



Sistemi a microprocessore

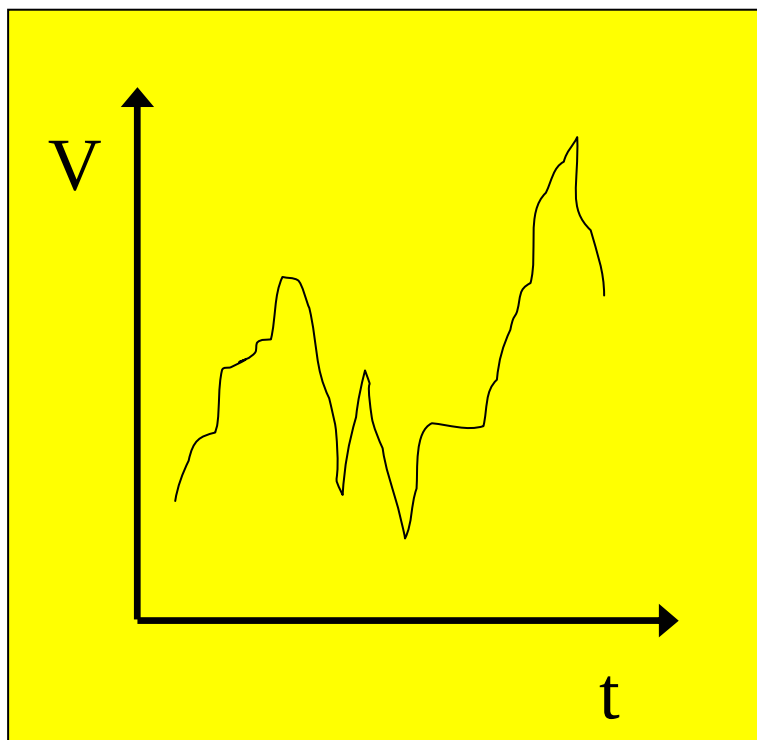
Programma:

- Segnali analogici e digitali
- Uso di segnali digitali per la rappresentazione dei numeri interi
- La memoria e la CPU
- I programmi in linguaggio macchina
- La connessione con le periferiche
- I sistemi operativi
- La rappresentazione di dati in memoria



Segnali analogici

Una grandezza fisica che fornisce informazioni sullo stato di un sistema costituisce un “segnale”. Se la grandezza fisica può assumere valori arbitrari in un dato intervallo il segnale si dice analogico.

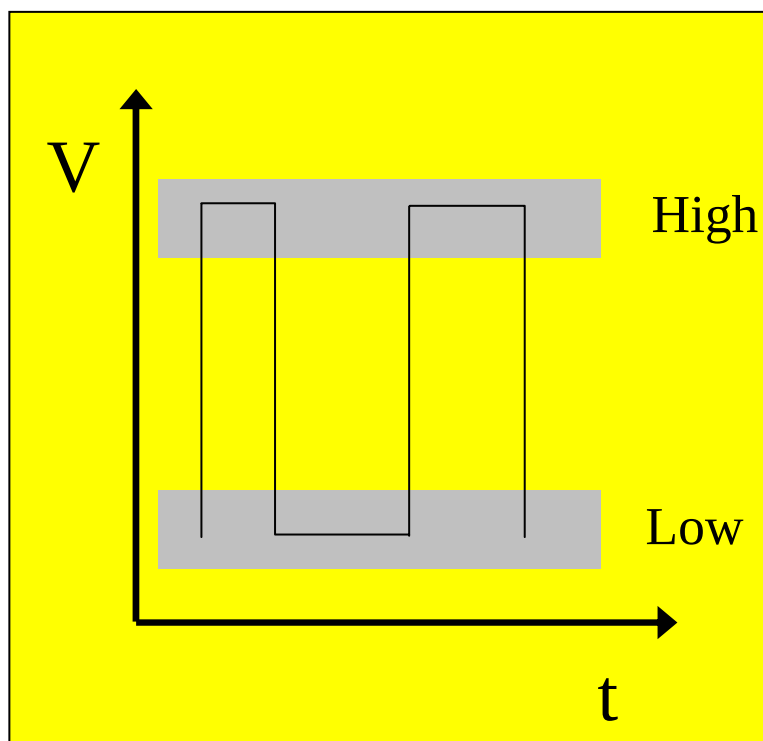


- Un segnale analogico può teoricamente codificare un numero infinito di stati di un sistema.
- La precisione con cui è possibile leggere un segnale analogico è tuttavia limitata ed influenzata da agenti esterni, per cui non è possibile distinguere in modo affidabile stati arbitrariamente “vicini”.



Segnali digitali

Un **segnale digitale** può assumere **due soli valori prestabiliti**, che vengono identificati con **0** e **1**, o con “**falso**” e “**vero**”.



- Un segnale digitale codifica **due soli** stati di un sistema, quindi servono **molti segnali** per rappresentare un sistema complesso.
- La presenza di **bande di tolleranza** attorno ai **valori nominali** di 0 e 1 fa sì che la lettura del segnale sia **riproducibile**, e quindi adatta all'implementazione di sistemi di calcolo.



Numerazione binaria

Quando scriviamo un numero intero, ad esempio 376, sottintendiamo la scrittura:

$$376 \equiv 3 \cdot 10^2 + 7 \cdot 10^1 + 6 \cdot 10^0$$

ovvero utilizziamo una rappresentazione in base 10. La stessa cosa si può fare utilizzando come base 2, il che richiede l'utilizzo di due sole cifre, 0 e 1. Ad esempio:

$$1101_2 \equiv 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13_{10}$$

La rappresentazione binaria è molto adatta all'utilizzo in sistemi digitali, in quanto ogni singola cifra o "bit" può essere associata ad un segnale digitale. N segnali digitali permettono di rappresentare i numeri interi positivi da 0 a $2^N - 1$; ad esempio, con 8 bits si va da 0 a 255, con 32 bits da 0 a 4.294.967.295



Numerazione esadecimale

La numerazione binaria è usata internamente nei calcolatori, ma è molto scomoda per gli umani; al suo posto si usa di solito la numerazione esadecimale (in base 16); servono in questo caso 16 simboli per i numeri da 0 a 15: si usano le cifre usuali (0...9) e le prime lettere dell'alfabeto (A=10, B=11,...,F=15). Ad esempio:

$$2F5A_{16} \equiv 2 \cdot 16^3 + 15 \cdot 16^2 + 5 \cdot 16^1 + 10 \cdot 16^0 = 12122_{10}$$

Il vantaggio della numerazione esadecimale rispetto a quella binaria è la compattezza. Rispetto a quella decimale il fatto che, essendo 16 una potenza di 2, ogni cifra esadecimale corrisponde esattamente a quattro cifre binarie; infatti:

$$\begin{aligned} (...B_7B_6B_5B_4B_3B_2B_1)_2 &= ... + B_7 2^7 + ... + B_4 2^4 + B_3 2^3 + ... + B_0 2^0 = \\ &... + \underbrace{(B_7 2^3 + ... + B_4 2^0)}_{E_0} 16^1 + \underbrace{(B_3 2^3 + ... + B_0 2^0)}_{E_1} 16^0 = \\ &... + E_1 16^1 + E_0 16^0 = (E_0 E_1)_{16} \end{aligned}$$

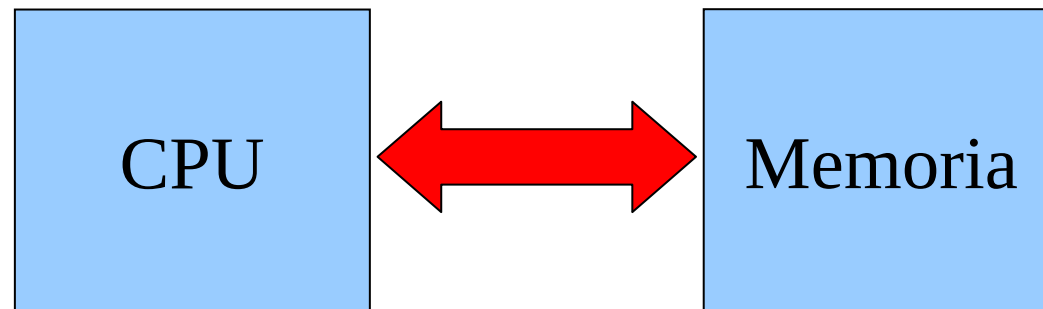
Ad esempio:

$$(1DA)_{16} = \underbrace{(0001)}_1 \underbrace{1101}_D \underbrace{1010}_A)_2$$



Sistemi a microprocessore

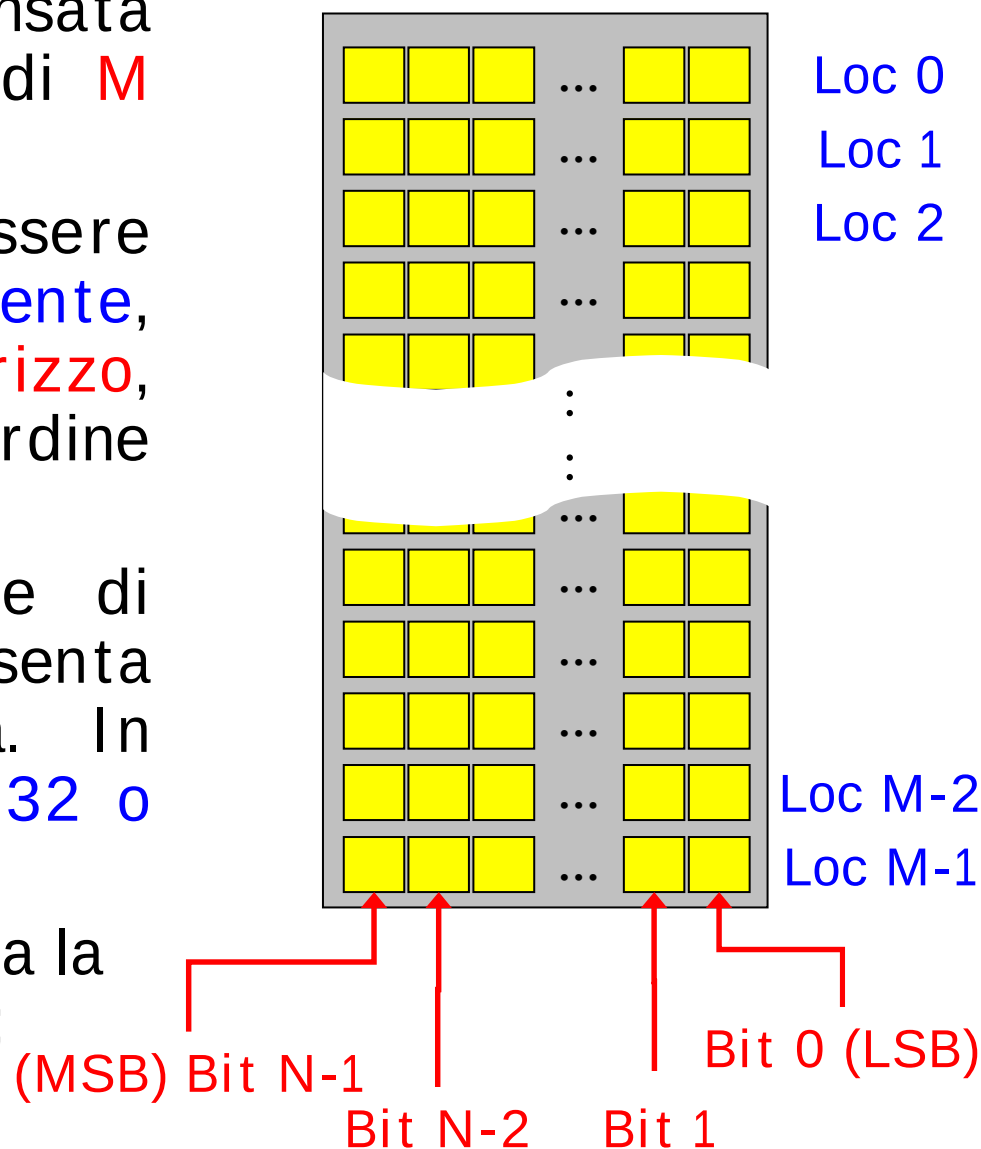
- Un generico sistema a microprocessore (μP) è composto da una CPU (Central Processing Unit) e da una memoria
- CPU e memoria possono scambiare informazioni tramite un canale di comunicazione bidirezionale.
- La memoria contiene i dati da elaborare e riceve i risultati dell'elaborazione. La CPU è la macchina che esegue le operazioni di manipolazione dei dati.





La memoria (RAM)

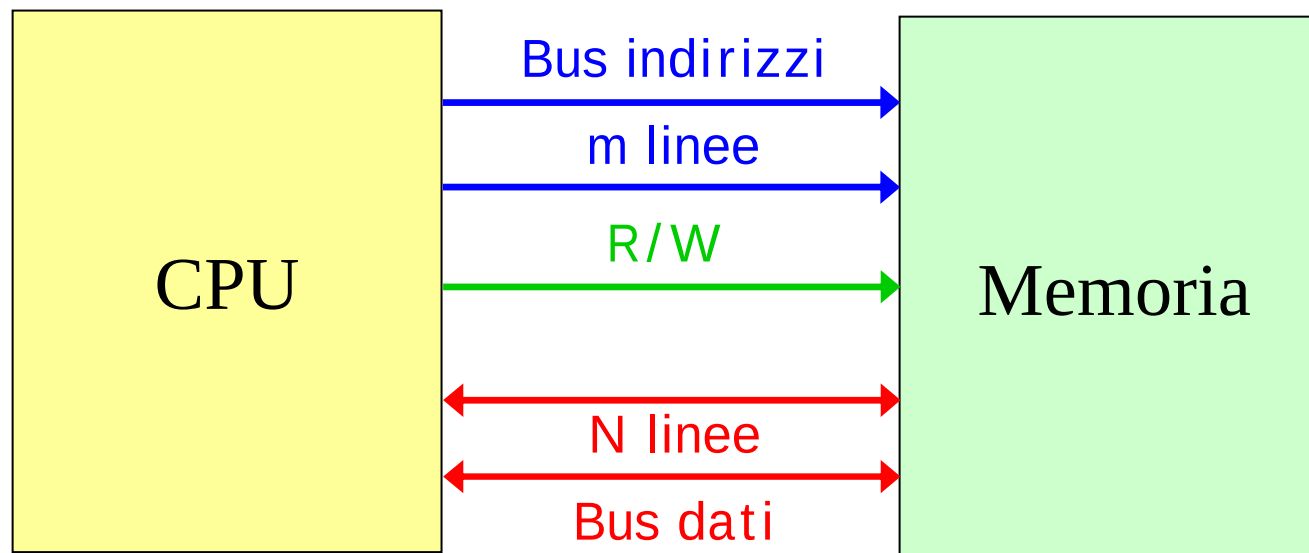
- La memoria può essere pensata come un insieme ordinato di **M** celle, ciascuna lunga **N** bits.
- Ogni singola cella può essere **scritta o letta individualmente**, specificando il suo **indirizzo**, ovvero il suo numero d'ordine nella sequenza.
- Una cella prende il nome di **parola**; il numero **N** rappresenta la **lunghezza** della parola. In genere si usano parole da **32 o 64 bits**.
- Il numero **M** rappresenta la **dimensione** della memoria; in genere, $M \approx 10^6 - 10^8$.





Connessione CPU-Memoria

- Esiste un **bus bidirezionale a N linee** che consente di trasferire **una parola** dalla memoria alla CPU (**lettura**) o viceversa (**scrittura**) (**bus dati**).
- C'è una **singola linea** che va dalla CPU alla memoria che indica se la CPU vuole leggere o scrivere (**R/W**).
- C'è un **bus monodirezionale a m linee** ($M=2^m$) che consente alla CPU di specificare quale locazione di memoria, tra le M disponibili, usare per l'operazione in corso (**bus indirizzi**).





Operazioni eseguite dalla CPU

- La CPU possiede alcune (10-100) **celle di memoria interne**, lunghe una parola, che prendono il nome di **registri** e vengono utilizzate per memorizzare i risultati intermedi delle operazioni (**l'accesso ai registri è più rapido dell'accesso alla memoria**).
- La CPU può **trasferire parole** dalla **memoria** ai **registri** o viceversa.
- La CPU può eseguire **operazioni aritmetiche** (+, -, x, :, **confronto**) tra numeri interi (rappresentati in complemento a due) **nei registri o in memoria**.
- Molte CPU possono eseguire operazioni anche su rappresentazioni in **virgola mobile**.



Esecuzione di un programma

La CPU utilizza la memoria anche per leggere le operazioni da eseguire. Ogni possibile operazione possiede un codice numerico che può essere scritto in una parola della memoria.

Al momento dell'accensione la CPU comincia a leggere l'istruzione scritta in una locazione prestabilita e la esegue. Quindi legge l'istruzione scritta nella locazione di memoria successiva e così via. La sequenza di istruzioni si dice "programma".

Una istruzione è di solito lunga più di una parola, in quanto il codice operativo deve essere seguito da alcuni parametri; ad esempio, l'istruzione "copia il contenuto della locazione di memoria X nel registro numero 7" potrebbe essere implementata così:





Esecuzione di un programma

- La CPU possiede uno speciale **registro** di **n** bits, detto **Program Counter**, che contiene **l'indirizzo dell'ultima istruzione eseguita** e che viene incrementato ad ogni ciclo.
- La maggior parte delle istruzioni richiede, per essere eseguita, di leggere o scrivere un dato in memoria. Di conseguenza **il ciclo di una istruzione prevede diversi accessi alla memoria**.
- Da un punto di vista logico, la memoria può dunque essere **divisa in due parti**: una che contiene il **programma**, ed una che contiene i **dati**.
- La **codifica delle istruzioni** (il **“linguaggio macchina”**) è **specifico** di ogni tipo di CPU, per cui i programmi in linguaggio macchina **non possono essere passati da una macchina all'altra**.



Salti e subroutines

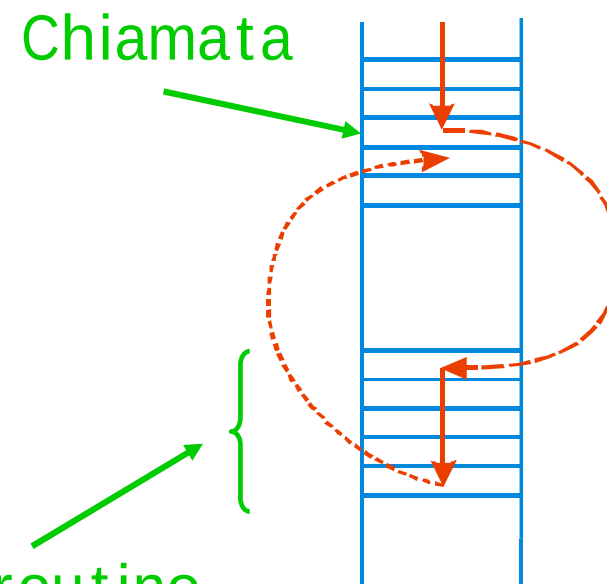
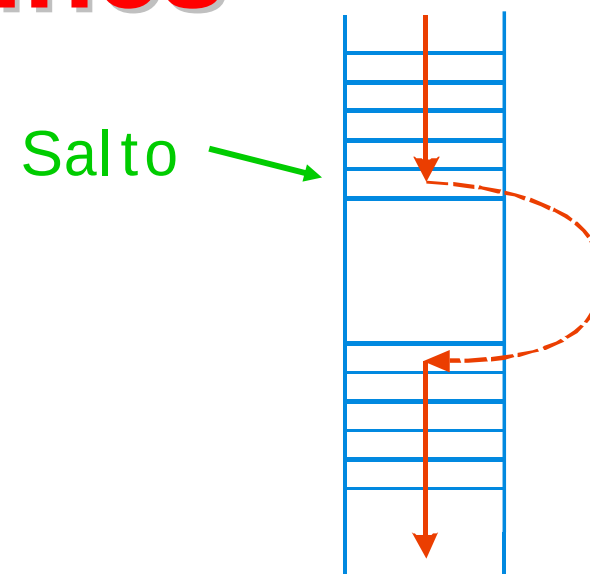
Oltre alle istruzioni che eseguono manipolazione di dati, la CPU può eseguire istruzioni che **alterano il flusso di esecuzione del programma**.

I salti possono essere **incondizionati** oppure **condizionati** al verificarsi di particolari condizioni $\Rightarrow$ CICLI DO-WHILE, BLOCCHI IF-THEN-ELSE

Una **subroutine** consiste in un blocco di istruzioni, **separate dal flusso principale**, che vengono raggiunte con un salto incondizionato.

Una particolare istruzione (**return**) consente poi di tornare al punto di partenza.

La possibilità di chiamare una subroutine implica che la CPU posseda un **metodo per ricordare la locazione di partenza**.



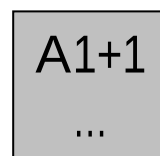
Subroutine



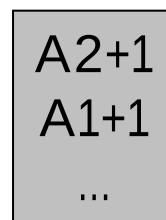
Lo stack

Lo **stack** è una memoria di tipo **LIFO** (Last In First Out) nella quale la CPU memorizza l'indirizzo dell'istruzione successiva a quella della chiamata di una **subroutine**, in modo da poter tornare indietro al momento opportuno.

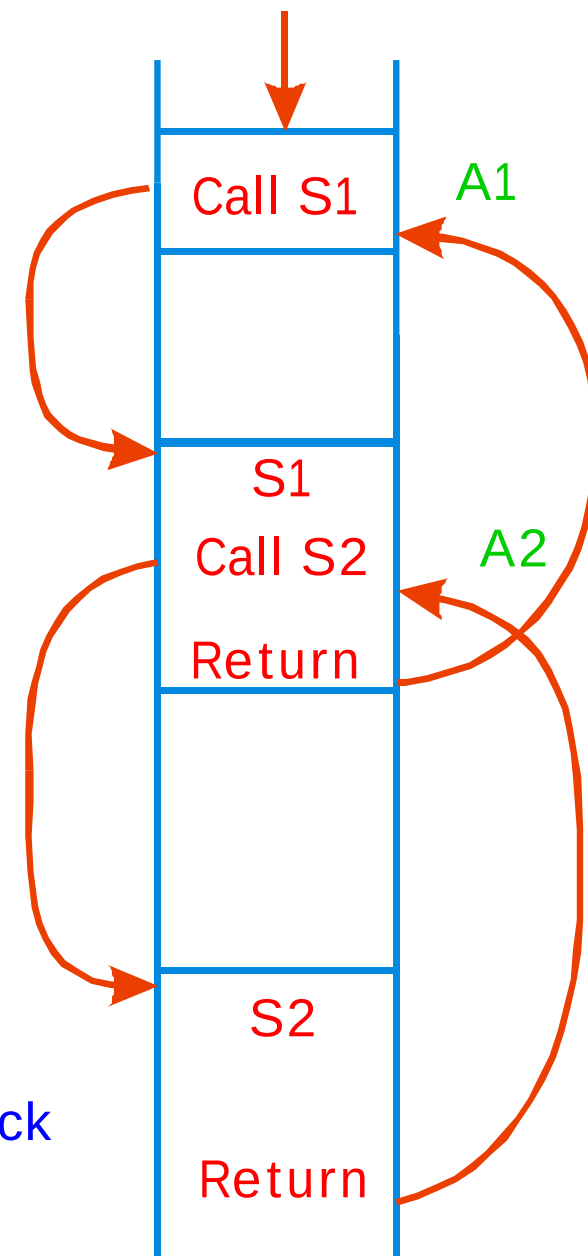
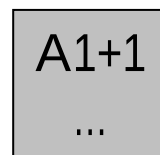
A1+1 nello stack



A2+1 nello stack



A2+1 letto dallo stack



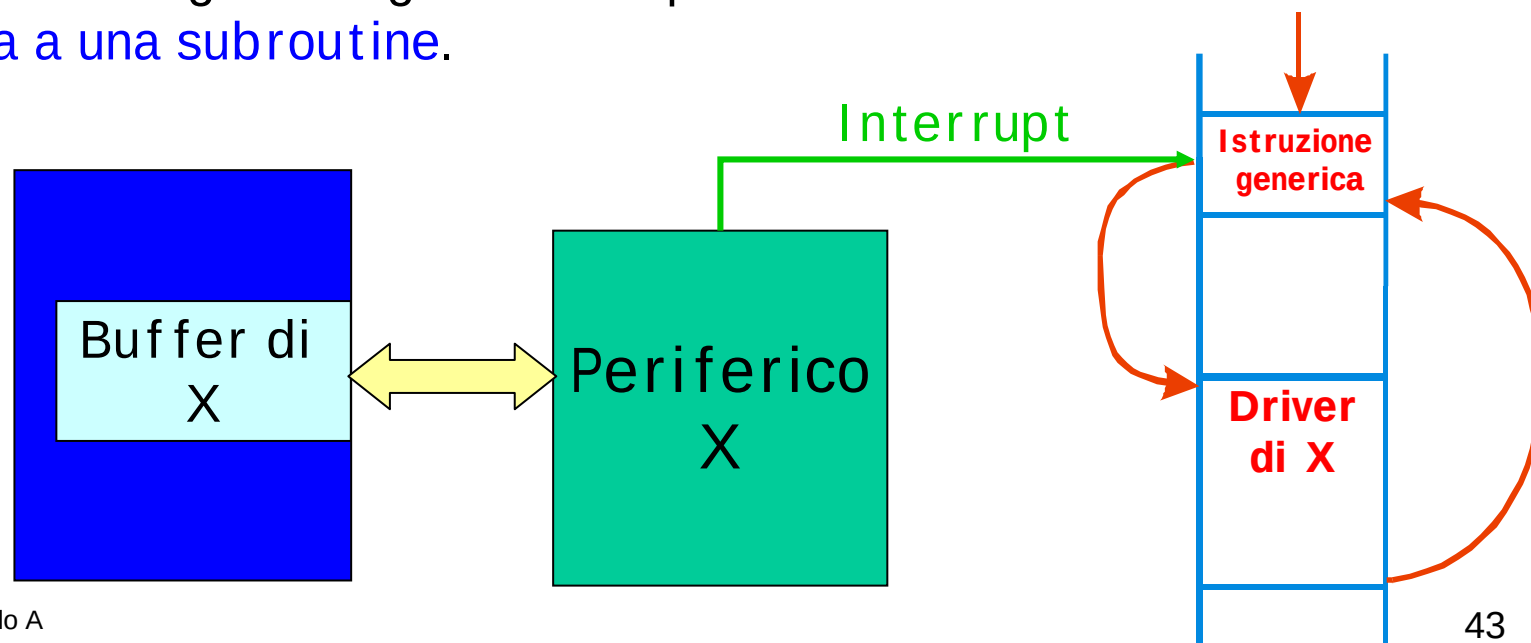


Gestione delle periferiche

Ogni **dispositivo** che deve comunicare con la CPU (dischi, tastiere, monitors, dispositivi di rete...) possiede una **zona riservata della memoria** nella quale può scrivere e/o leggere dati da scambiare con la CPU (**I/O buffer**).

Per la CPU quindi, **l'accesso ai periferici** consiste in un accesso alla memoria.

Eventi particolari (la fine di una elaborazione, la presenza di dati da leggere) vengono **segnalati** dai periferici per mezzo di un **interrupt**. Si tratta di un segnale digitale che produce il **medesimo effetto** di una chiamata a una subroutine.





Sistemi μ P: riassunto

- Un sistema μ P è composto da CPU, memoria e periferici.
- La CPU è l'unità di elaborazione; la memoria è un insieme ordinato di parole che possono contenere dati scritti in forma binaria.
- La CPU legge dalla memoria le istruzioni che deve eseguire (programma) e i dati su cui operare.
- La CPU è capace di eseguire operazioni aritmetiche (sia intere che in virgola mobile) sui dati.
- La CPU, oltre ad eseguire sequenze lineari di istruzioni, può eseguire salti e chiamate a subroutine.
- L'accesso del sistema a dispositivi esterni (dischi magnetici, tastiere, monitors, dispositivi di rete...) avviene attraverso la memoria. Esistono locazioni speciali tramite le quali avviene la comunicazione
- I periferici possono segnalare alla CPU eventi particolari (operazioni concluse, necessita di leggere dati) tramite linee di interrupt.



Sistemi μP

