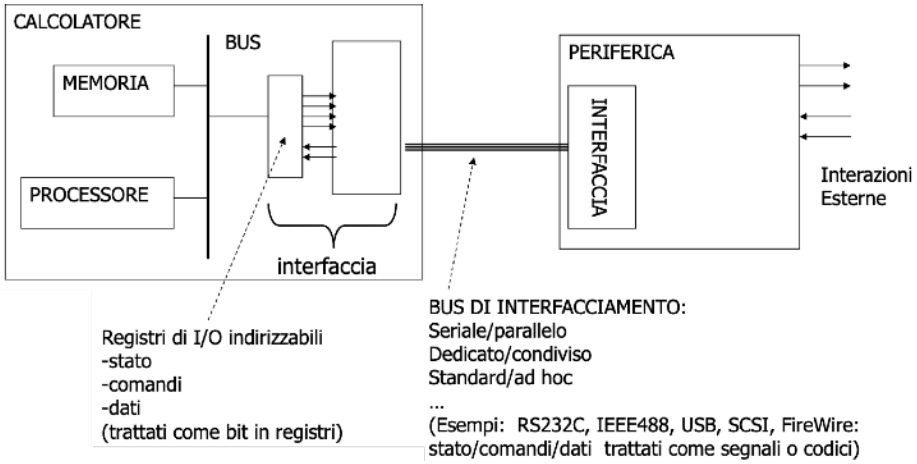
05 - INPUT E OUTPUT A LIVELLO HARDWARE

ACCESSO ALLE PERIFERICHE

Le unità periferiche interagiscono con il calcolatore, e cioè con il processore e la memoria centrale, attraverso **interfacce di ingresso/uscita** (dette anche **adattatori**) collegate al bus di sistema.

- Le interfacce sono quindi dispositivi circuitali che consentono al calcolatore di scambiare informazioni con le unità periferiche
- La struttura delle interfacce di ingresso/uscita è costituita, in modo molto schematico, da alcuni registri leggibili e/o scrivibili da parte del processore, ma comunicanti anche con il mondo esterno (periferiche), e da circuiti ausiliari

Collegamento di una periferica al calcolatore (a livello di sistema)



Registri di interfaccia di una periferica (schema semplificato)

- **Registro dati periferica**
  - contiene i dati che la periferica deve leggere o scrivere
- **Registro comandi (o controllo) periferica**
  - contiene indicazioni sulle operazioni che la periferica deve compiere e sul suo modo di funzionamento
- **Registro** di stato periferica
  - contiene informazioni sullo stato di funzionamento della periferica (ad esempio "pronta" a gestire un nuovo dato, o "occupata")
  - bit ***Ready***: relativo alla possibilità, per il processore, di svolgere un'operazione di lettura o scrittura di un dato
    - una periferica di ingresso (es. tastiera) è in stato di pronto quando un tasto è stato premuto e quindi esiste un carattere nel suo registro dati che il processore può leggere
    - una periferica di uscita (es. stampante) è pronta se può ricevere un dato dal processore per stamparlo.

Esempi di funzionamento

- Una **stampante** può funzionare in modo semplificato come segue:
  - appena "accesa" Ready=1, registro dati vuoto
  - quando la CPU scrive un dato nel registro dati Ready va a 0
  - quando la stampante ha finito di stampare il dato e il registro dati è nuovamente vuoto, Ready torna a 1

- Una **tastiera** può funzionare in modo semplificato come segue:

- appena “accesa” Ready=0, registro dati vuoto
- quando viene premuto un tasto, Ready va a 1
- quando la CPU legge il dato, Ready torna a 0

Istruzioni di input output specializzate

Port Mapped I/O

- Il processore interagisce con le periferiche utilizzando istruzioni specializzate per l’I/O.
- Tali istruzioni operano sui registri delle periferiche
- Gli indirizzi di tali registri sono detti Port
- Istruzioni fondamentali:
  - IN `port_di_input` (legge dal registro associato a `port_di_input`)
  - OUT `port_di_output` (scrive nel registro associato a `port_di_output`)

Memory mapped I/O

- Il processore accede ai registri delle periferiche con le stesse istruzioni di **load/store** utilizzate per accedere alla memoria assegnando ai registri delle periferiche indirizzi omogenei con quelli della memoria
- Utilizzando questo schema un’adeguata porzione dello spazio di indirizzamento va riservata per gli indirizzi delle periferiche.
- Il processore MIPS utilizza questo schema riservando i primo 256 indirizzi di memoria per indicare le periferiche (da 0x0000 0000 a 0x0000 00FF)

Tecniche di gestione dell’input/output

La gestione dell’I/O può avvenire tramite:

- **Controllo di programma**
- **Meccanismo di interruzione**
- **Accesso Diretto alla Memoria (DMA)**

Le tecniche di gestione devono consentire **interazioni** tra calcolatore e periferica per svolgere:

- la ***sincronizzazione*** tra periferica e calcolatore;
- il ***trasferimento*** del dato da periferica a calcolatore o viceversa.

esempi - sincronizzazione e trasferimento

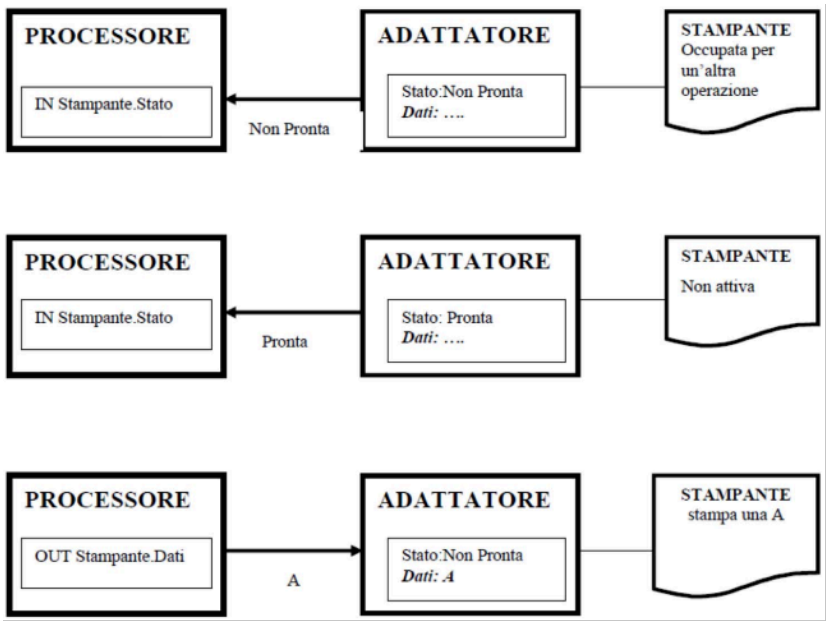
- **Stampa di un carattere su stampante: interazioni**
  - la stampante deve segnalare al calcolatore che è disponibile (pronta) ad accettare un nuovo dato per la stampa (sincronizzazione);
  - il calcolatore esegue un’istruzione di uscita `OUT` sul registro dati della stampante, che potrà quindi eseguirne la stampa (trasferimento).
- **Acquisizione di un carattere da tastiera: interazioni**
  - la tastiera deve segnalare al calcolatore che è stato premuto un tasto e quindi esiste un carattere disponibile (sincronizzazione);
  - il calcolatore esegue un’istruzione di ingresso `IN` dal registro dati della tastiera per acquisire il valore del carattere (trasferimento).

Gestione dell’I/O a controllo di programma

- la sincronizzazione e il trasferimento vengono eseguiti dal programma di gestione della periferica (**device driver**)
- Il programma di gestione della periferica
  - **legge il registro di stato** dell’interfaccia della periferica (`IN registro_di_stato`)
  - se lo stato è “periferica non pronta” il programma entra nel ciclo di attesa che la periferica sia pronta e torna a leggere il registro di stato; altrimenti
  - **esegue il trasferimento** del dato tramite:

- `IN_registro_dati` se la periferica è di ingresso (ad es. lettura dato da tastiera)
- `OUT_registro_dati` se la periferica è di uscita (ad es. scrittura dato verso stampante)

esempio - stampante a controllo di programma



In realtà non è la periferica bensì il suo adattatore (detto anche interfaccia o controllore) che gestisce i registri e l'interazione con la CPU.

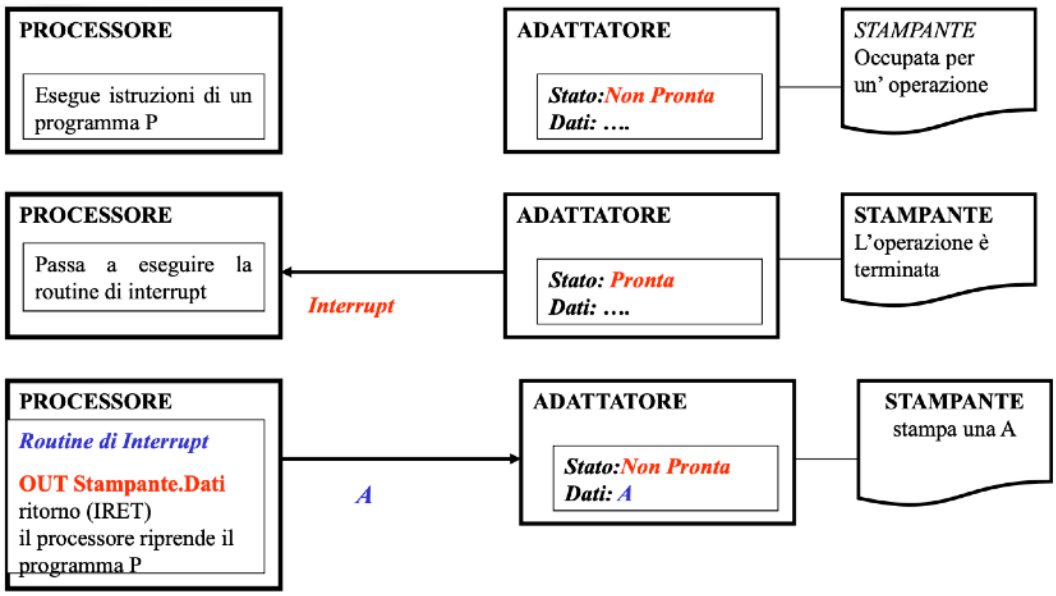
Gli adattatori accoppiano i diversi tipi di CPU con i diversi tipi di periferica.

Dal controllo di programma al meccanismo di interrupt

Problema: la gestione delle periferiche a controllo di programma implica da parte del processore un **ciclo di attesa di periferica pronta**.
Nel Sistema Operativo invece vogliamo che il processore ponga il processo in attesa e passi ad eseguire altre attività fino a quando la periferica non diventa pronta

Questo obiettivo viene ottenuto utilizzando il **meccanismo di interrupt:**
- richiede che una periferica segnali tramite un apposito **interrupt** il passaggio da occupata a pronta
- ovvero la transizione del bit Ready da 0 a 1

esempio - stampante gestita da meccanismo di interrupt



Gestione dell'I/O basato su interrupt

Il meccanismo di interruzione (**interrupt**) si basa su:

- un insieme di **eventi** rilevati a livello hardware dal processore (**interruzioni**). Ad es. un segnale opportuno di una periferica, una condizione di errore …
- Un insieme di **funzioni**, ognuna associata in generale ad un evento. Ogni funzione viene chiamata **gestore dell'interrupt o routine di interrupt**

Quando il processore rileva un evento:

- **interrompe il programma** in esecuzione
- **salta automaticamente ad eseguire la routine di interrupt** corrispondente, ed esegue il servizio richiesto
- al termine dell'esecuzione della routine, **torna ad eseguire il programma** che era stato interrotto (in modo **trasparente** rispetto al programma stesso)

esempio - tastiera con gestione a interrupt

Acquisizione di un carattere da tastiera con gestione a interrupt

- Quando l'interfaccia della periferica scrive il dato nel registro dati, con un **segnale allerta il processore (sincronizzazione).**
- Il processore **interrompe** l'esecuzione del programma in corso e salta automaticamente a eseguire la parte di programma che legge il registro dell'interfaccia (**trasferimento**).
- Al termine di questo, il processore riprende il programma interrotto.

Il meccanismo di interruzione è **simile** a quello di invocazione di una funzione e infatti prevede:

- la cessione del controllo dell'esecuzione tra il programma interrotto e la routine di risposta all'interrupt.
  - è quindi necessario **salvare sullo stack l'indirizzo di ritorno** di tale programma (e cioè l'indirizzo dell'istruzione successiva a quella in cui il programma è stato interrotto)
- il ritorno all'esecuzione del programma interrotto
  - le routine di risposta all'interrupt utilizzano, per il ritorno, una istruzione macchina **IRET speciale** e diversa da RFS.

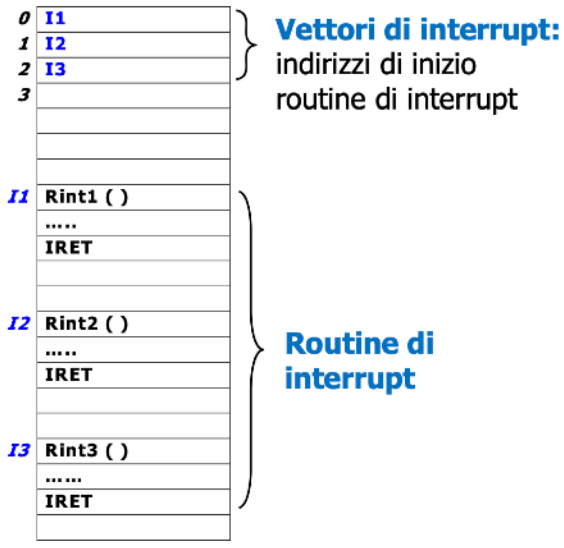
La **differenza fondamentale** rispetto alle chiamate a funzione è che queste sono attivate dal programma in esecuzione, mentre le routine di interrupt sono "scatenate" da eventi rilevati dal processore e asincroni rispetto all'esecuzione del programma che viene interrotto

Interrupt annidati

- il salvataggio dell'indirizzo di ritorno sullo stack consente di gestire interruzioni annidate, al pari di quanto accade nella gestione delle funzioni nidificate

- Un interrupt può quindi interrompere, in linea di principio, la routine di risposta di un altro interrupt (interrupt annidati) in base a livelli di priorità degli interrupt

Vettori e routine di interrupt



Accesso diretto alla memoria - DMA

Il meccanismo di accesso diretto alla memoria (**DMA – Direct Memory Access**) da parte di periferiche prevede che:

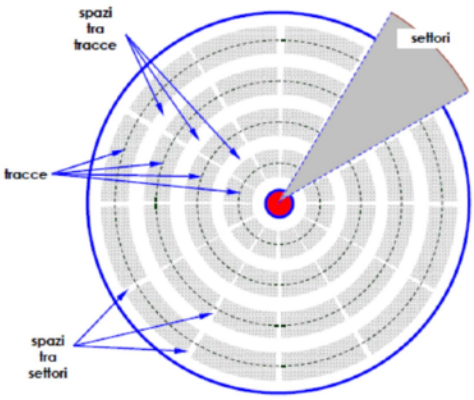
- Il driver della periferica trasferisca in modo **autonomo**, cioè **senza l'intervento del processore,** un certo numero di dati in memoria centrale o dalla memoria centrale
- Il meccanismo di DMA è utilizzato da periferiche quali i dischi magnetici dotate di interfacce "intelligenti" (DMA e controllori del disco) che sono in grado di trasferire velocemente uno o più settori di disco da o in memoria centrale

Il **DMA Controller (o Adattatore DMA**) effettua il trasferimento dati a blocchi tra periferiche e memoria autonomamente, cioè senza l'intervento da parte del processore. Il controllore del DMA è quindi in grado di **generare i segnali per accedere a memoria** (indirizzo, dati, controllo)

Dispositivi di memorizzazione non-volatile

- Le interfacce dei dispositivi di memorizzazione non-volatile **Hard Disk Drivers (HDD)** e **Solid State Drive (SSD)** utilizzano la tecnica di gestione ad accesso diretto alla memoria (**DMA**) per il trasferimento **fisico** dei dati
- Dal punto di vista logico (File System) l'indirizzamento dei dati si basa sullo schema di indirizzamento **LBA (Logical Block Address)** nel quale l'intero disco è rappresentato come un vettore lineare di blocchi
- Nel seguito useremo il termine **Volume** per indicare qualsiasi dispositivo di memorizzazione non volatile di massa, siano essi **HDD** oppure **SSD**, dotato di uno schema di indirizzamento **LBA**.

- A livello logico il **blocco** costituisce l'unità fondamentale di informazione che viene trasferita con una sola operazione tra il disco e la memoria centrale
- A livello fisico su HHD, i dati sono memorizzati su disco in cerchi concentrici detti tracce e le tracce sono suddivise in settori.
- Il settore (o più d'uno) è l'unità che viene trasferita con una sola operazione di accesso tra il disco e la memoria centrale (il blocco è quindi un multiplo del settore).
- L'operazione di trasferimento è controllata dall'interfaccia del disco



DMA

La tecnica **DMA** prevede le seguenti fasi:

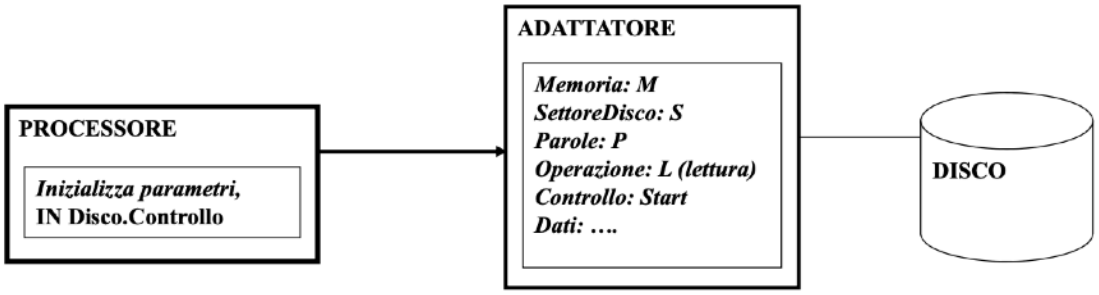
1. **Predisposizione** → Una procedura (del Sistema Operativo) scrive nei registri dell'interfaccia (o adattatore) delle periferica:
   - l'indirizzo della **memoria** dal quale iniziare il trasferimento
   - l'indirizzo sulla **periferica** dal quale iniziare il trasferimento
   - il **numero di dati** (parole di memoria) da trasferire
   - la **direzione** del trasferimento (lettura da periferica o scrittura su periferica)

2. **Attivazione** → La procedura del Sistema Operativo scrive nel registro di controllo dell'interfaccia il comando di avviamento **"start"** dell'operazione e quindi il processore passa ad eseguire altri programmi. La periferica può a questo punto procedere in modo **autonomo** nel trasferimento.

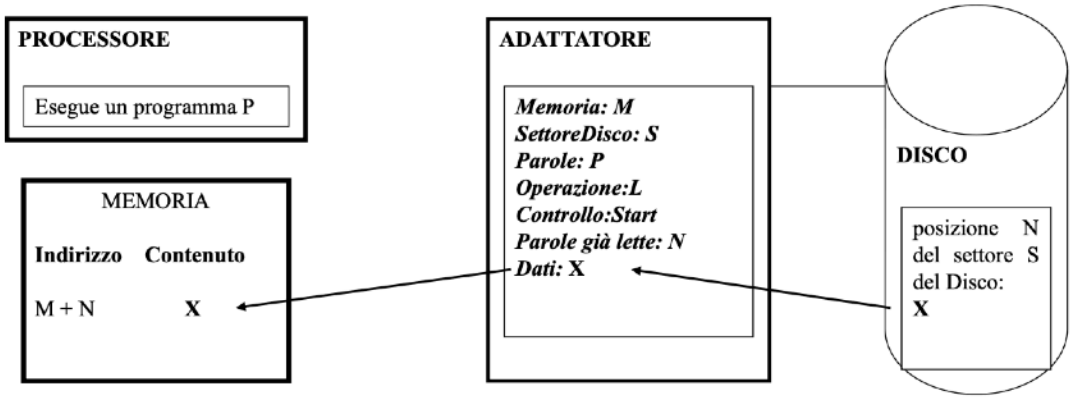
3. **Trasferimento di un blocco di dati**
   - Nel caso di **lettura** da periferica, ogni volta che dalla periferica arriva un dato all'interfaccia, l'interfaccia scrive il dato in memoria, incrementa l'indirizzo in memoria, decrementa il numero di dati ancora attesi e, se non sono arrivati tutti, aspetta il prossimo dato
   - Nel caso di **scrittura** su periferica, ogni volta che la periferica è disponibile ad accettare nuovi dati, l'interfaccia preleva il dato da memoria, incrementa l'indirizzo in memoria, decrementa il numero dei dati ancora da trasferire e, se non sono trasferiti tutti, aspetta che la periferica sia disponibile per il prossimo dato

4. **Conclusione** → Se l'ultimo dato è stato trasferito, l'interfaccia segnala al processore tramite **interrupt** che l'operazione in DMA si è conclusa

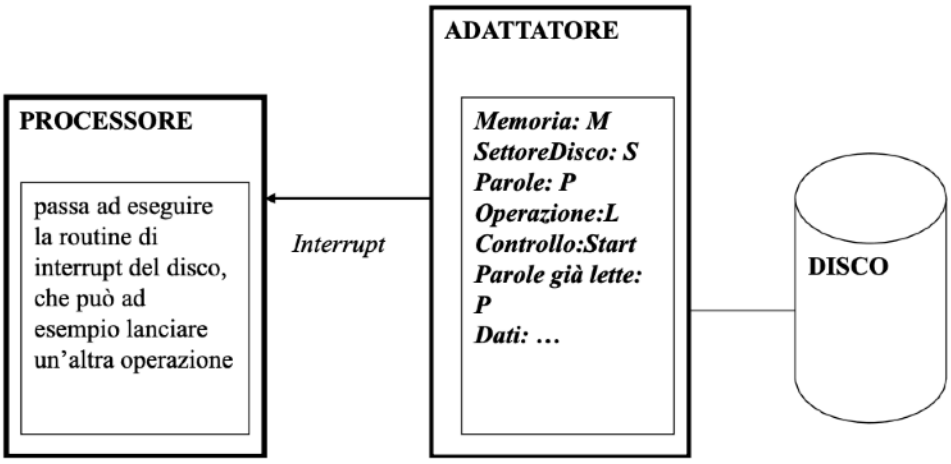
esempio di operazione di lettura da disco: il processore inizializza l'adattatore del disco



esempio di operazione di lettura da disco: N parole già trasferite da disco



esempio: l'adattatore del disco genera un interrupt alla fine della lettura di P parole



DMA e sistema operativo

Ai fini della realizzazione del Sistema Operativo, le periferiche gestite in **DMA** richiedono che il Sistema Operativo gestisca opportunamente delle aree di memoria (dette buffer) per il trasferimento dati in modalità DMA da/per le periferiche.

Inoltre, Linux nella definizione dei ***device driver*** distingue tra:
- **Block Devices**: periferiche a blocchi in DMA, tipicamente i dischi (caratterizzate da accesso casuale o diretto)
- **Character Devices**: periferiche a caratteri, tipicamente tastiera e stampante (caratterizzate da accesso sequenziale)

IL BUS DEL CALCOLATORE

Le unità funzionali di un calcolatore (processore, coprocessori, memoria, unità di interfacciamento alle periferiche) sono interconnesse tramite un organo di collegamento: il **BUS**

Una **transazione di bus (o ciclo di bus)**: è un insieme di operazioni sul bus che permette di raggiungere un obiettivo
- **Transazione di trasferimento**: trasferisce una certa quantità di informazione tra due unità funzionali
- **Transazione di interrupt**: protocollo di segnalazione di un interrupt alla CPU e accettazione, (predisposizione dell'attivazione della routine di risposta all'interrupt)

La quantità di informazione trasferita da un singolo ciclo di bus varia in genere da 8 a 64 bit.

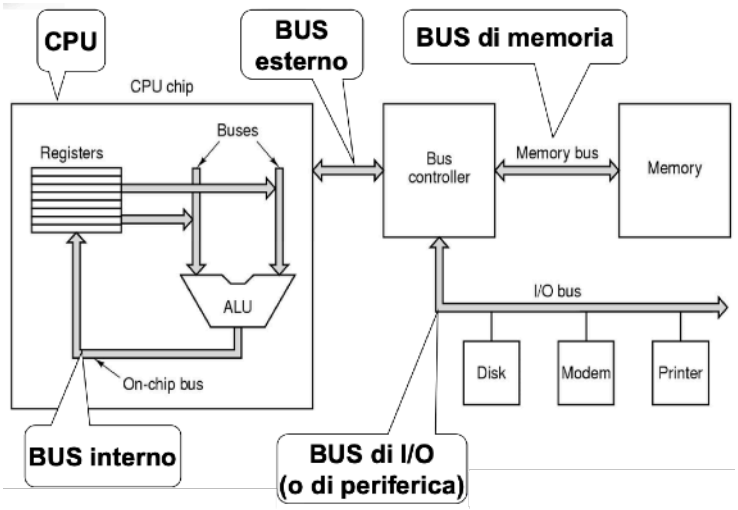
Bus interni ed esterni

I **BUS** del calcolatore si distinguono in:
- **BUS interni**, confinati all'interno d'una singola unità funzionale, i quali collegano i blocchi funzionali contenuti nell'unità;
- **BUS esterni**, che collegano le unità funzionali; sono solitamente standardizzati

I calcolatori più semplici sono dotati di **un solo BUS esterno (BUS di sistema)** che collega CPU, memoria e unità di I/O

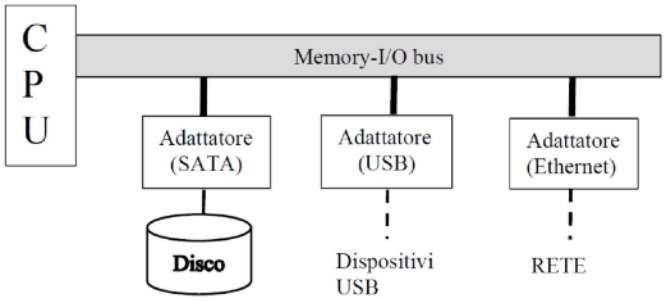
La maggior parte dei calcolatori odierni è dotata di **numerosi BUS esterni**, in particolare:
- **BUS di memoria**, collegante la CPU e le unità funzionali di memoria (banchi di memoria);
- **BUS di I/O**, collegante la CPU e le unità funzionali di I/O

Sistema con diversi BUS



Memory-I/O bus unico

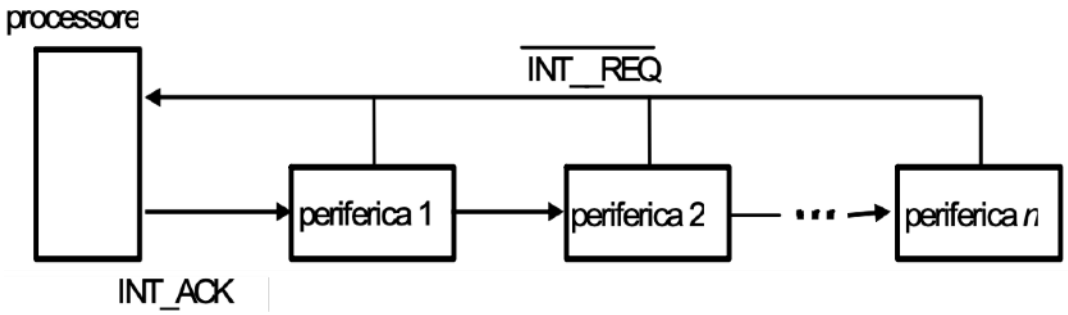
Esempio di bus unico (Memory-I/O bus) per la connessione della CPU con la memoria e con gli adattatori delle periferiche; questo schema è tipicamente impiegato nei sistemi memory mapped I/O come MIPS



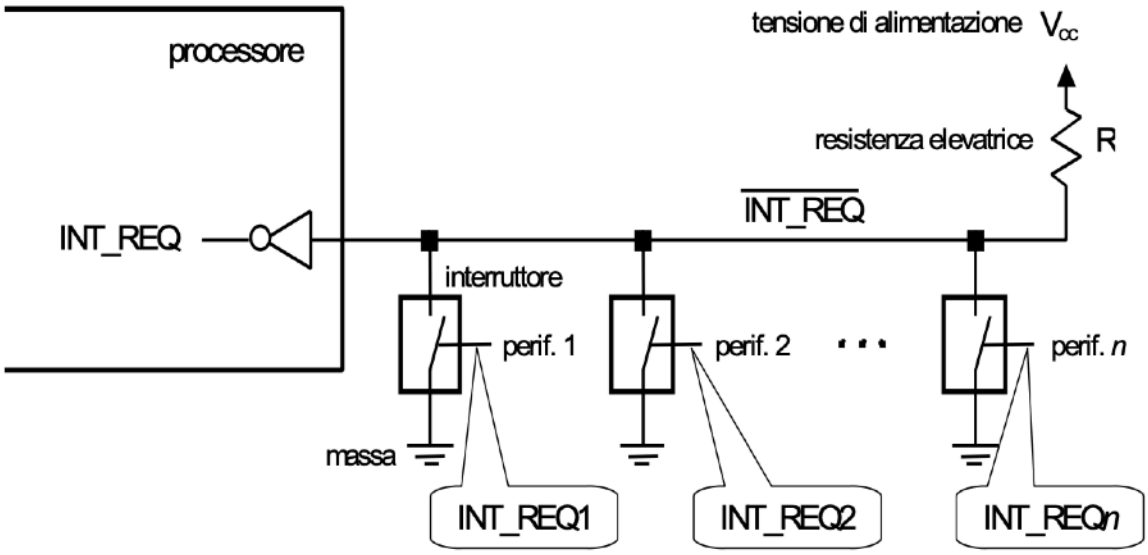
Transazione di interrupt

È l'insieme di operazioni che conduce da una segnalazione di interrupt da parte di una periferica alla sua presa in carico da parte della CPU.
Esistono diversi modi di collegare le linee di interrupt delle periferiche alla CPU; ad esempio:

- `INT_REQ` (richiesta di interrupt) in `wired_or`
- `INT_ACK` (accettazione dell'interrupt) in **daisy chain** per la propagazione del segnale
  - in corrispondenza della generazione del segnale di `INT_ACK` può partire un ciclo di trasferimento (lettura) in cui il processore acquisisce dalla periferica interessata **il vettore di interrupt**



Collegamento WIRED NOR di una linea di INT_REQ (segnale attivo basso)



Transazione di trasferimento

Una transazione di trasferimento si può generalmente scomporre in due fasi principali:

- fase di arbitraggio: serve a selezionare un'unità, detta MASTER, che controlla il bus durante l'operazione
- fase di trasferimento vero e proprio, durante la quale avvengono le seguenti operazioni
  - il MASTER seleziona un'altra unità, detta SLAVE, con la quale operare
  - il MASTER indica la direzione del trasferimento:

    - lettura (dallo SLAVE verso il MASTER)
    - scrittura (dal MASTER verso lo SLAVE)
  - viene effettuato il vero e proprio trasferimento di unità di informazione tra MASTER e SLAVE

Le unità che svolgono il ruolo di Master e di Slave sono fissate per ogni singola operazione di trasferimento del bus, ma possono variare tra operazioni diverse

Unità funzionali che possono diventare MASTER

- Nei sistemi multiprocessore le **diverse CPU** possono tutte diventare **MASTER**
- In determinate circostanze anche altre unità funzionali possono assumere il ruolo di **MASTER**

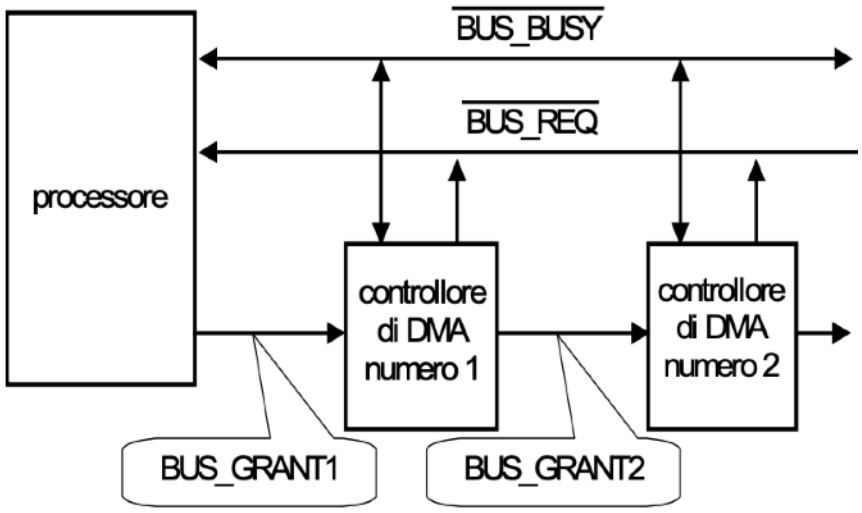
Master	Slave	Esempio
CPU	Memoria	Prelievo istruzioni e lettura/scrittura dati
CPU	Unità di I/O	Ricezione/invio dati da/a un'unità di I/O
CPU	Coprocessore	La CPU dà istruzioni al coprocessore
I/O	Memoria	Accesso diretto alla memoria (DMA)*
Coprocessore	CPU	Il coprocessore legge operandi dalla CPU

esempio - arbitraggio centralizzato in Daisy Chain

- fase di arbitraggio per il **prossimo MASTER** sovrapposta alla fase di trasferimento controllata dal **MASTER corrente**
- questo meccanismo di arbitraggio prevede:
  - un'unità funzionale apposita, che svolge la funzione di **arbitro del BUS**; talvolta la funzione di arbitro è svolta dalla CPU
  - alcune linee che collegano l'arbitro alle unità funzionali potenziali richiedenti il controllo del BUS:
    - **Bus Request** (richiesta di cessione del controllo) in ***wired or***
    - **Bus Busy** (indicazione che esiste un Master corrente che sta eseguendo un trasferimento) in ***wired or***
    - **Bus Grant** (conferma di cessione del controllo) in ***daisy chain***

- Il segnale `BUS_BUSY` asserito dall'unità Master fino a quando ha terminato di usare il bus per un trasferimento per indicare che il bus è occupato. Dato che questa linea è leggibile da tutte le unità, l'unità che diventa Prossimo Master può controllarla e attendere che il bus diventi libero prima di iniziare a utilizzarlo per il proprio trasferimento.

esempio - arbitraggio tra CPU (arbitro) e diversi controllori di DMA



Linee dei BUS

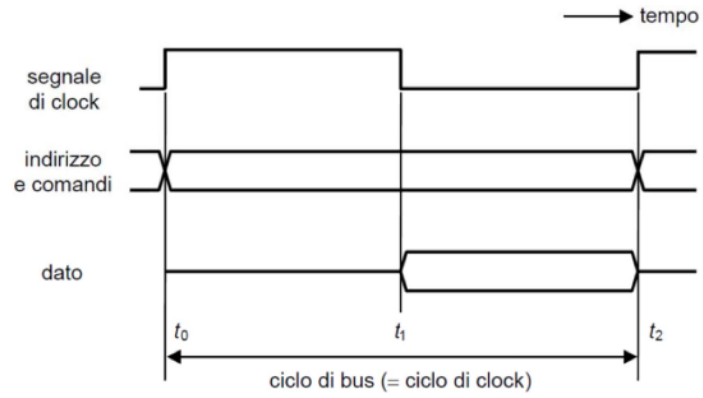
Riassumendo, un bus basato sui meccanismi descritti deve possedere le seguenti linee:

- **INT_REQ e INT_ACK**
- **BUS_BUSY, BUS_REQ e BUS_GRANT**
- **RW**
- **linee di indirizzo**
- **linee dati**

Sincronizzazione delle operazioni

- tutti i tipi di operazione richiedono di adottare schemi di **sincronizzazione**
- tali schemi sono resi complicati da numerosi problemi
  - le velocità di risposta delle varie unità differiscono
  - le complessità delle operazioni che le varie unità sono in grado di svolgere differiscono
  - i segnali di controllo del bus richiedono tempo per propagarsi da un'unità a un'altra
  - inoltre questo tempo può variare in base alla distanza (fisica e strutturale) tra le unità coinvolte
  - anche i segnali che partono nello stesso istante di tempo da una medesima sorgente possono
  - arrivare a destinazione sfasati tra loro
- per tutti questi motivi ci sono numerosi e svariati metodi di sincronizzazione, che si differenziano in base a questi elementi
  - i requisiti di velocità e di complessità logica che impongono alle unità
  - il numero di linee di BUS che richiedono
  - la velocità raggiungibile nelle diverse transazioni

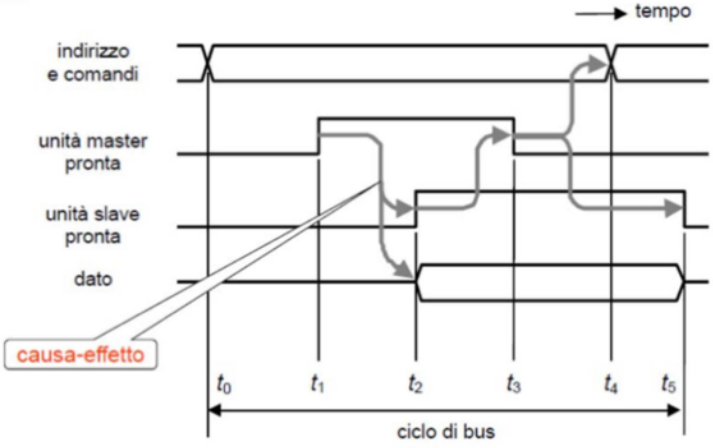
Bus Sincrono - lettura



Sul fronte di salita del clock il Master imposta i segnali di comando e indirizzamento che si propagano fino alle unità collegate sul bus.

Sul fronte di discesa del clock, le unità collegate osservano i valori presenti sul bus: l'unita che riconosce il proprio indirizzo (diventando Slave) risponde impostando sulle linee dati i valori dei segnali che si propagano al Master, che sul prossimo fronte di salita acquisisce i dati e completa la lettura.

Bus asincrono - lettura



Il principio fondamentale del meccanismo asincrono è il seguente: q**uando un'unità pone dei segnali sul bus li mantiene attivi fino a quando non riconosce un segnale di risposta che indica che i segnali sono stati ricevuti** (invece di mantenerli attivi per un tempo prefissato, determinato dal clock, come nel meccanismo sincrono). Questo principio è chiamato **Full Handshake** o **Full Interlocking**.

Bus asincrono - scrittura