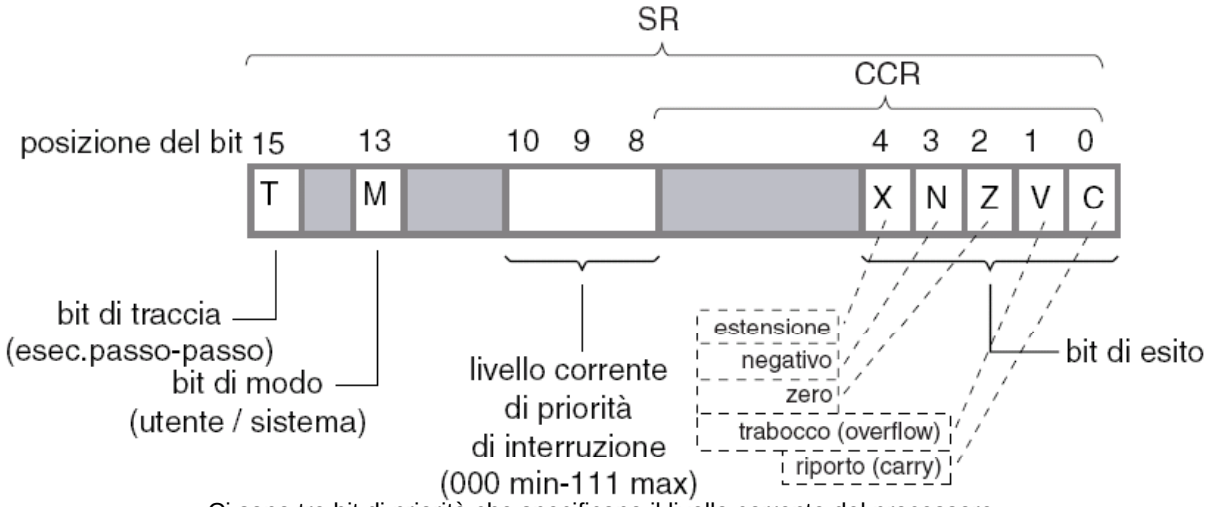
- Vantaggi dell'interruzione:
  - conferisce alla periferica la possibilità di richiedere il servizio solo quando serve realmente
  - non implica, da parte del programmatore, la conoscenza dei tempi di azione caratteristici della periferica
  - lascia libero il processore di dedicarsi ad altro quando le periferiche non hanno bisogno di servizio
- Svantaggi dell'interruzione:
  - ha una certa complessità circuitale, variabile però secondo come è realizzata effettivamente
  - ha una certa complessità organizzativa, specialmente in sede di definizione dei livelli di priorità (se sono previsti, naturalmente)
- È una tecnica adatta per gestire un insieme di periferiche differenziate per modi e tempi di funzionamento, con quantità mutevoli di dati da scambiare e caratterizzate da comportamento aleatorio e imprevedibile.
- In concreto tuttavia è molto usata a rappresenta il metodo principale di interazione con il sistema operativo.

GESTIONE DI INTERRUZIONE NEI PROCESSORI M68000 E PENTIUM

Esempi Concreti

- Di seguito la routine di interruzione generica data prima come esempio è adattata per i processori commerciali Motorola 68000 (a 16 bit di dato e 32 bit di indirizzo) e Intel Architecture a 32 bit (sia per dato sia per indirizzo), o IA-32.
- Entrambi i processori in esame dispongono di meccanismo di interruzione vettorizzata.
- Le istruzioni macchina generiche sono trasposte in istruzioni M68000 e IA-32 specifiche, con gli aggiustamenti del caso per adattamenti vari.
- Per le particolarità si rimanda ai due esempi e agli elenchi di istruzioni macchina (vedi testo).

Registro di Stato SR di M68000



Ci sono tre bit di priorità che specificano il livello corrente del processore. La richiesta di interruzione è accettata se e solo se ha livello di priorità maggiore o uguale a quello corrente del processore, e eleva il livello del processore al proprio (al rientro, il livello originale viene ripristinato). Per capire come associare un livello di priorità alla richiesta, si veda la trattazione specifica del processore M68000.

M68000 - Programma Corrente

Programma principale

MOVE	#LINEA, PUNTA	predisponi il puntatore al buffer di memoria
CLR	FINE_LINEA	azzera l'indicatore di fine linea
ORI.B	#4, REG_CONTR	abilita interruzione dal lato di tastiera (IE_T = 1)
MOVE	#0x100, SR	abilita interruzione dal lato di processore (IE = 1)
...		

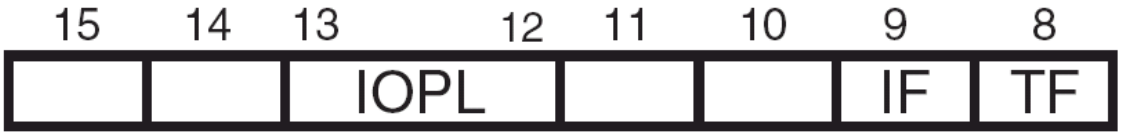
predispone il meccanismo di interruzione e lo attiva

. . .

Routine di servizio di interruzione da tastiera

SERVI_T:	MOVEM.L	A0 / D0, $-(A7)$	impila i registri A0 e D0 per salvarne il contenuto
	MOVEA.L	PUNTA, A0	carica in A0 il puntatore al buffer di memoria
	MOVE.B	DATO_ING, D0	leggi da tastiera il carattere scrivendolo nel registro D0
	MOVE.B	D0, $(A0)+$	e poi scrivilo nel buffer di memoria
	MOVE.L	A0, PUNTA	aggiorna il puntatore al buffer di memoria
	CMPI.B	#0x0D, D0	verifica se il carattere preso da tastiera sia di fine linea
	BNE	RIENTRA	se no rientra al programma chiamante
	MOVE	#1, FINE_LINEA	se sì attiva l'indicatore di fine linea
	ANDI.B	#0xFB, REG_CONTR	disabilita interruzione dal lato di tastiera (IE_T = 0)
RIENTRA:	MOVEM.L	$(A7)+$, A0 / D0	spila i registri D0 e A0 ripristinandone il contenuto
	RTE		rientra da interruzione al programma chiamante

Registro di Stato IA-32



La struttura del registro di stato IA-32 è piuttosto complessa (qui se ne illustra solo una parte, e i bit di esito non sono mostrati), e ci sono ben quattro modi di utente / sistema (selezionabili tramite i due bit IOPL), che impattano in modo diverso sulla gestione di interruzione. Per lo scopo presente basta sapere che in almeno uno dei quattro modi il bit IF (interrupt flag) permette di abilitare o disabilitare il meccanismo di interruzione dal lato di processore.

IA-32 – Programma Corrente

Programma principale

```
...             ...                 prepara puntatore buffer
                                    memoria (istr. omesse)
MOV             $0, FINE_LINEA      azzera l'indicatore di fine
                                    linea
MOV             $4, %BL             prepara la costante 4 nel
                                    registro BL
OR              %BL, REG_CONTR      abilita interruzione dal lato
                                    di tastiera (IE_T = 1)
STI                                 abilita interruzione dal lato
                                    di processore (IF = 1)
...
```

predispone il meccanismo di interruzione e lo attiva

Routine di servizio di interruzione da tastiera

```
SERVI_T:  PUSH    %EAX              impila il registro EAX per
                                    salvarne il contenuto
          PUSH    %EBX              impila il registro EBX per
                                    salvarne il contenuto
          MOV     PUNTA, %EAX       carica in EAX il puntatore al
                                    buffer di memoria
          MOV     DATO_ING, %BL     leggi da tastiera il carattere
                                    scrivendolo nel registro BL
          MOV     %BL, (%EAX)       e poi scrivilo nel buffer di
                                    memoria
          INC     (%EAX)            aggiorna il puntatore al buf-
                                    fer di memoria
          CMP     $0x0D, %BL        verifica se il carattere preso
                                    da tastiera sia di fine linea
          JNE     RIENTRA           se no rientra al programma
                                    chiamante
          MOV     $4, %BL           se sì prepara la costante 4
                                    nel registro BL
          XOR     %BL, REG_CONTR    disabilita interruzione dal
                                    lato di tastiera (IE_T = 0)
          MOV     $1, FINE_LINEA    attiva l'indicatore di fine
                                    linea
RIENTRA:  POP     %EBX              spila il registro EBX ripristi-
                                    nandone il contenuto
          POP     %EAX              spila il registro EAX ripristi-
                                    nandone il contenuto
          IRET                      rientra da interruzione al
                                    programma chiamante
```

Concetto di DMA

Controllore di DMA

ACCESSO DIRETTO ALLA MEMORIA (DMA)

Accesso Diretto a Memoria

- L'accesso diretto a memoria (direct memory access o DMA), è una tecnica hardware specifica per periferiche capaci di trasferire blocchi di dati di grande dimensione, a velocità elevata e con frequenza.
- Per esempio, è usata spesso da dischi (ad alta velocità), scheda di rete e altre periferiche funzionanti a velocità elevata.
- Si basa sull'uso di dispostivi ad hoc e di segnali di controllo specifici nel bus del calcolatore.

Funzionamento (1/2)

- La periferica manda una richiesta di trasferimento (in lettura o scrittura) a un dispositivo specifico, chiamato controllore di DMA (o DMA controller).
- Il controllore di DMA inoltra la richiesta al processore, tramite una linea apposita del bus del calcolatore (chiamata in genere BUS_REQ, bus request).
- Il processore recepisce la richiesta e sospende subito l'esecuzione lasciando libero il bus del calcolatore.
- Il controllore di DMA acquisisce il controllo del bus e gestisce il trasferimento dal blocco di dati tra la periferica e un tampone (buffer) di memoria designato allo scopo, secondo la periferica interessata dall'operazione.

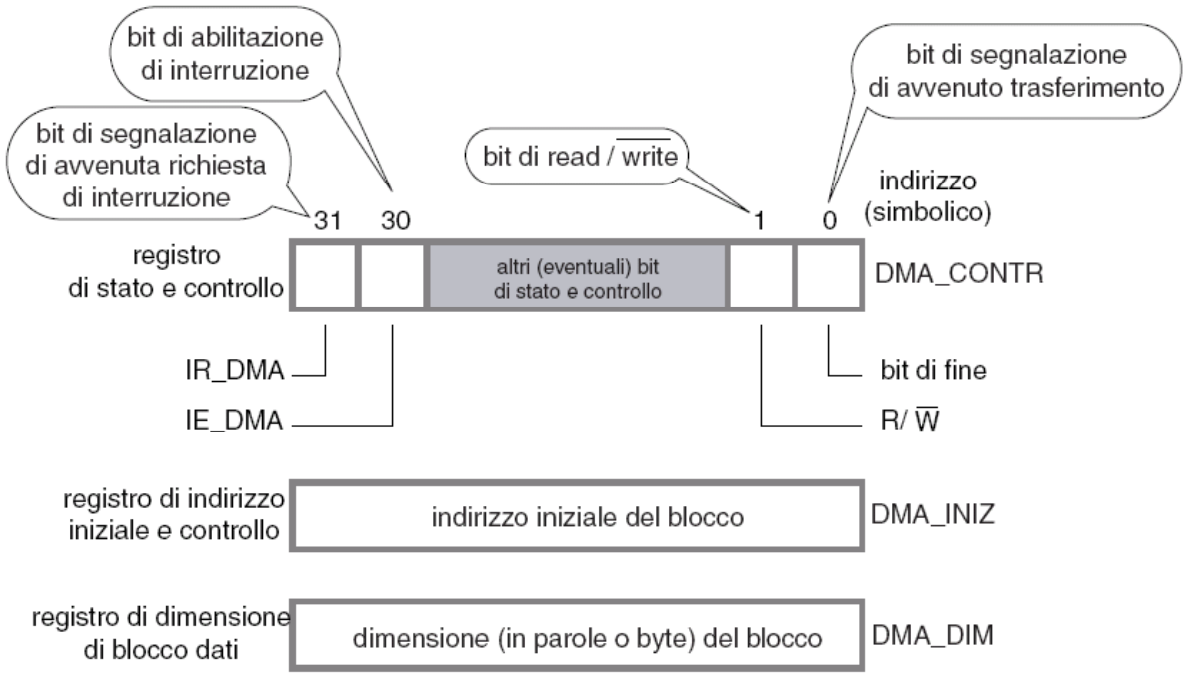
Funzionamento (2/2)

- Non appena il blocco è stato trasferito, il controllore di DMA restituisce il controllo del bus e il processore, rimasto fermo per tutta la durata del trasferimento del blocco di dati, riprende l'attività sospesa.
- Per il processore la sospensione è del tutto trasparente e passa inosservata. Esso si può rendere conto del trasferimento avvenuto esaminando lo stato del controllore, cioè leggendone il registro di stato.
- Spesso però il controllore di DMA segnala che un blocco è stato trasferito mandando al processore una richiesta di interruzione, non appena il processore è ripartito.

Controllore di DMA

- Onde gestire il trasferimento del blocco, il controllore di DMA contiene alcuni registri specifici:
  - indirizzo iniziale del tampone di memoria
  - dimensione del tampone, ovvero del blocco di dati da trasferire (di solito la dimensione è espressa in byte)
  - registro di controllo, contenente vari bit:
    - R / W, per specificare se l'operazione sia di lettura o scrittura
    - bit di fine, si attiva quando il blocco è interamente trasferito
    - IR_DMA, si attiva quando il controllore richiede interruzione
    - IE_DMA, serve per abilitare o disabilitare il meccanismo di interruzione (che è facoltativo) del lato di controllore
- Il processore deve inizializzare registri e stato del controllore di DMA all'inizio, poi il controllore funziona in modo autonomo (tranne quando va riconfigurato).

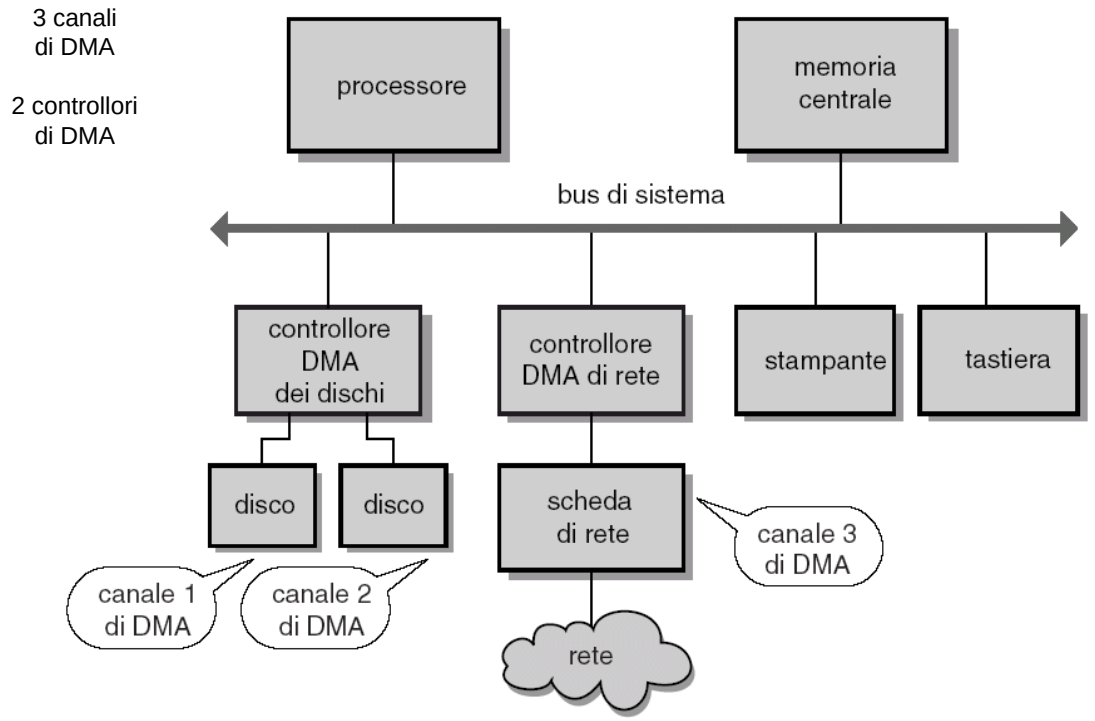
Controllore di DMA



Controllore di DMA

- Per il funzionamento del segnale di controllo BUS_REQ, si veda come funziona l'arbitraggio del bus.
- Il controllore di DMA può gestire più periferiche, e per ciascuna di esse dispone di un gruppo di registri, programmato opportunamente (di solito dal SO).
- Ogni gruppo di registri va sotto nome di canale di DMA, ed è legato a una periferica specifica.
- Il controllore di DMA può essere un dispositivo fisicamente separato dal processore, ma spesso è realizzato sullo stesso componente integrato.
- Il calcolatore può disporre di un certo numero di canali di DMA, dedicati alle periferiche capaci e veloci.

Canali di DMA



Vantaggi e Svantaggi

- Vantaggi del DMA:
  - gestisce in modo efficiente le periferiche capaci e veloci
  - interferisce con l'attività normale del processore (tenendolo fermo) solo per il tempo minimo strettamente necessario per trasferire il blocco di dati
- Svantaggi del DMA:
  - ha una notevole complessità circuitale, dovuta al controllore
- Il DMA è comunque una tecnica molto usata per le periferiche capaci (molti dati) e veloci.
- Dato che spesso il trasferimento avvenuto viene segnalato al processore tramite interruzione, il DMA si presenta più come un raffinamento della tecnica di interruzione che come un metodo a sé stante.

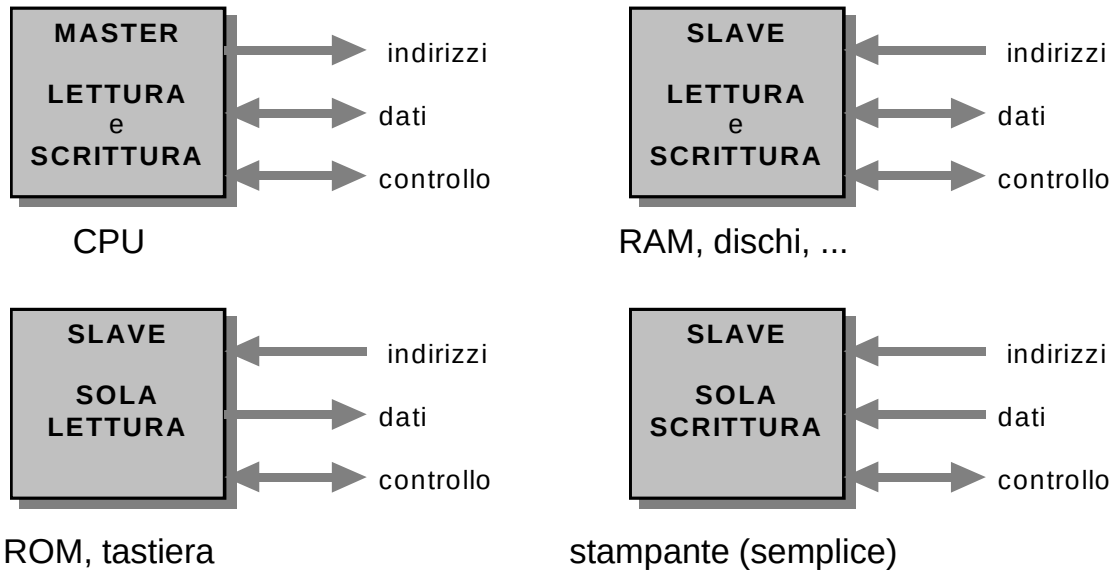
Arbitraggio centralizzato e distribuito

ARBITRAGGIO DEL BUS

Ruolo di Master e Slave

- In ogni istante una sola unità funzionale possiede il controllo del bus, cioè decide quali operazioni di trasferimento eseguire.
- Generalmente la CPU possiede il controllo del bus (o dei bus, se ce n'è più d'uno), ma può anche cedere temporaneamente questo ruolo ad altre unità funzionali.
- L'unità funzionale che detiene il controllo del bus si chiama MASTER (o unità principale):
  - decide quale operazione vada eseguita, lettura oppure scrittura, e in quale istante di tempo
  - decide quale sia l'unità funzionale da dove leggere oppure dove scrivere
- Le unità funzionali rimanenti, che non detengono il controllo del bus, si chiamano SLAVE (o unità secondarie).

Connessione al Bus



Il ruolo di master è univocamente associato all'autorizzazione a emettere l'indirizzo e il comando di lettura o scrittura

Unità Funzionale e Ruolo

MASTER	SLAVE	ESEMPIO
CPU	memoria	prelievo istruzioni e lettura/scrittura dati
CPU	unità di I/O	ricezione/invio dati da/a un'unità di I/O
CPU	coprocessore	la CPU dà istruzioni al coprocessore
I/O	memoria	accesso diretto alla memoria (DMA)
coprocessore	CPU	il coprocessore legge operandi dalla CPU

alcune unità funzionali hanno ruolo esclusivo di master o slave,
altre posso cambiare ruolo secondo l'attività da svolgere

Arbitraggio del Bus

- Normalmente il processore ha il ruolo di MASTER tra le varie unità funzionali, le quali sono in ruolo di SLAVE.
- Tuttavia in determinate circostanze anche altre unità funzionali possono assumere a tempo determinato il ruolo di MASTER, per scopi particolari:
  - unità di I/O: possono diventare MASTER per trasferire dati direttamente con la memoria, senza bisogno della CPU (come per esempio deve fare il controllore di DMA)
  - co-processore: può diventare MASTER per prelevare operandi dalla memoria (coadiuva il processore nell'eseguire alcune istruzioni molto specializzate, come quelli in virgola mobile)
- Di seguito si vedrà il caso del controllore di DMA.

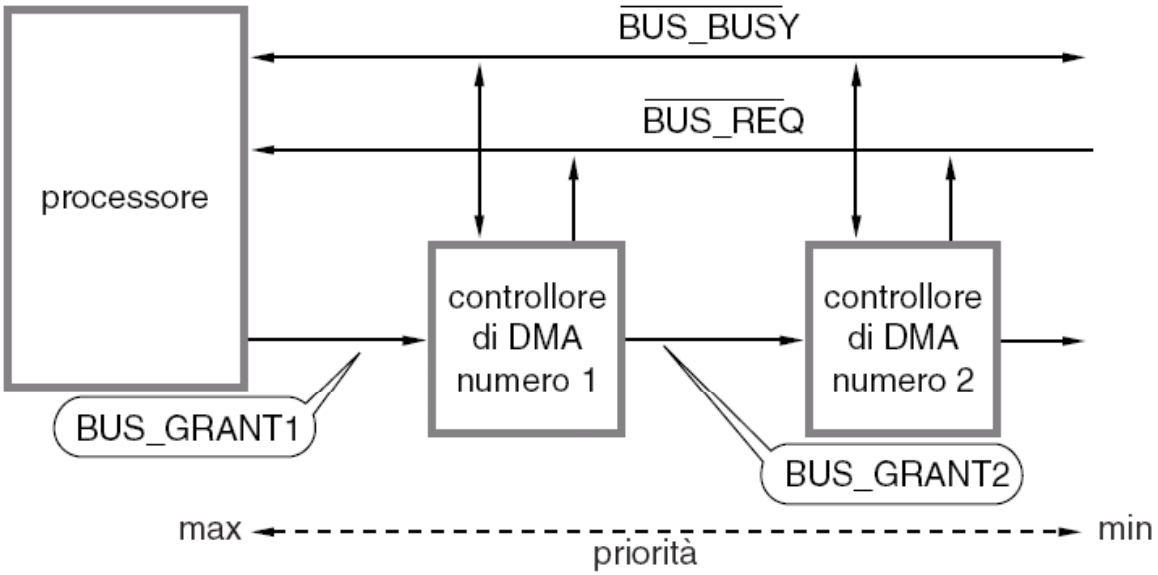
Arbitraggio del Bus

- Il bus del calcolatore può avere, in ogni istante, un solo MASTER.
- In caso di cessione del ruolo di MASTER da un'unità funzionale a un'altra, occorre dunque un meccanismo di arbitraggio del bus, che ne regoli l'utilizzo da parte delle unità funzionali.
- Esistono due meccanismi principali di arbitraggio:
  - centralizzato
  - distribuito
- Essi si distinguono per la presenza di un arbitro centralizzato, che dirime i conflitti di uso del bus, o l'assenza di tale arbitro.

Arbitraggio Centralizzato

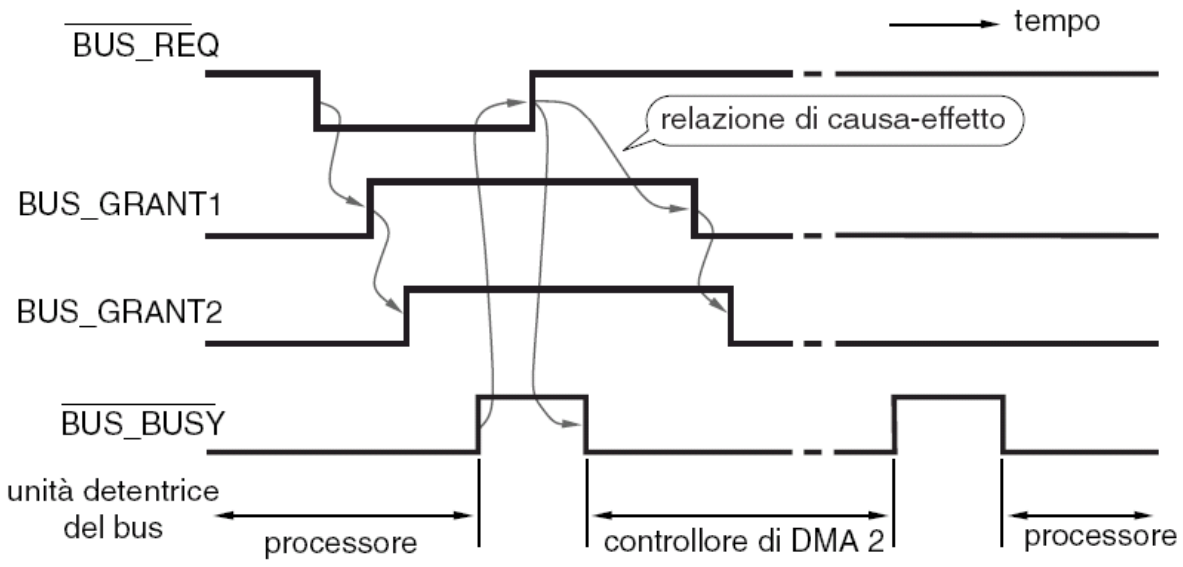
- Il meccanismo di arbitraggio centralizzato prevede:
  - un'unità funzionale apposita, che svolge la funzione di arbitro del bus
  - alcune linee (appartenenti al bus di controllo) che collegano l'arbitro alle unità funzionali potenziali richiedenti il controllo del bus, nel modo seguente:
    - Bus Request - richiesta di cessione del controllo
    - Bus Grant - conferma di cessione del controllo
- L'arbitro realizza il meccanismo di cessione del controllo del bus, vale a dire del ruolo di MASTER.
- Spesso l'arbitro è in realtà il processore stesso, o è integrato nel processore, come nell'esempio seguente, relativo alla gestione di DMA in collegamento a festone.

Arbitraggio Centralizzato



la linea Bus Busy (bus occupato) viene tenuta costantemente attiva dall'unità che detiene correntemente il controllo del bus

Andamento Temporale



Arbitraggio Distribuito

- Il meccanismo di arbitraggio distribuito prevede le risorse e le funzionalità seguenti:
  - non esiste arbitro (le unità arbitrano collettivamente)
  - ciascuna unità funzionale ha priorità fissata e diversa, espressa da un codice univoco associato all'unità stessa
  - le unità funzionali che abbisognano del bus sono in contesa e attuano un protocollo di scambio di informazione (vale a dire la priorità), per accordarsi su chi abbia priorità massima e pertanto vinca la contesa
  - l'unità vincitrice acquisisce il controllo del bus
- Il protocollo di scambio di informazione non è banale, e pertanto per i particolari si rimanda al testo.

Bus sincrono e asincrono

Operazioni di lettura e scrittura

FUNZIONAMENTO DEL BUS

Funzionamento del Bus

- In generale, ogni operazione sul bus corrisponde a un ciclo di bus. I cicli di bus sono classificabili come segue:
  - lettura di una parola di memoria
  - scrittura di una parola di memoria
  - lettura di un registro di I/O
  - scrittura di un registro di I/O
  - riposo: il bus non viene usato
- Una singola operazione di lettura o scrittura può anche svilupparsi su più cicli di bus:
  - l'uso di più cicli di bus si può rendere necessario quando un processore veloce debba trasferire dati con un'unità funzionale lenta
- Più avanti si vedrà un'operazione multiciclo.

Suddivisione in Cicli di Bus

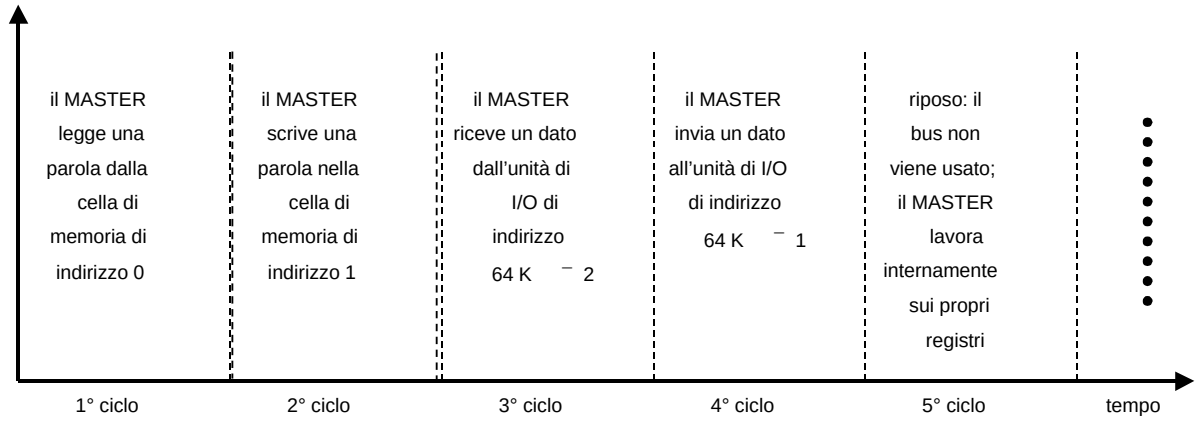


diagramma temporale (schematico)
dei cicli di bus

Bus Sincrono e Asincrono

- Per realizzare la scansione dei cicli di bus esistono due metodi fondamentali:
  - bus sincrono
  - bus asincrono
- I due metodi si distinguono in modo fondamentale per la presenza o meno di un segnale di clock, o segnale di scansione del tempo (temporizzazione).
- La presenza o assenza del segnale di clock impatta fortemente su come deve avvenire la sincronizzazione tra unità master e slave.

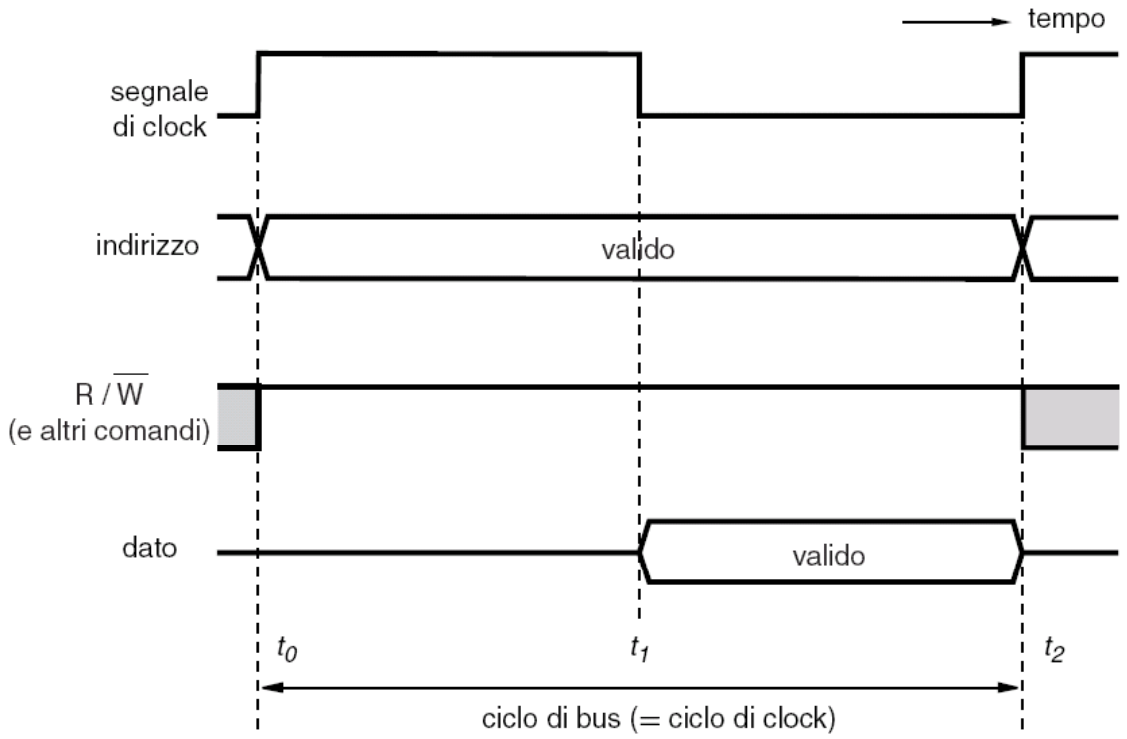
Bus Sincrono

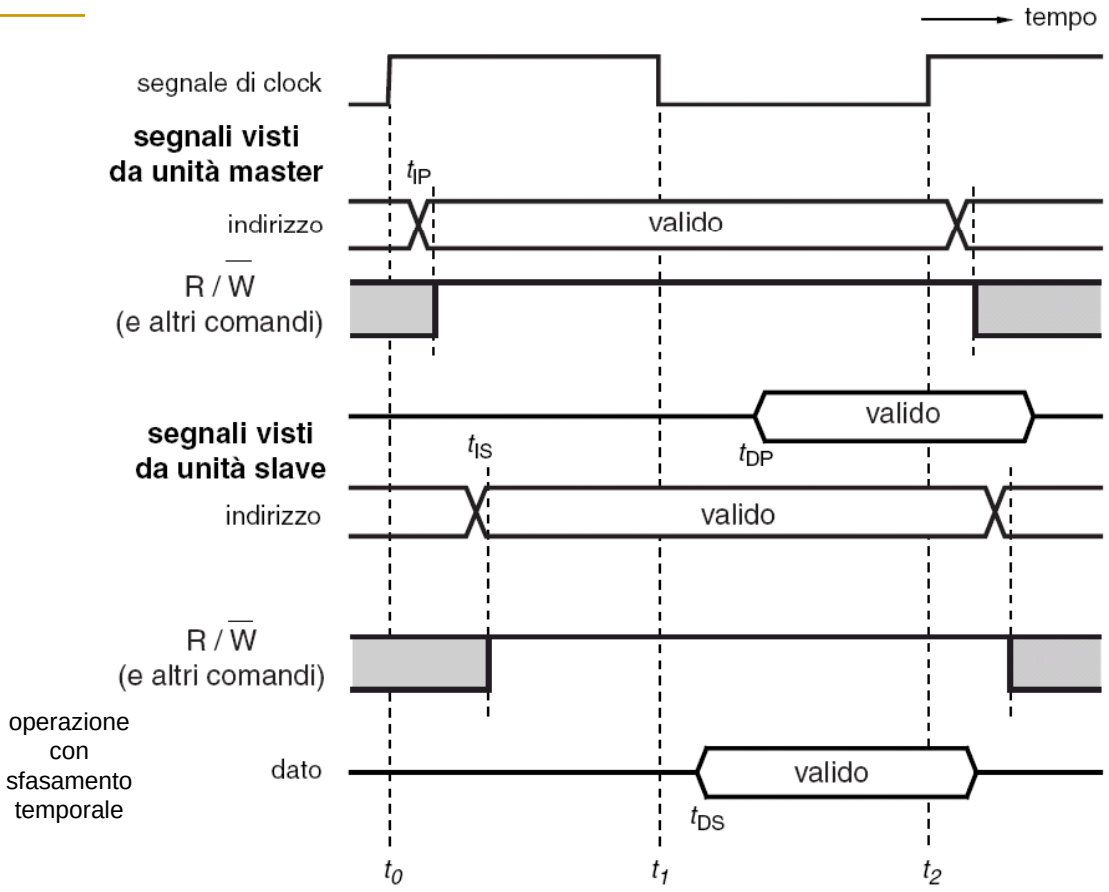
- Il bus di controllo contiene una linea per il segnale di clock, a frequenza prestabilita.
- Il clock viene distribuito a tutte le unità funzionali collegate al bus.
- Il segnale di clock scandisce le varie transizioni di segnale e il passaggio da un ciclo di bus al ciclo successivo.
- Tutte le unità funzionali sanno sempre quando si passa al ciclo successivo.

Esempio - Lettura

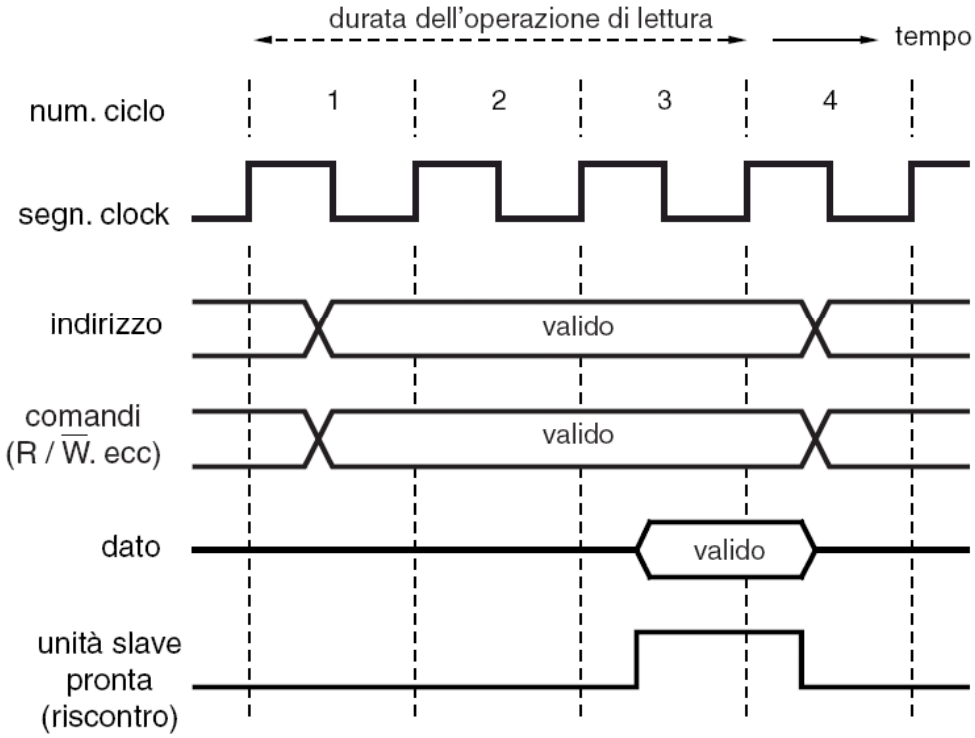
- Istante t_0: il MASTER (processore) emette l'indirizzo tramite il bus indirizzi e attiva il comando di lettura tramite il bus di controllo (insieme all'indirizzo o subito dopo averlo stabilizzato).
- Istante t_1: lo SLAVE (memoria) ha pronta la parola e la fornisce tramite il bus dati.
- Istante t_2: il MASTER legge la parola dal bus dati, toglie l'indirizzo dal bus indirizzi e disattiva il comando di lettura nel bus di controllo.
- Nota sui ritardi:
  - oltre ai ritardi di propagazione dei fronti salita e discesa dei vari segnali, sia il MASTER sia lo SLAVE hanno dei ritardi interni, dovuti allo loro struttura
  - questi ritardi dipendono in generale dalla specifica tecnologia di bus adottata
- Nel seguito è mostrato anche un diagramma temporale che espone tali ritardi, di propagazione e interni alle unità.

Bus Sincrono - Lettura





Bus Sincrono - Lettura Multiciclo



Bus Asincrono

- Non esiste alcun segnale di clock comune alle unità funzionali collegate al bus del calcolatore.
- Le transizioni di segnale e i passaggi da un ciclo di bus al ciclo successivo non sono sincronizzati.
- Le unità funzionali osservano il bus di controllo:
  - quando avviene una transizione di segnale, significa che si verifica un avanzamento dell'operazione
  - il bus è dotato di opportuni segnali di controllo, di cui vengono osservate le transizioni
  - tali segnali realizzano un protocollo di sincronizzazione a segnale (per esempio "full-handshake")
- Naturalmente indirizzo e dato passano sui bus rispettivi.

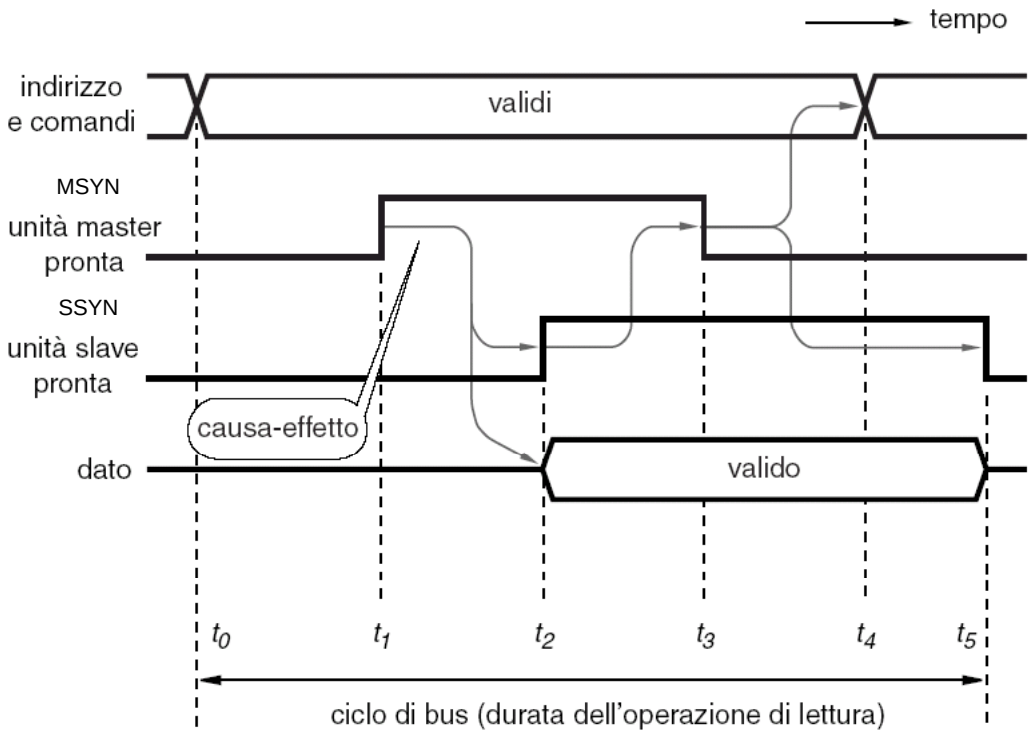
Bus Asincrono

- Oltre ai segnali di controllo già visti per il bus sincrono, tranne il segnale di clock che qui manca, sono presenti due segnali appositi:
  - Unità Master Pronta (master synchronisation, MSYN): l'unità MASTER segnala di avere emesso indirizzo (tramite il bus indirizzi) e dato il comando di lettura o scrittura R / W (tramite il bus di controllo)
  - Unità Slave Pronta (slave synchronisation, SSYN): l'unità SLAVE segnala di avere completato l'operazione (emesso il dato se è lettura, acquisito il dato se è scrittura)
- Lo scambio di tali segnali speciali realizza la sincronizzazione; nella forma tipica comprende quattro fasi distinte (vedi diagramma di seguito).

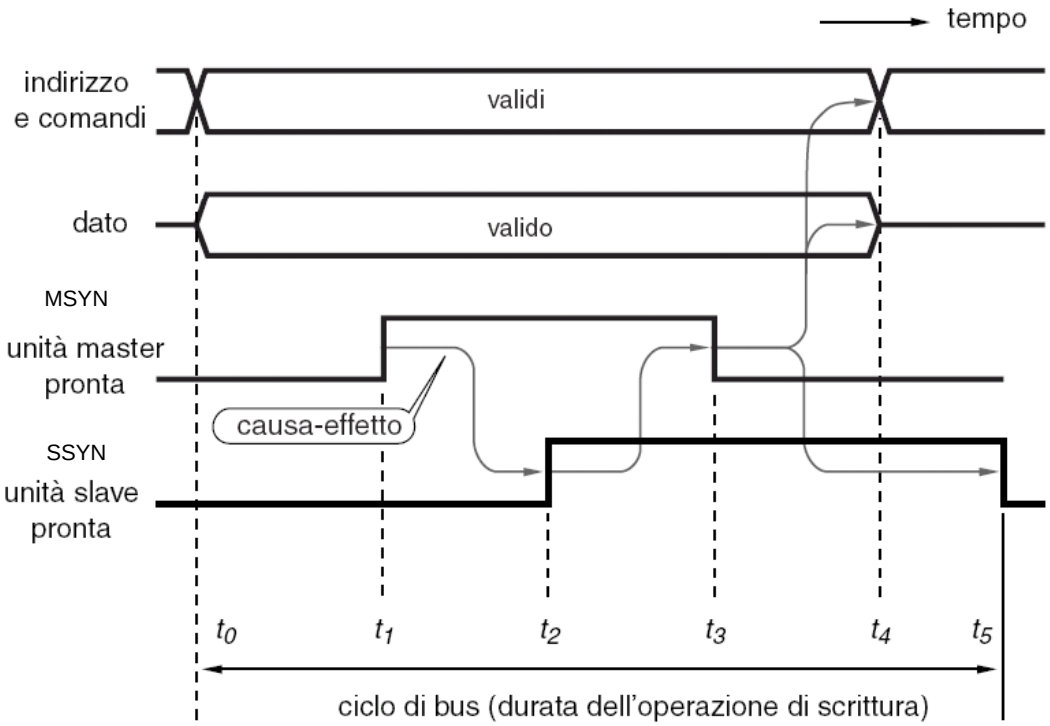
Esempio - Lettura

- Il MASTER (processore) manda l'indirizzo sul bus indirizzi, attiva il comando di lettura, e dopo averli tutti stabilizzati, attiva MSYN.
- Lo SLAVE (memoria) esegue l'operazione, nel minore tempo possibile, e fornisce la parola sul bus dati; poi attiva SSYN.
- Il MASTER legge la parola dal bus dati, toglie l'indirizzo e disattiva il comando di lettura; poi disattiva anche MSYN.
- Infine, lo SLAVE disattiva SSYN:
  - poiché non esiste un segnale di clock che marchi quando attivare o disattivare i vari segnali di controllo, nel diagramma temporale vanno indicati i rapporti causa-effetto esistenti tra i vari segnali di controllo
  - i segnali di controllo transitano di valore in reazione a transizioni precedenti
- Di seguito c'è il diagramma temporale delle fasi.

Bus Asincrono - Lettura



Bus Asincrono - Scrittura



Porta di lettura e scrittura
Porta seriale e parallela

INTERFACCE DI PERIFERICA (PORTE DI I/O)

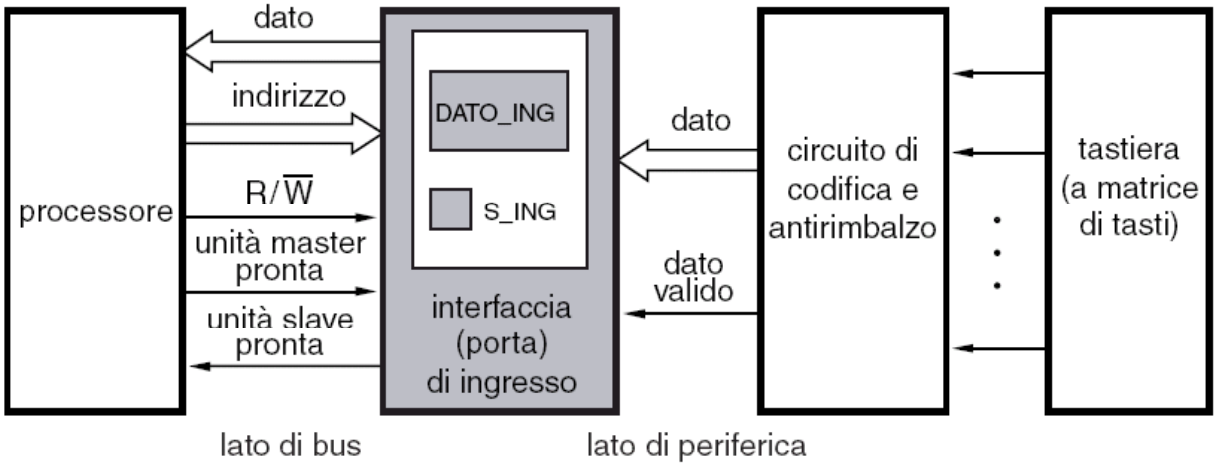
Interfaccia di I/O

- Le interfacce di I/O hanno una struttura che generalizza (in modo più o meno ampio) l'esempio di porta di terminale di prima.
- Ci sono alcune categorie di interfacce più o meno standardizzate, che si trovano ovunque.
- Le due più semplici e molto diffuse sono:
  - porta parallela
  - porta seriale
- Sono illustrare di seguito, con schema circuitale di massima; per i dettegli vedi il testo.

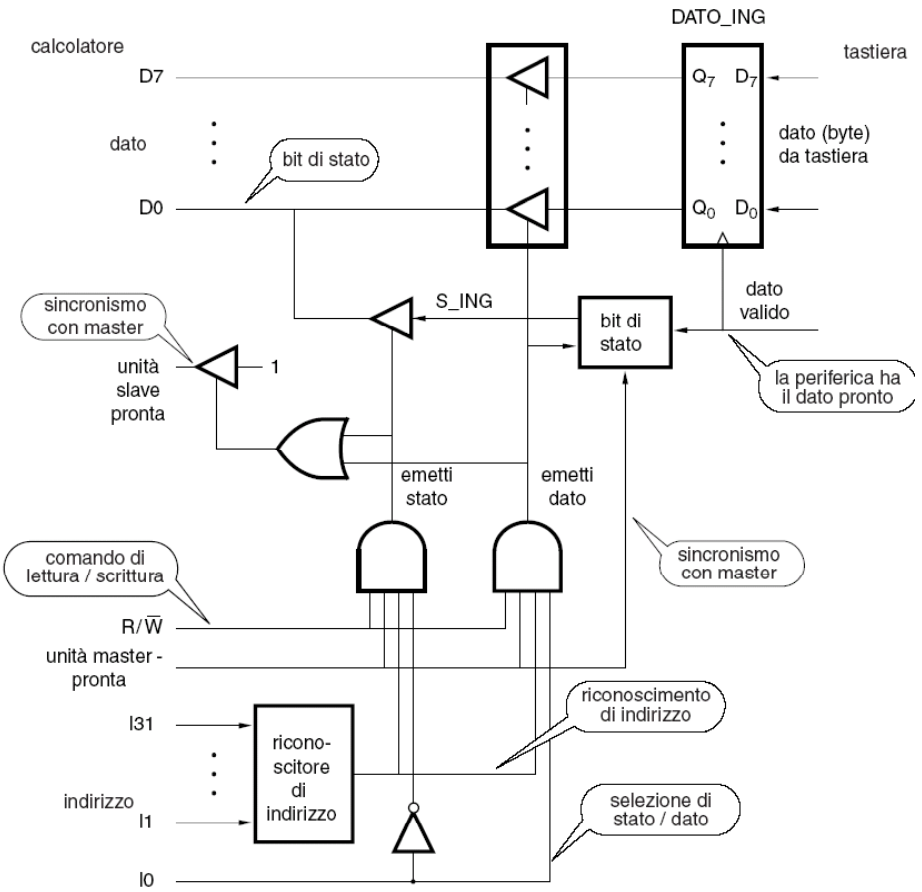
Porta Parallela

- La porta parallela scambia tra calcolatore e periferica una parola per volta (un byte o una parola più lunga, 16, 32 bit, ecc).
- È dotata di registro di dato, registro di stato (con bit di stato ed eventuali altri), ed eventuali registri di controllo (per funzioni varie).
- Dal lato di periferica, scambia il dato con la periferica stessa, e segnali di sincronizzazione vari con essa (vedi schema).
- Dal lato di calcolatore, ha la struttura ormai nota; di seguito sono mostrate la versione in sola lettura e quella in sola scrittura, entrambe collegate a bus di tipo asincrono (per il commento allo schema circuitale di lettura si veda il testo); ovviamente si possono collegare anche a bus di tipo sincrono.
- Naturalmente le due versioni si possono riunire a formare una porta parallela funzionante in lettura e scrittura (vedi testo).

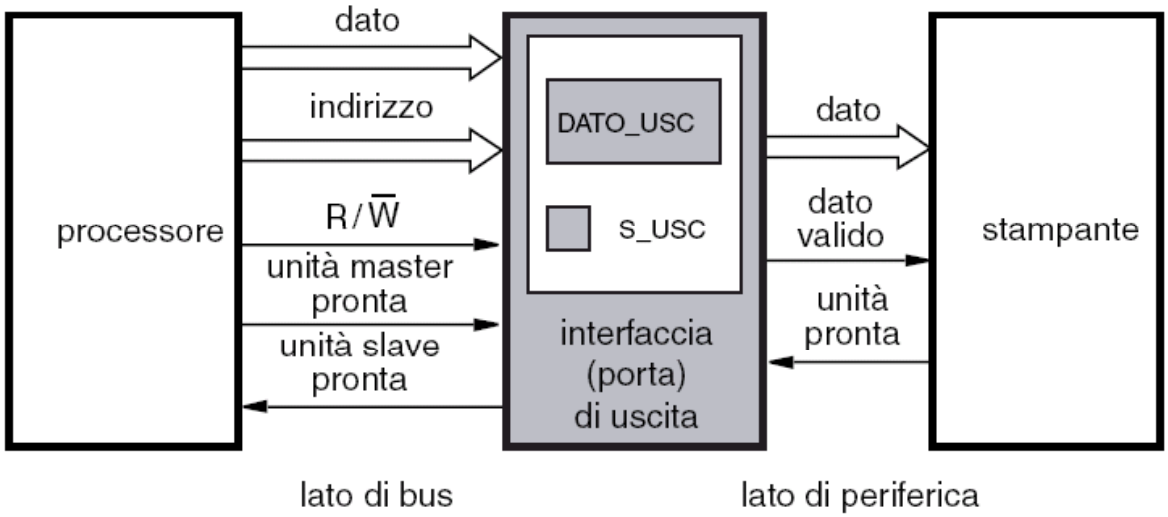
Porta Parallela di Ingresso



schema circuitale di porta parallela in ingresso



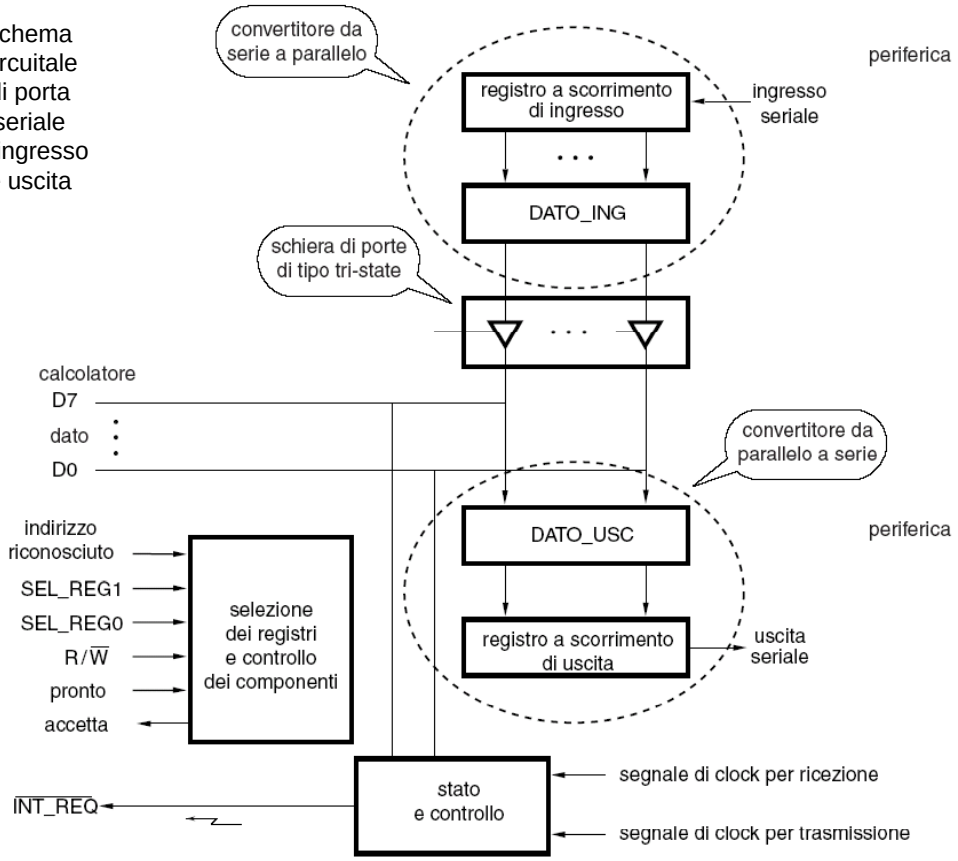
Porta Parallela di Uscita



Porta Seriale

- La porta seriale scambia tra calcolatore e periferica una parola per operazione (un byte o una parola più lunga, 16, 32 bit, ecc).
- È dotata di registro di dato, registro di stato (con bit di stato ed eventuali altri), ed eventuali registri di controllo (per funzioni varie).
- Dal lato di periferica, scambia il dato con la periferica stessa, e segnali di sincronizzazione vari con essa (vedi schema), in modo seriale, cioè un bit per volta.
- Dal lato di calcolatore, ha la struttura ormai nota, e comunque scambia il dato in forma parallela (giacché il bus dati del calcolatore è un fascio di linee, non una linea singola).
- Pertanto la porta seriale contiene un dispositivo di conversione da seriale e parallelo, e viceversa (di fatto, un registro a scorrimento).
- Di seguito è mostrato direttamente lo schema circuitale di massima, per porta seriale funzionante in lettura e scrittura.

schema circuitale di porta seriale in ingresso e uscita

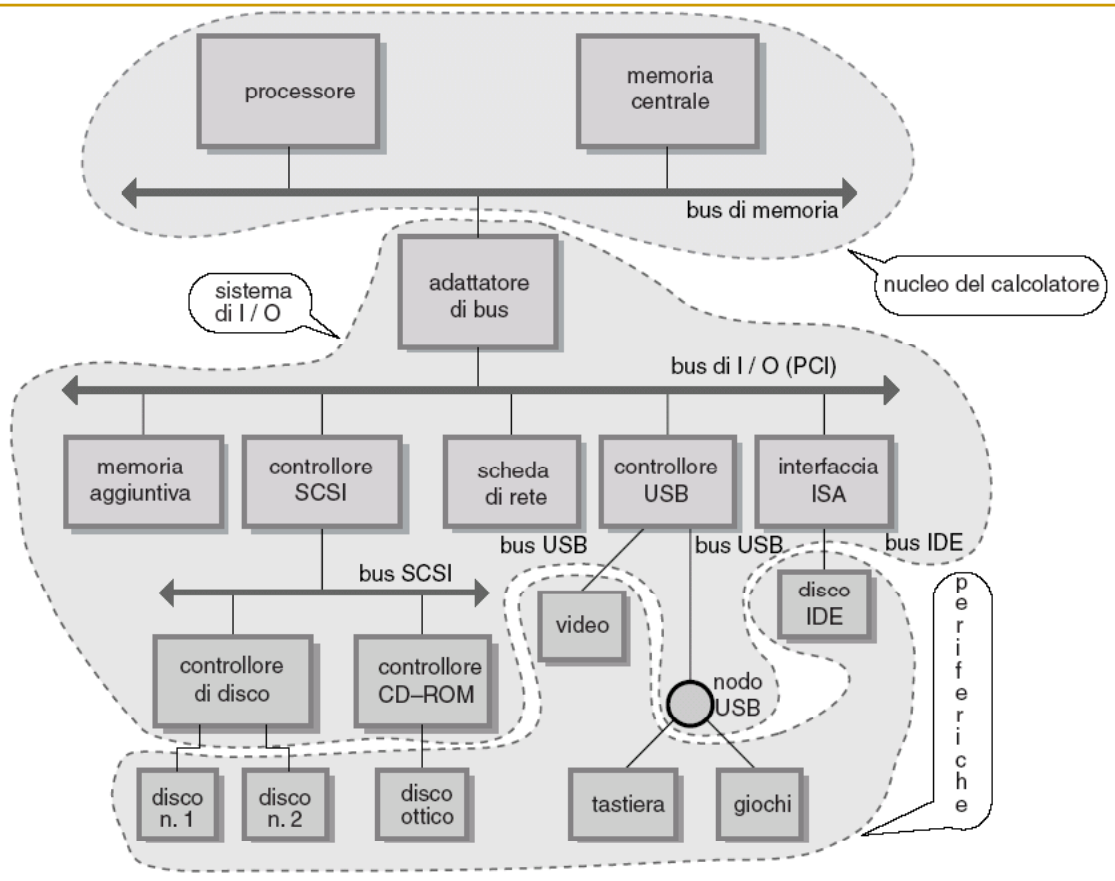


Uno sguardo d'insieme

SISTEMA DI INGRESSO-USCITA

Struttura Generale

- Il calcolatore contiene almeno un bus, per collegare processore e memoria; se esso ha solo tale funzione è detto bus di memoria (memory bus).
- Se il bus è unico, allora collega sia la memoria sia le interfacce di I/O, ed è detto bus di sistema (system bus).
- Spesso tuttavia è presente un secondo bus, cui è collegato l'insieme di interfacce che costituiscono il sistema di I/O.
- Un'unità apposita (bridge o ponte) collega i due bus di memoria e di I/O, e trasferisce le operazioni di I/O (che hanno origine dal processore o comunque dal MASTER), sul bus di I/O.
- Ciascuna interfaccia è poi collegata alla periferica (o a un gruppo di periferiche) tramite un bus specializzato per tale periferica; questi bus si dicono esterni (o di periferia).



Bus di Sistema e di I/O

- In genere il bus di memoria (se è presente) è specifico del processore considerato, perché deve funzionare a prestazioni elevatissime (numero di dati scambiabili nell'unità di tempo).
- Il bus di sistema (se è presente) è comunemente standard; vi sono vari standard disponibili, più o meno simili.
- Il bus di I/O (se è presente) è pure standard; vi sono vari standard disponibili, più o meno simili.
- In linea di principio, il calcolatore potrebbe anche contenere due o più bus di I/O, variamente specializzati per gruppi di periferiche.
- Anche i bus esterni di periferica sono più o meno standard; data la varietà di periferiche disponibili, tali standard di bus esterno sono però molto numerosi e radicalmente differenziati.

Bus di Periferia

- CENTRONICS, fa riferimento alla porta parallela CENTRONICS (a 8 bit), nota anche come LPT.
- RS-232 (e successive evoluzione), fa riferimento alla porta seriale RS-232, nota anche come COM.
- IDE (per dischi e simili), fa riferimento alla porta IDE
- SCSI (per dischi veloci e simili), fa riferimento alla porta SCSI (parallela, a 8 e 16 bit).
- USB (per periferiche a media velocità), fa riferimento alla porta USB (seriale, funziona a pacchetti).
- e molti altri ancora …

Porte di I/O Standard

- Ai bus esterni (di periferia) standard, fanno riscontro altrettante porte di I/O standard. Per esempio:
  - CENTRONICS parallela, a 8 bit
  - RS-232, seriale (ha diverse evoluzioni)
  - porte di tastiera e puntatore (mouse)
  - porta USB (seriale, a pacchetti)
  - porta IDE e SCSI
  - e altre porte di natura più specialistica (audio, video, ecc)
- Spesso la porta seriale è realizzata tramite il componente integrato USART (Universal Synchronous Asynchronous Receiver Transmitter), che contiene tutte le funzionalità di scambio dati per periferiche seriali, in versione sincrona e asincrona, ed è programmabile in tali modi di funzionamento.