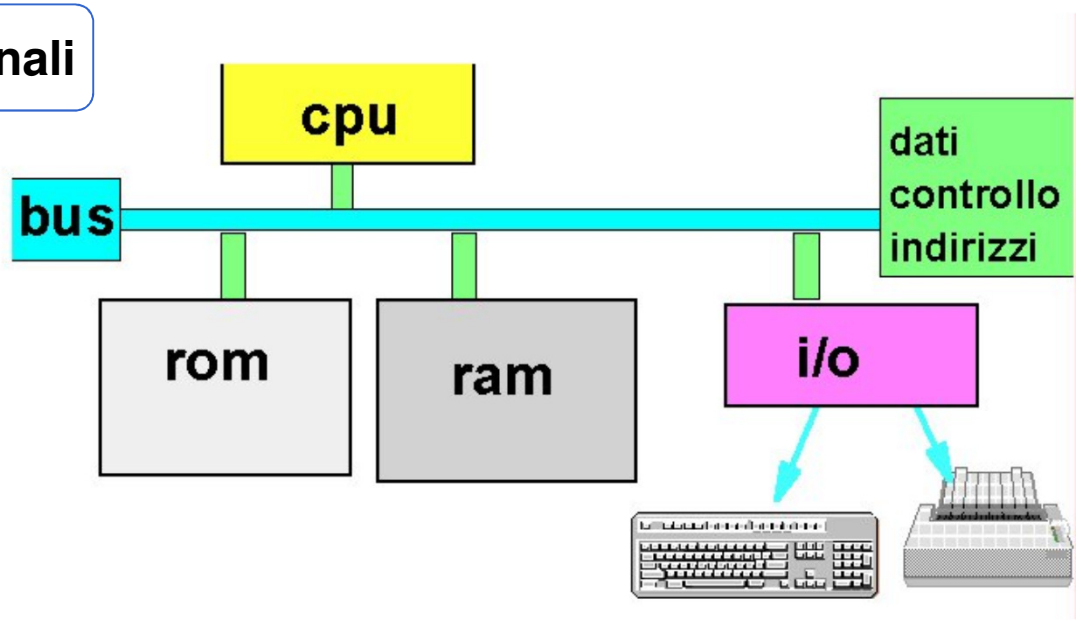
ARCHITETTURA DI UN ELABORATORE

Unità funzionali



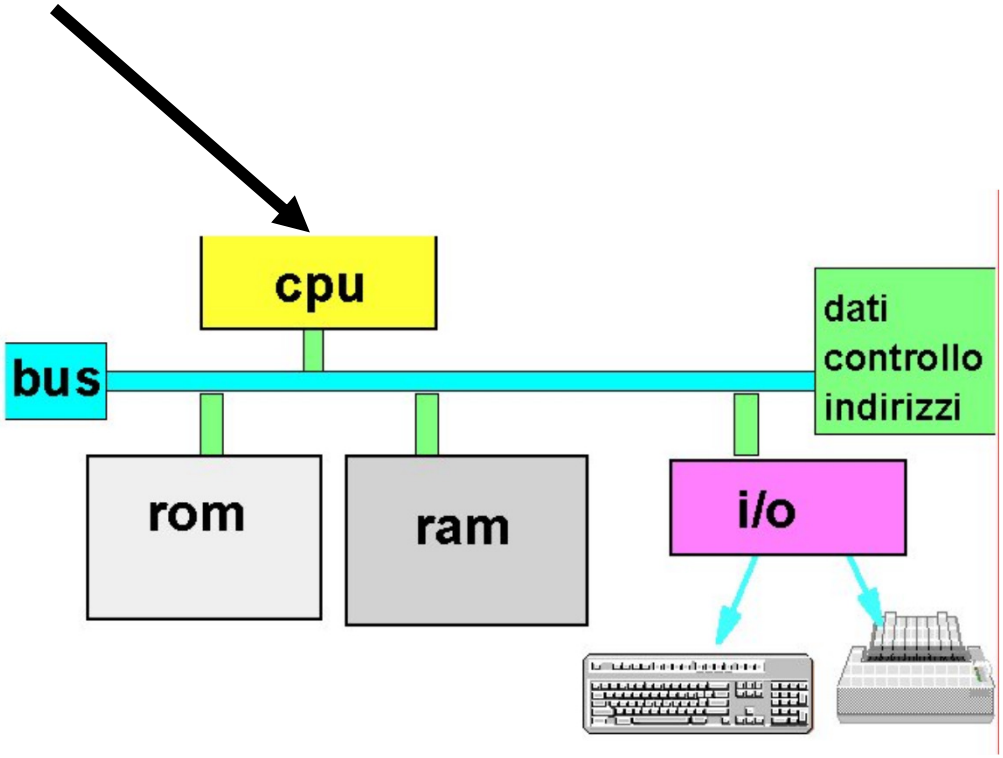
Ispirata al modello della **Macchina di Von Neumann**
(Princeton, Institute for Advanced Study, anni ’40)
Macchina di Von Neumann:

- Non distingueva fra RAM e ROM
- Non aveva un bus ma collegamenti punto-punto

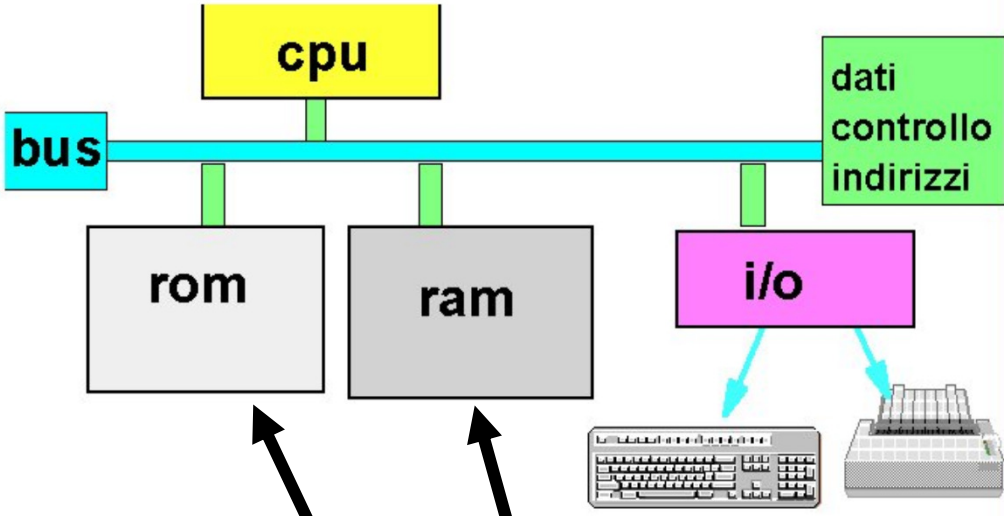
HARDWARE

CPU (Central Processing Unit), o **Processore**

CPU: svolge le elaborazioni e il trasferimento dei dati, cioè *esegue i programmi*



HARDWARE



RAM & ROM

- Dimensioni relativamente limitate
- Accesso molto rapido

RAM (***Random Access Memory***), e
ROM (***Read Only Memory***)
Insieme formano la *Memoria centrale*

HARDWARE

ATTENZIONE

RAM è volatile (perde il suo contenuto quando si spegne il calcolatore)

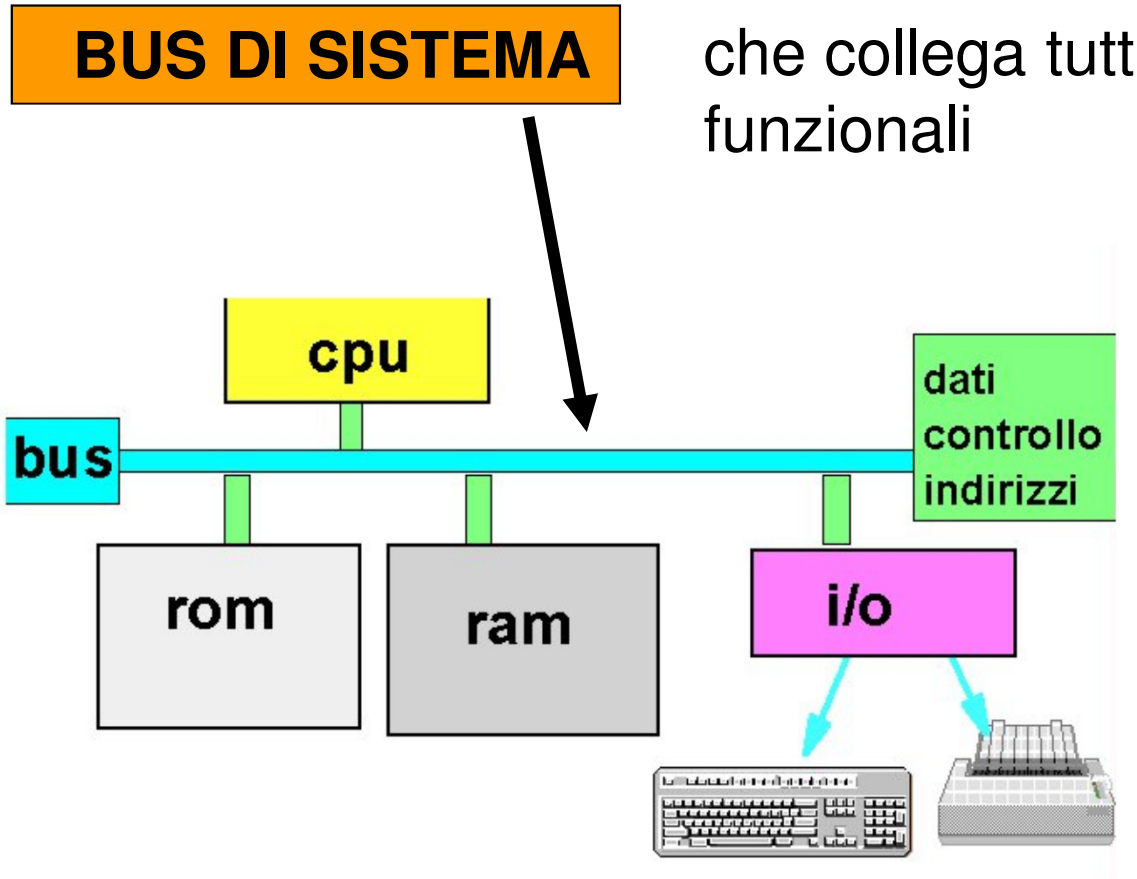
- usata per memorizzare dati e programmi

ROM è persistente (mantiene il suo contenuto quando si spegne il calcolatore) ma il suo ***contenuto è fisso e immutabile***

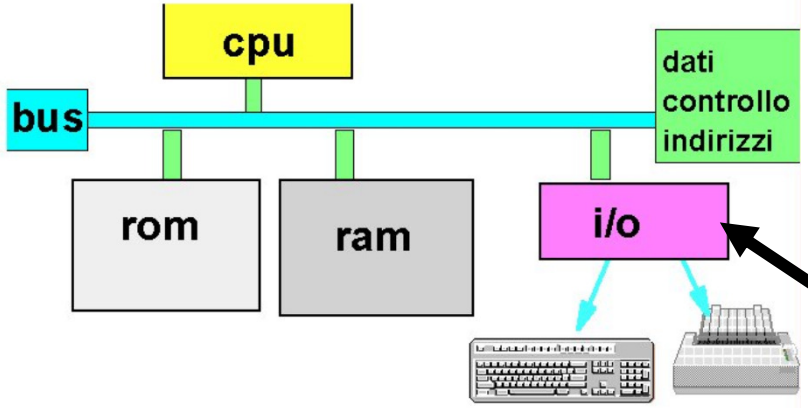
- usata per memorizzare programmi di sistema (tipicamente *firmware*)

HARDWARE

È una “linea di comunicazione” che collega tutti gli elementi funzionali



HARDWARE

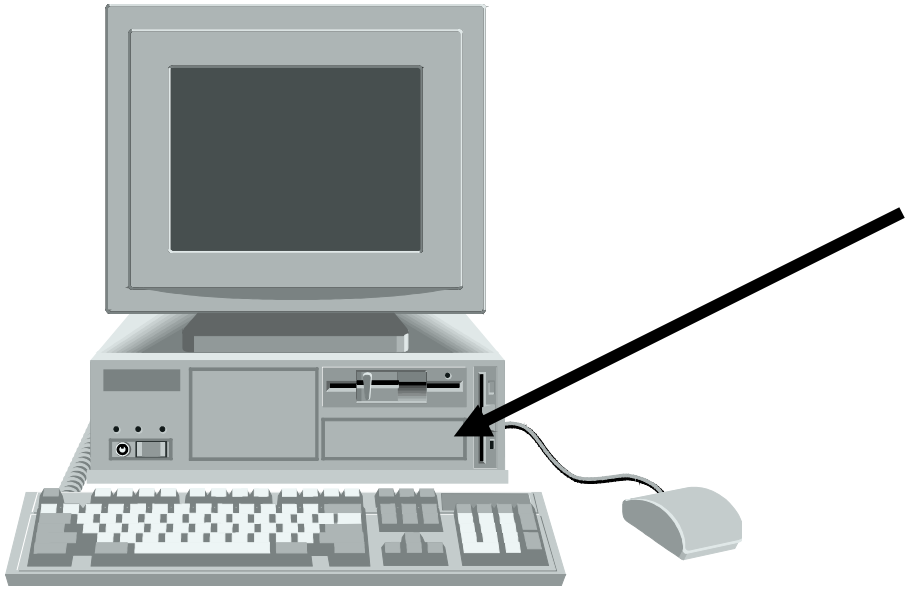


Sono usate per far comunicare il calcolatore con l'esterno (in particolare con l'utente)

UNITÀ DI INGRESSO/ USCITA (I/O)

- Tastiera e Mouse
- Video e Stampante
- Scanner
- Tavoletta grafica
- **Dispositivi di memoria di massa**
- ...

HARDWARE



MEMORIA DI MASSA

- HD
- CD
- DVD
- …
- PenDrive
- …

- memorizza **grandi quantità** di informazioni
- **persistente** (le informazioni non si perdono spegnendo la macchina)
- accesso molto meno rapido della memoria centrale (**milli**secondi contro **nano**secondi; differenza 10^6)

TECNOLOGIA DIGITALE

CPU, memoria centrale e dispositivi sono realizzati con **tecnologia elettronica digitale**

Dati e operazioni vengono codificati a partire da due valori distinti di grandezze elettriche:

- tensione alta (V_H, ad es. 5V o 3.3V)
- tensione bassa (V_L, ad es. 0V)

A tali valori vengono convenzionalmente **associate le due cifre binarie 0 e 1:**

- **logica positiva: $1 \leftrightarrow V_H$, $0 \leftrightarrow V_L$**
- logica negativa: $0 \leftrightarrow V_H$, $1 \leftrightarrow V_L$

TECNOLOGIA DIGITALE (segue)

Dati e operazioni vengono codificati tramite **sequenze di bit**

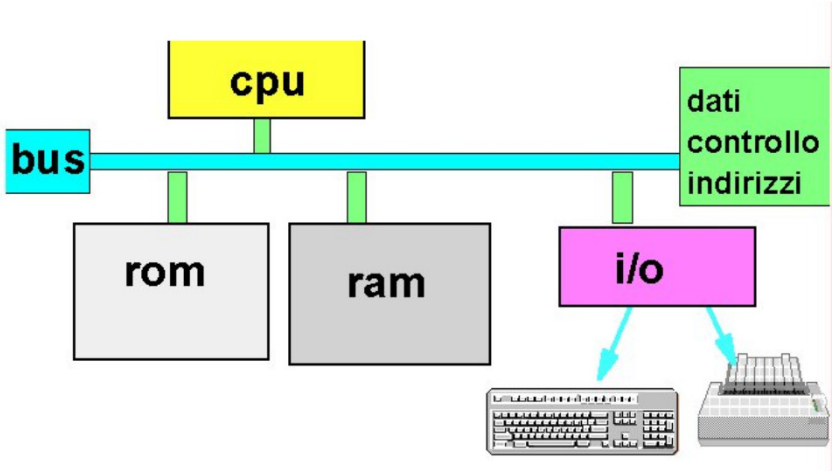
01000110101 ....

CPU è in grado di operare soltanto in aritmetica binaria, effettuando operazioni *elementari*:

- somma e differenza
- scorrimento (shift)
- ...

Lavorando direttamente sull'hardware, **l'utente è forzato a esprimere i propri comandi *al livello della macchina*, tramite *sequenze di bit***

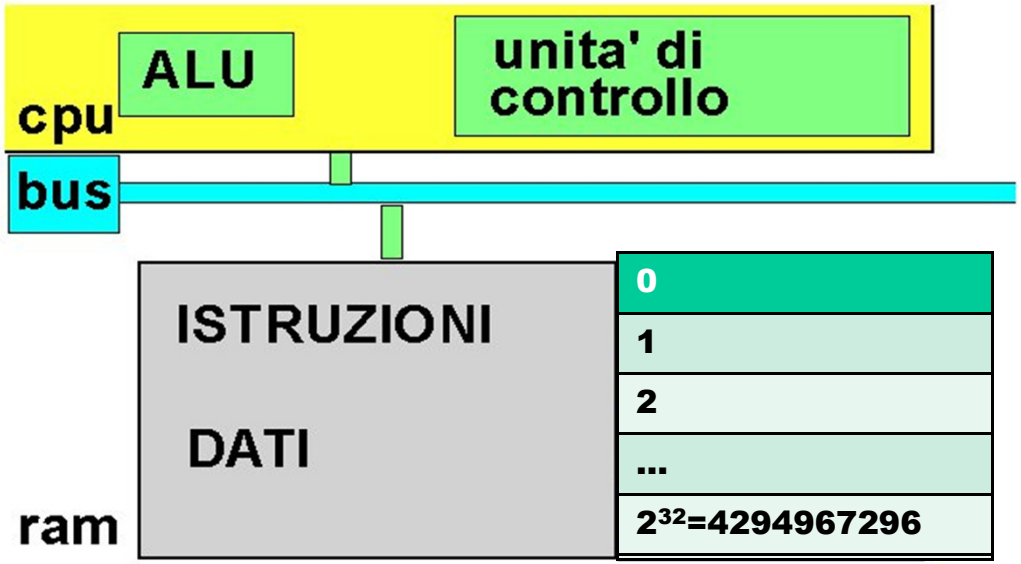
MACCHINA DI VON NEUMANN



UNITÀ FUNZIONALI fondamentali

- **Processore (CPU)**
- **Memoria Centrale (RAM & ROM)**
- **Unità di I/O (ingresso/uscita)**
- **Bus di sistema**

CPU & MEMORIA



- **ALU (Arithmetic & Logic Unit)**
- **Unità di Controllo**
- **Registri**

UNITÀ DI ELABORAZIONE (CPU)

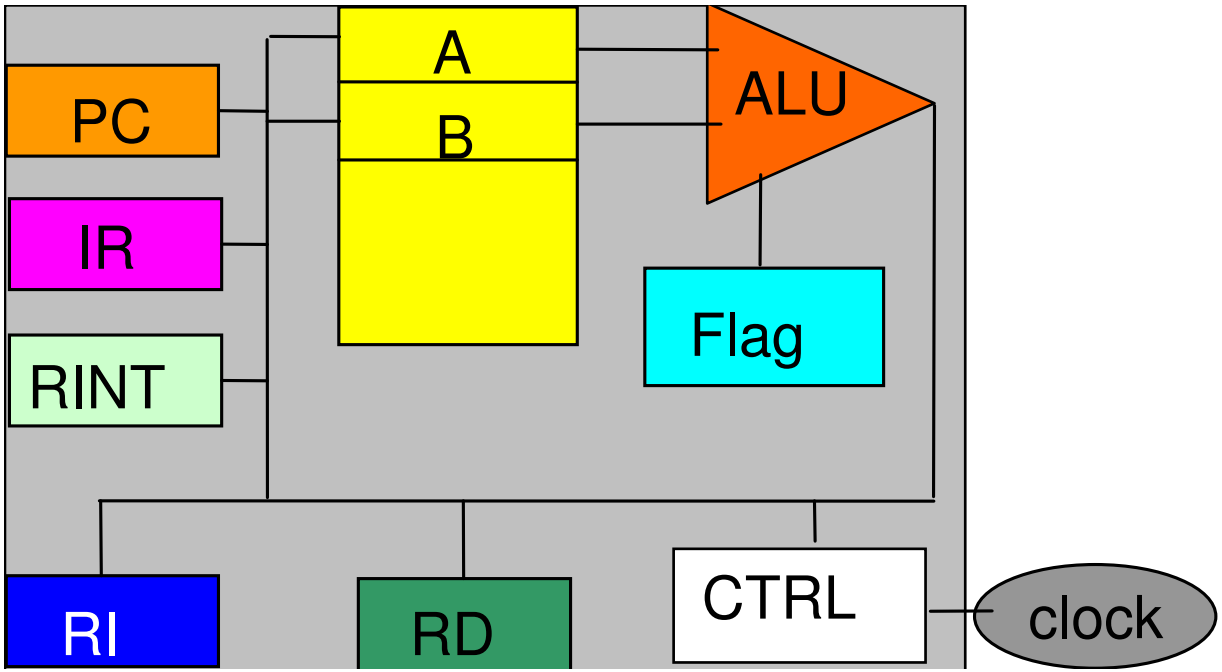
ALU (Arithmetic/Logic Unit)

Esegue le operazioni aritmetiche e logiche elementari

Unità di Controllo (*Control Unit*): ***controlla e coordina l'attività della CPU***. In particolare, controlla il trasferimento dei dati tra memoria e registri e la decodifica e l'esecuzione delle istruzioni

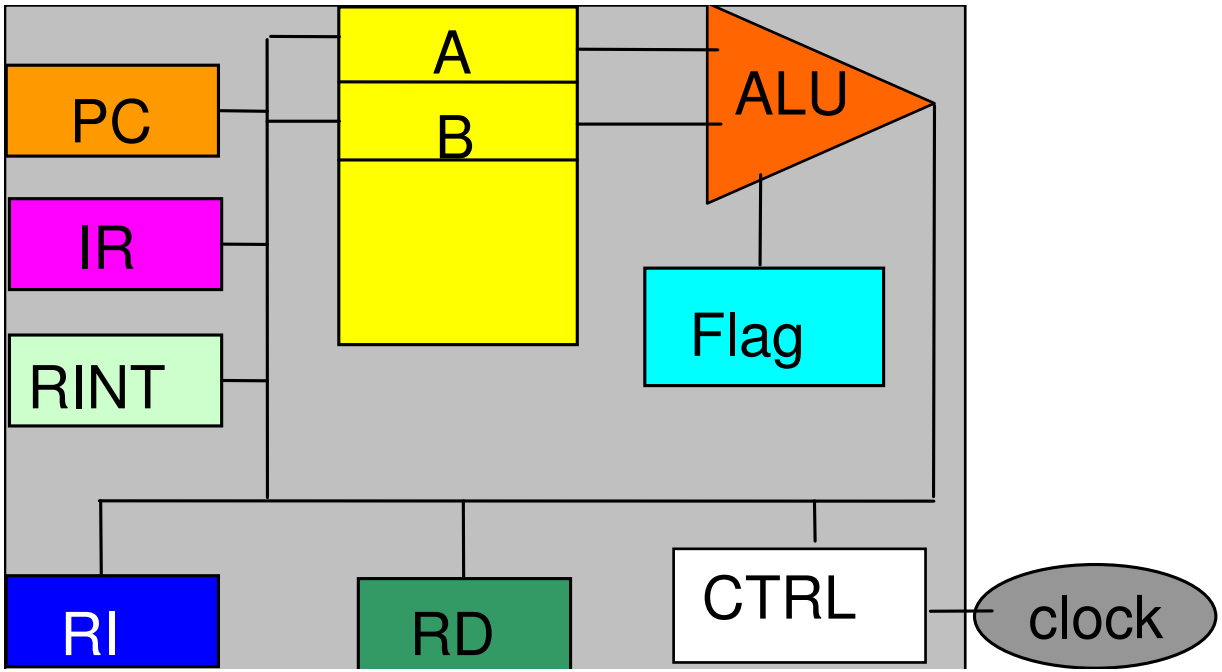
I registri sono *locazioni* usate per memorizzare dati, istruzioni, o indirizzi ***all'interno della CPU***. L'accesso ai registri è *molto veloce*

UNITÀ DI ELABORAZIONE (CPU)



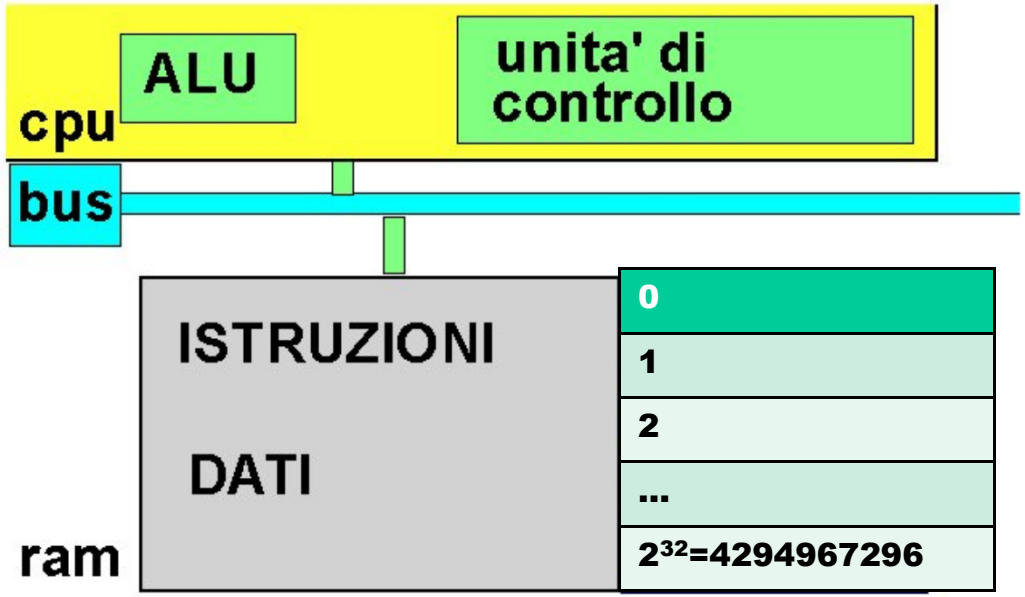
Il clock dà la base dei tempi necessaria per mantenere il sincronismo fra le operazioni

UNITÀ DI ELABORAZIONE (CPU)



I registri (qui **A**, **B**, **PC**, **Flag**,…) sono *locazioni* usate per memorizzare dati, istruzioni, o indirizzi ***all'interno della CPU***. Altissima velocità di accesso

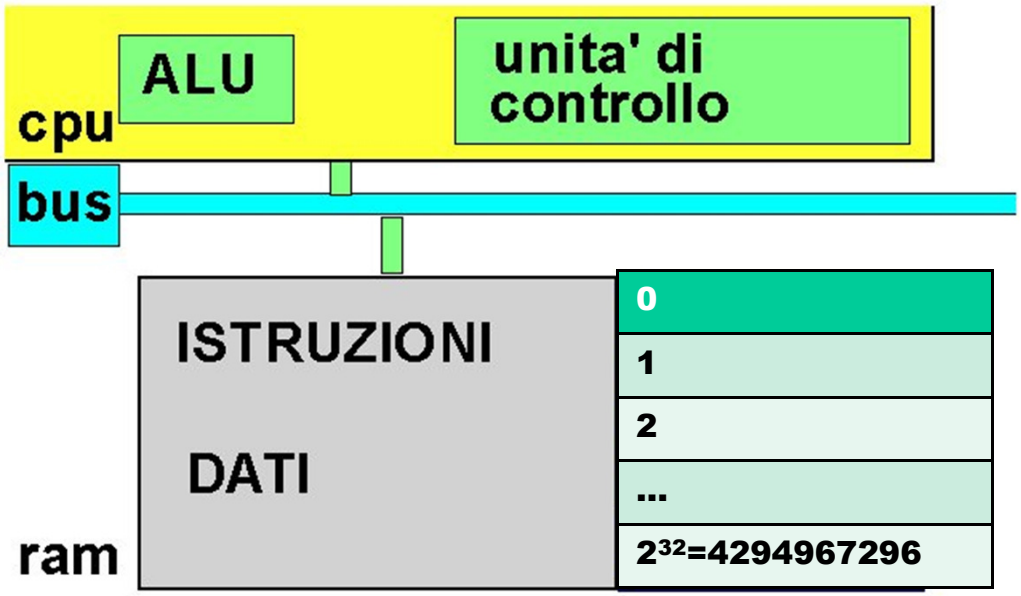
UNITÀ DI ELABORAZIONE (CPU)



La memoria centrale è una **collezione di celle *numerate***, che possono contenere **DATI e ISTRUZIONI**

Le istruzioni sono disposte in memoria **in *celle di indirizzo crescente***

UNITÀ DI ELABORAZIONE (CPU)



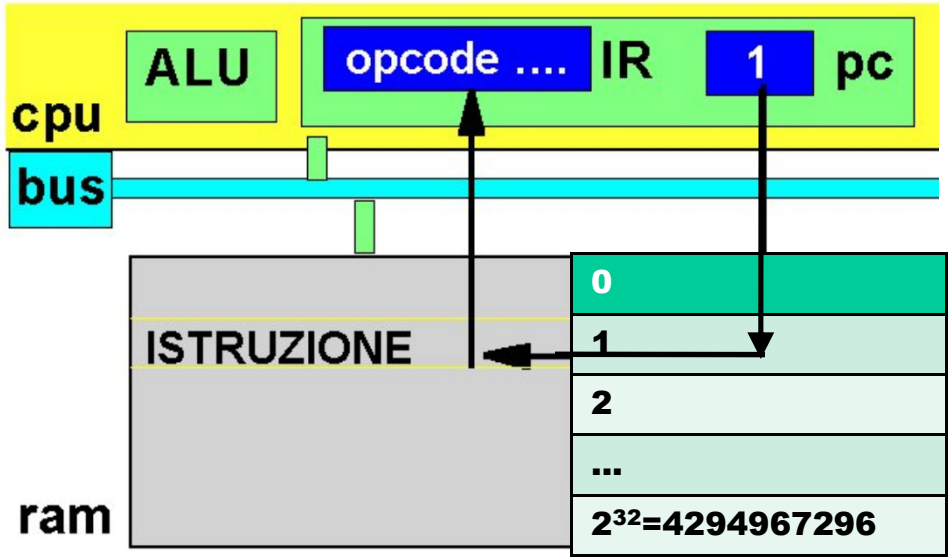
L'unità di controllo fa funzionare l'elaboratore

Da quando viene acceso a quando è spento, essa esegue in continuazione il **ciclo di *prelievo/decodifica/esecuzione*** (**fetch**/**decode**/**execute**)

IL CICLO fetch/decode/execute

FETCH

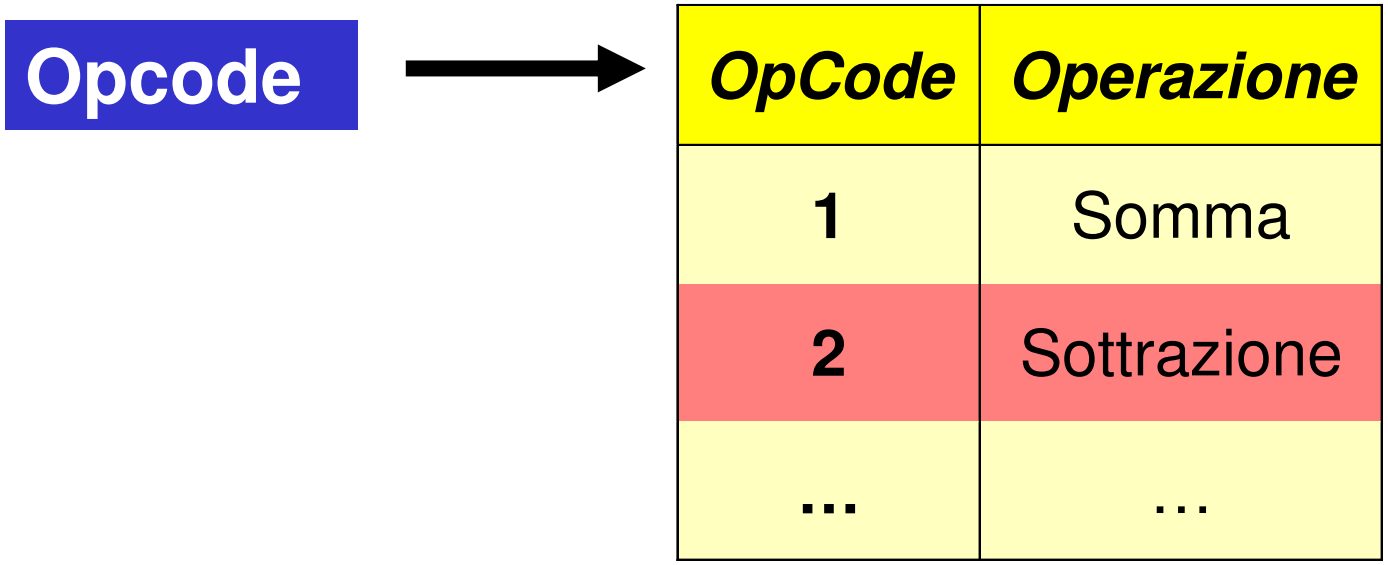
- si accede alla **prossima istruzione** (cella il cui indirizzo è contenuto nel registro **PC**) ...
- … e **la si porta dalla memoria centrale**, memorizzandola nel ***Registro Istruzioni*** **(IR)**



IL CICLO fetch/decode/execute

DECODE

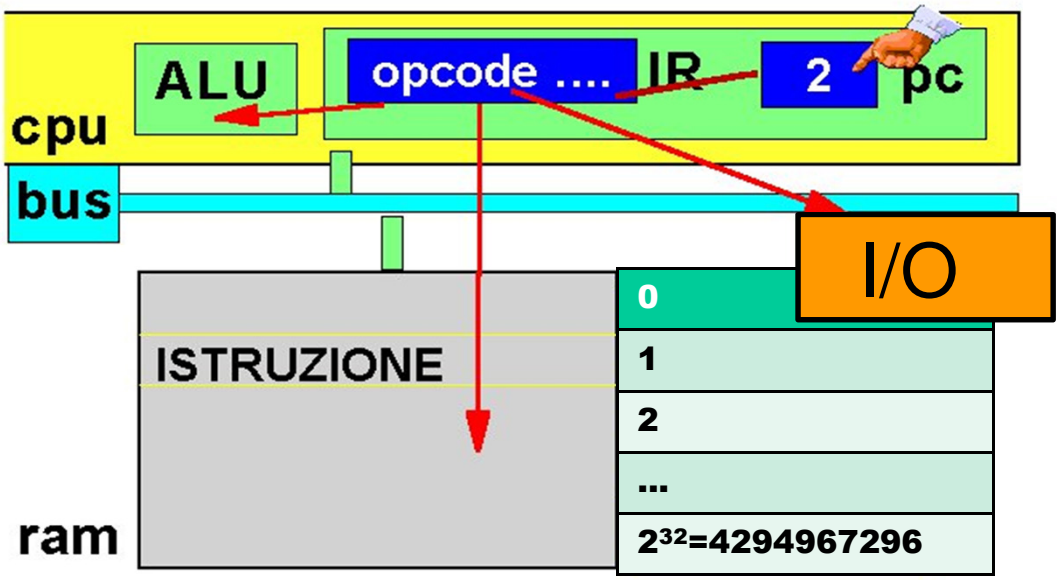
- **si decodifica il tipo dell'istruzione** in base al suo ***OpCode*** (codice operativo)



IL CICLO fetch/decode/execute

EXECUTE

- **si individuano i dati** usati dall'istruzione
- **si trasferiscono tali dati** nei registri opportuni
- **si esegue l'istruzione**



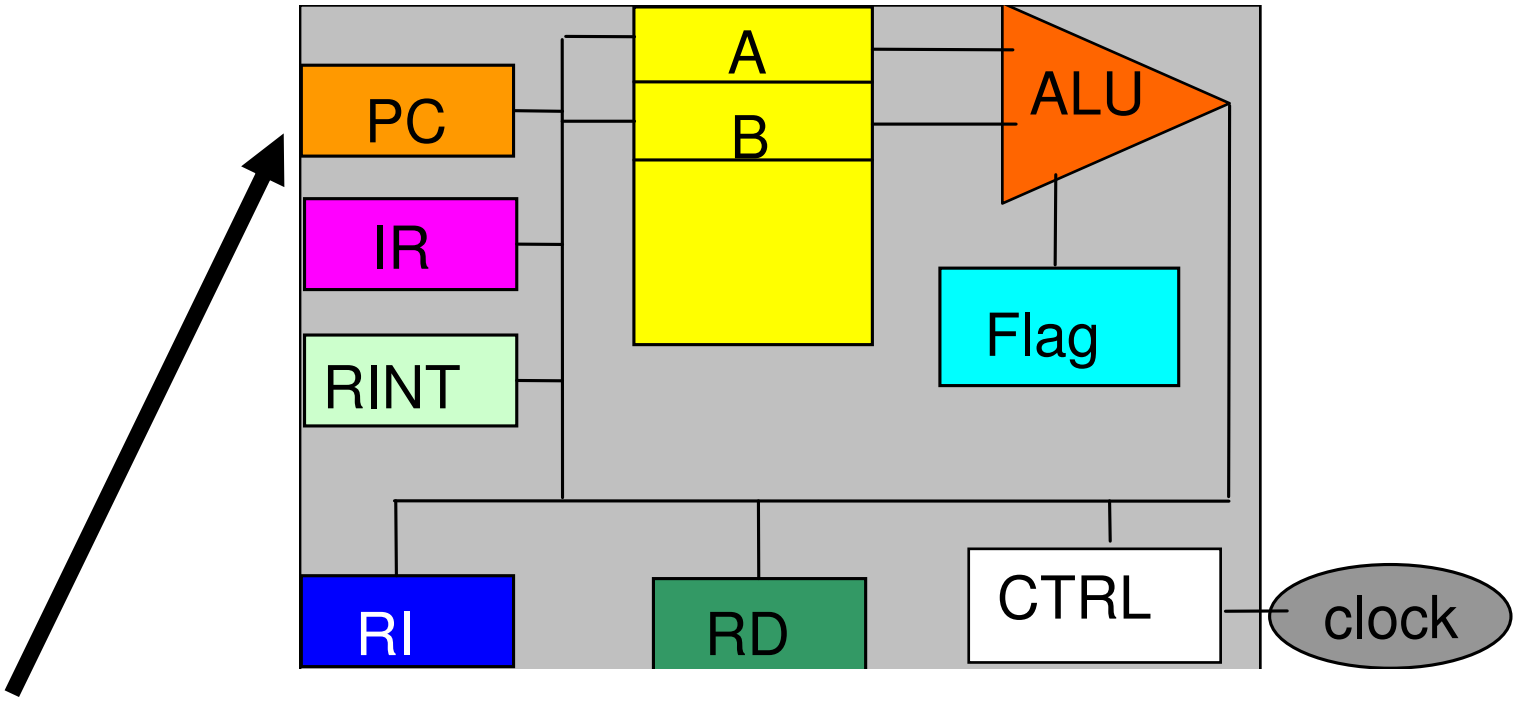
IL CICLO fetch/decode/execute

ATTENZIONE

Istruzioni particolari possono ***alterare il prelievo delle istruzioni da celle consecutive***:

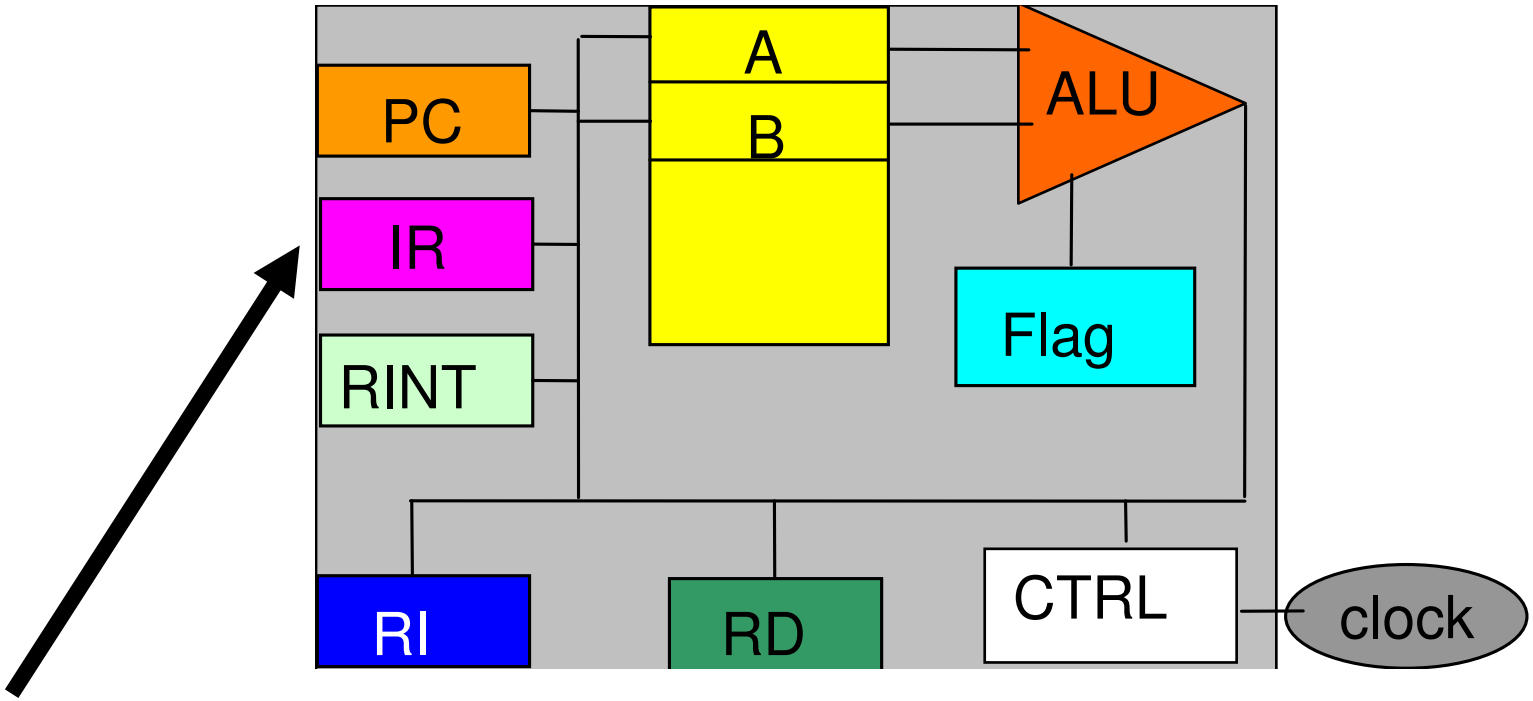
- istruzioni di **salto**
- istruzioni di **chiamata a sotto-programmi**
- istruzioni di **interruzione**

I REGISTRI



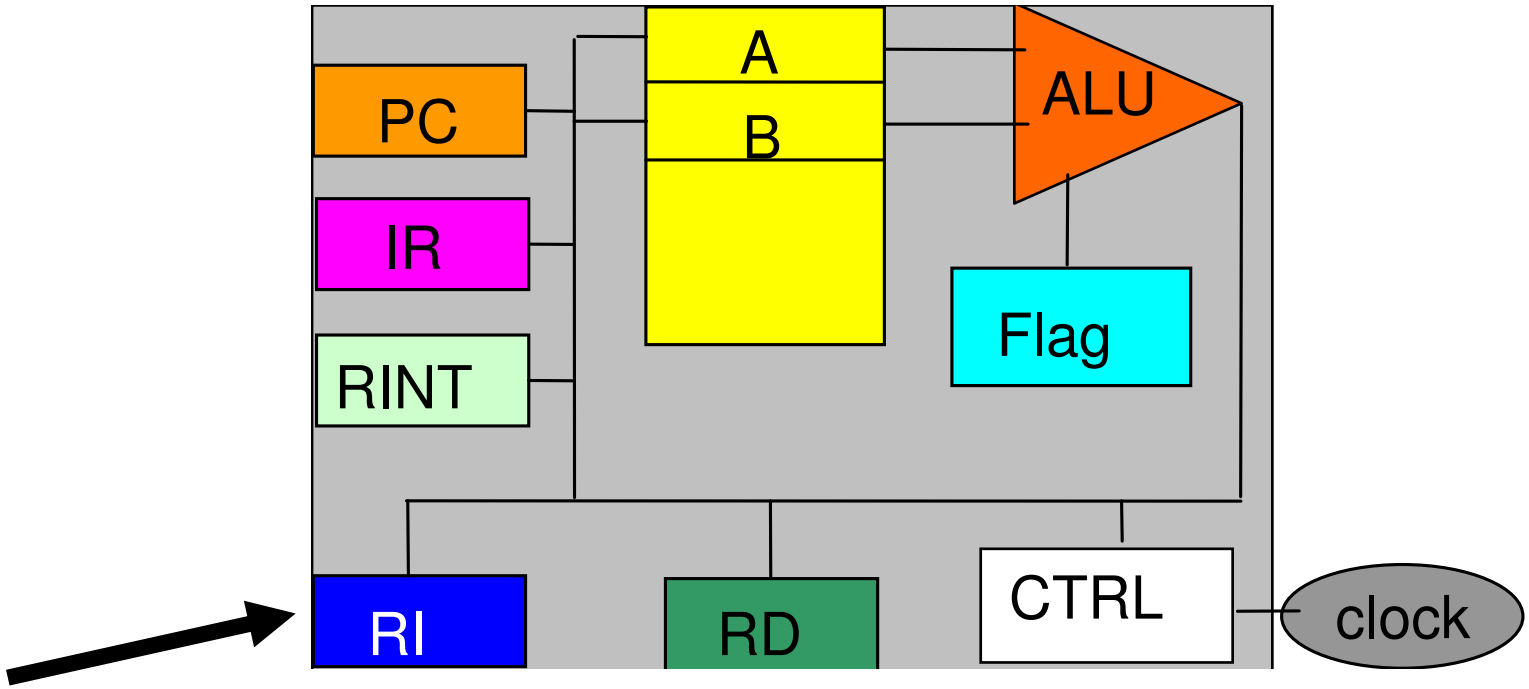
Program Counter (PC)
Indica l'indirizzo della cella di memoria che contiene la prossima istruzione da eseguire

I REGISTRI



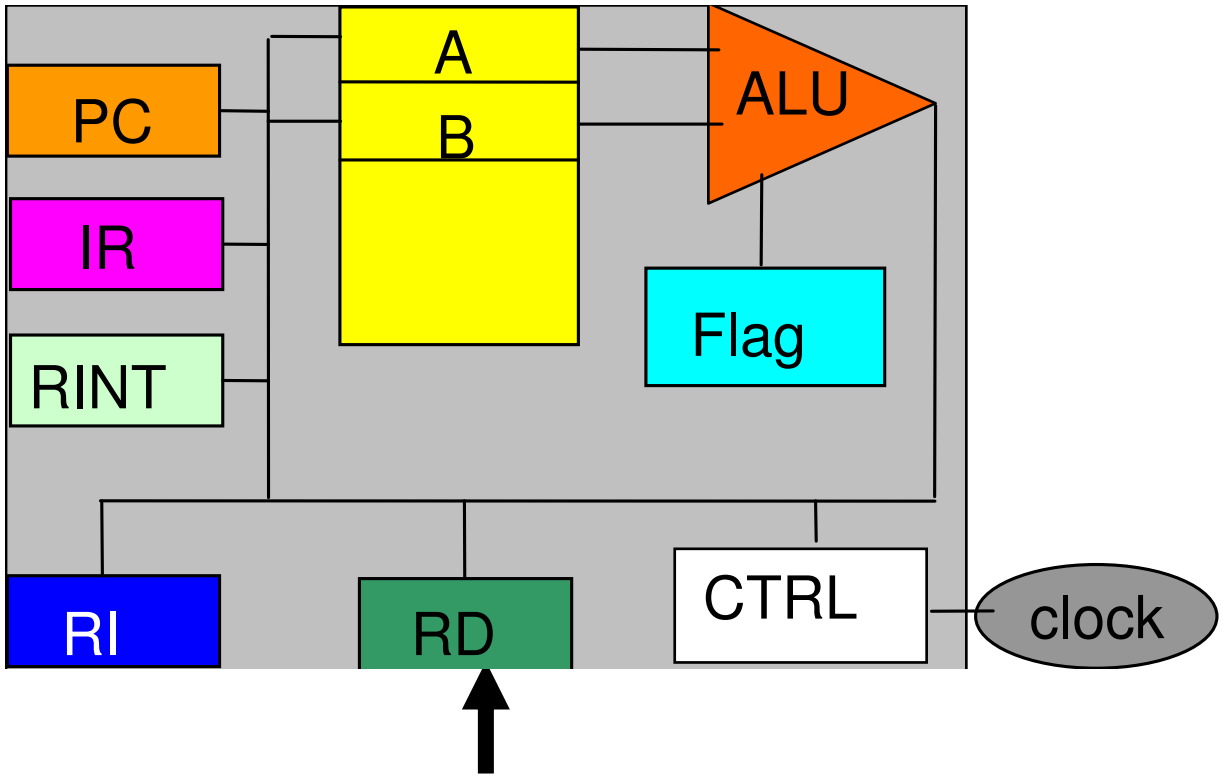
Instruction Register (IR)
Contiene la vera e propria istruzione da eseguire

I REGISTRI



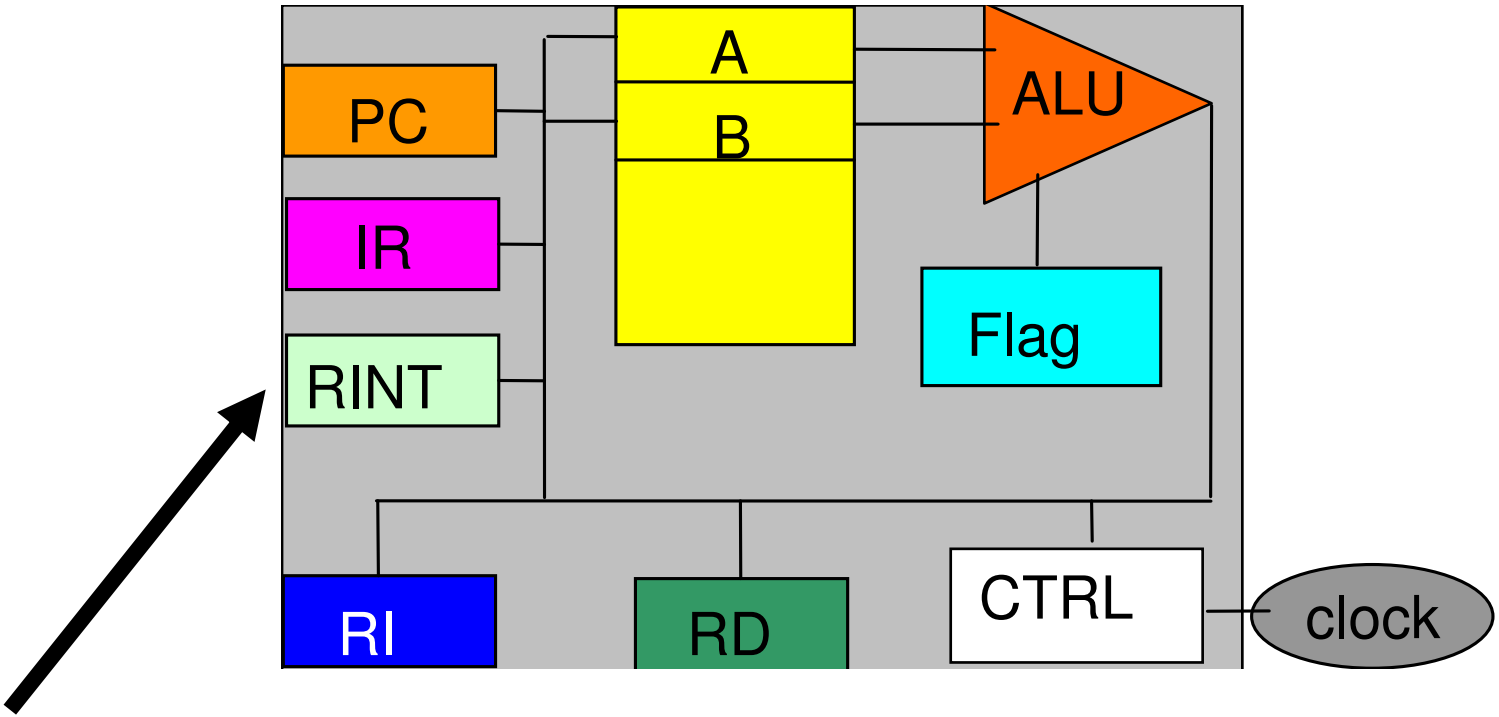
Registro Indirizzi (RI)
Contiene l'indirizzo della cella di memoria da selezionare per il trasferimento di un dato da/verso la CPU

I REGISTRI



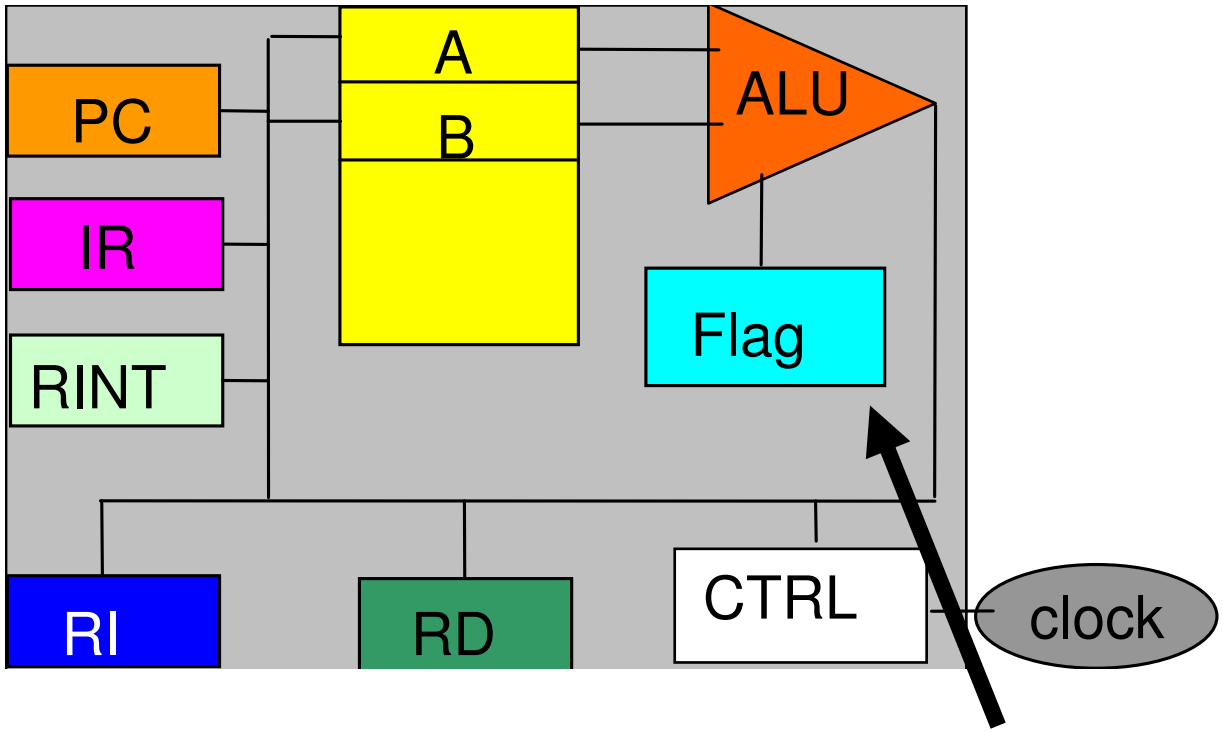
Registro Dati (RD) o *Accumulatore*
Contiene il dato attualmente oggetto di elaborazione e il risultato al termine dell'esecuzione

I REGISTRI



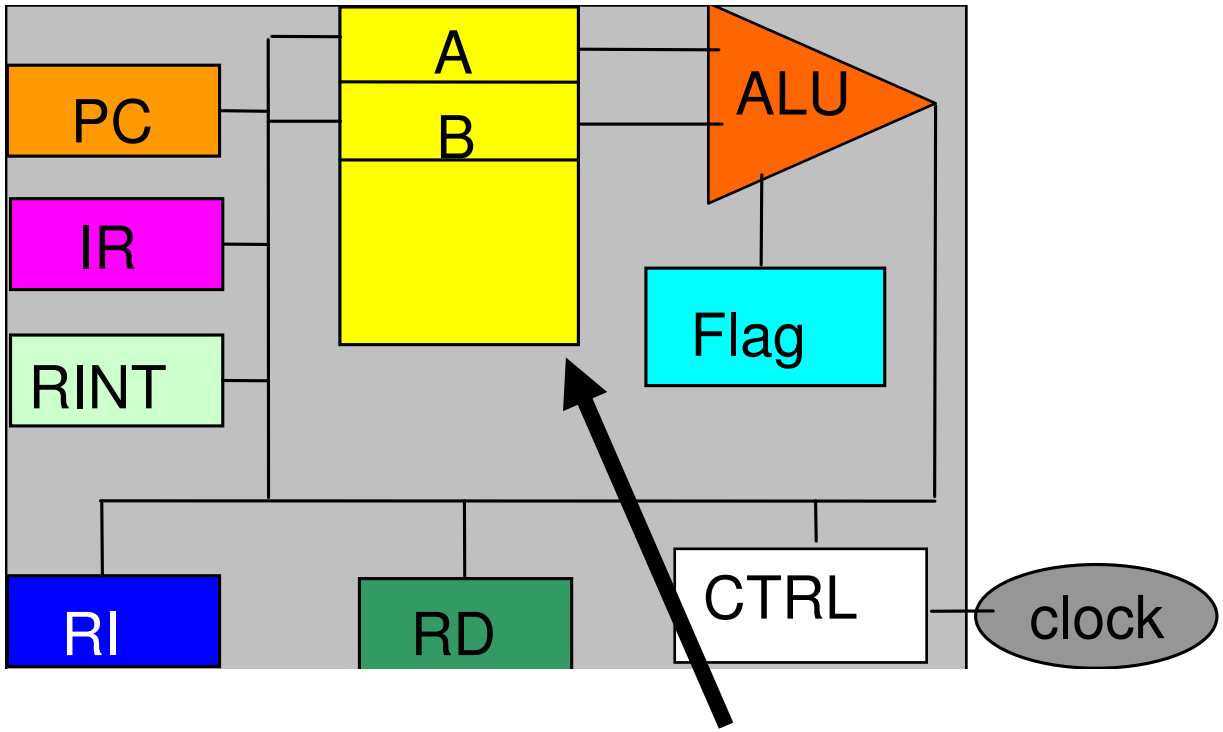
Registro Interruzioni (RINT)
Serve per scopi particolari (non discussi in questo corso, vedi Sistemi Operativi L-A)

I REGISTRI



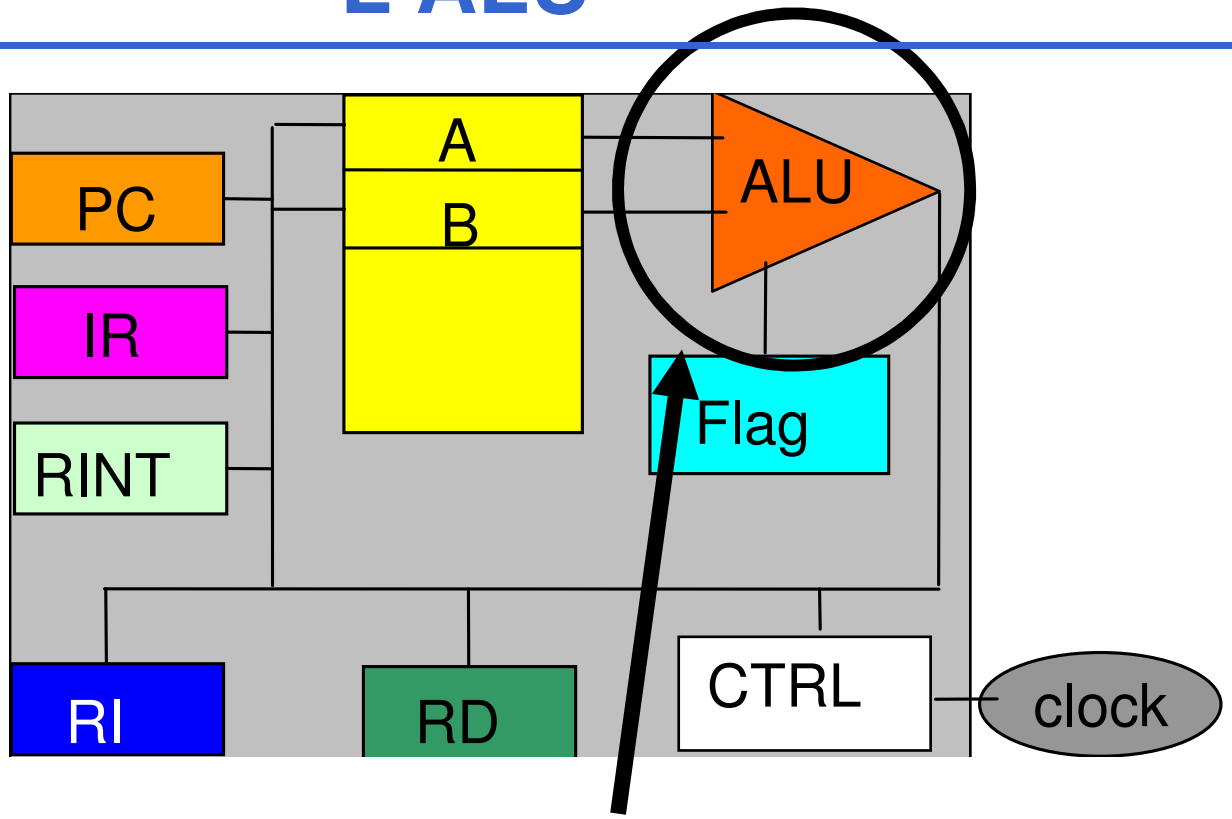
Registro dei Flag (Flag)
Ogni flag indica la presenza/assenza di una proprietà nell'ultimo risultato generato dalla ALU. Altri bit riassumono lo stato del processore

I REGISTRI



Registri di uso generale (A,B,C,...)
Sono usati per contenere ***sia dati*** (in particolare, operandi di operazioni aritmetico/logiche) ***sia indirizzi***

L'ALU



Esegue operazioni aritmetiche, logiche e confronti sui dati della memoria centrale o dei registri

Può essere semplice e molto veloce (architettura RISC) o a volte con set di istruzioni molto numerose (architettura CISC)

L'ALU (segue)

ESEMPIO SEMPLICE:

ALU in grado di eseguire **somma**, **sottrazione**, **prodotto**, **divisione** con due operandi contenuti nei registri A e B.

1. I due operandi vengono caricati nei registri A e B;
2. La ALU viene attivata da un comando inviato dalla CPU che specifica il tipo di operazione;
3. Nel registro A viene caricato il risultato dell'operazione eseguita dalla ALU;
4. Il registro FLAG riporta sui suoi bit indicazioni sul risultato dell'operazione (riporto, segno, etc.).

Alterazione di due bit nel registro **Flag**: ***carry*** (riporto) e ***sign*** (segno)

LA MEMORIA CENTRALE (1)

INDIRIZZAMENTO

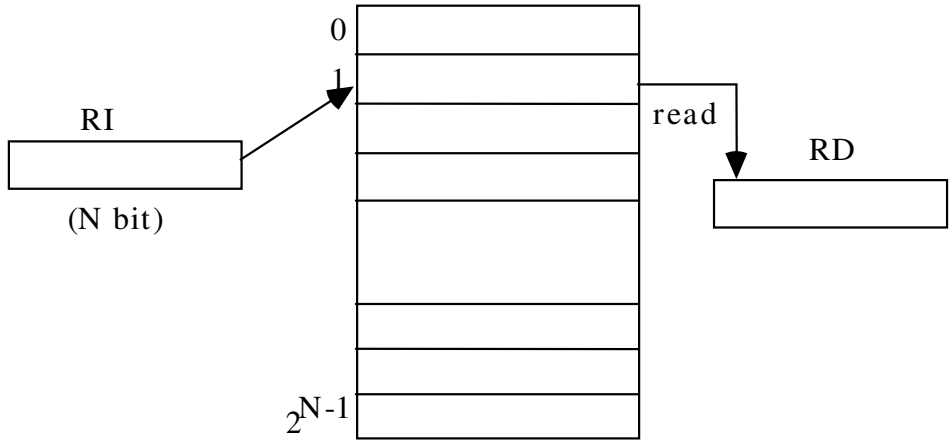
- È l'attività con cui l'elaboratore seleziona una particolare cella di memoria
- Per farlo, l'elaboratore pone l'indirizzo della cella desiderata nel Registro Indirizzi (RI)
  - **se RI è lungo *N* bit, si possono indirizzare 2^N celle di memoria** (numerate da 0 a 2^N-1)
  - esempio: N=10 $\Rightarrow$ 1024 celle
- **Oggi**, RI è lungo tipicamente **32/64 bit**
  $\rightarrow$ *32 bit:* ***SPAZIO INDIRIZZABILE di 4 GB***

  $\rightarrow$ *64 bit:* ***SPAZIO INDIRIZZABILE di 16 ExaByte =***
  17,179,869,184 GByte

LA MEMORIA CENTRALE (2)

OPERAZIONI

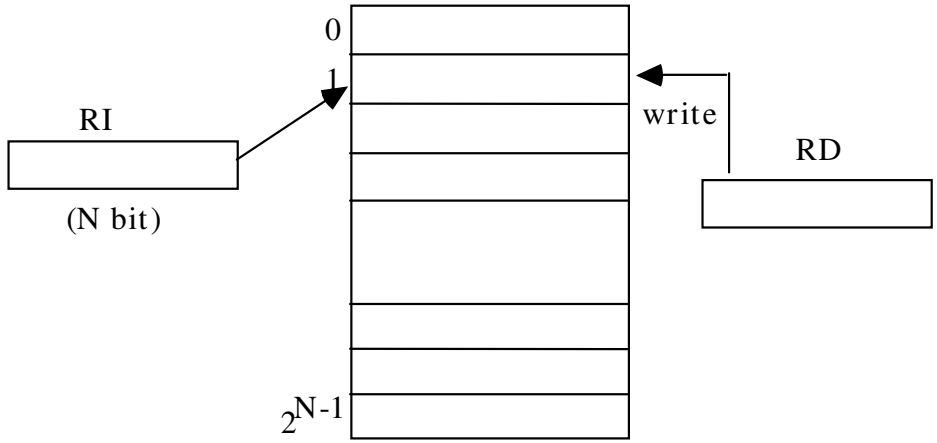
- **Lettura *(Read)***: il contenuto della cella di memoria indirizzata dal Registro Indirizzi è copiato nel Registro Dati



LA MEMORIA CENTRALE (3)

OPERAZIONI

- **Scrittura *(Write)***: il contenuto del Registro Dati è copiato nella cella di memoria indirizzata dal Registro Indirizzi



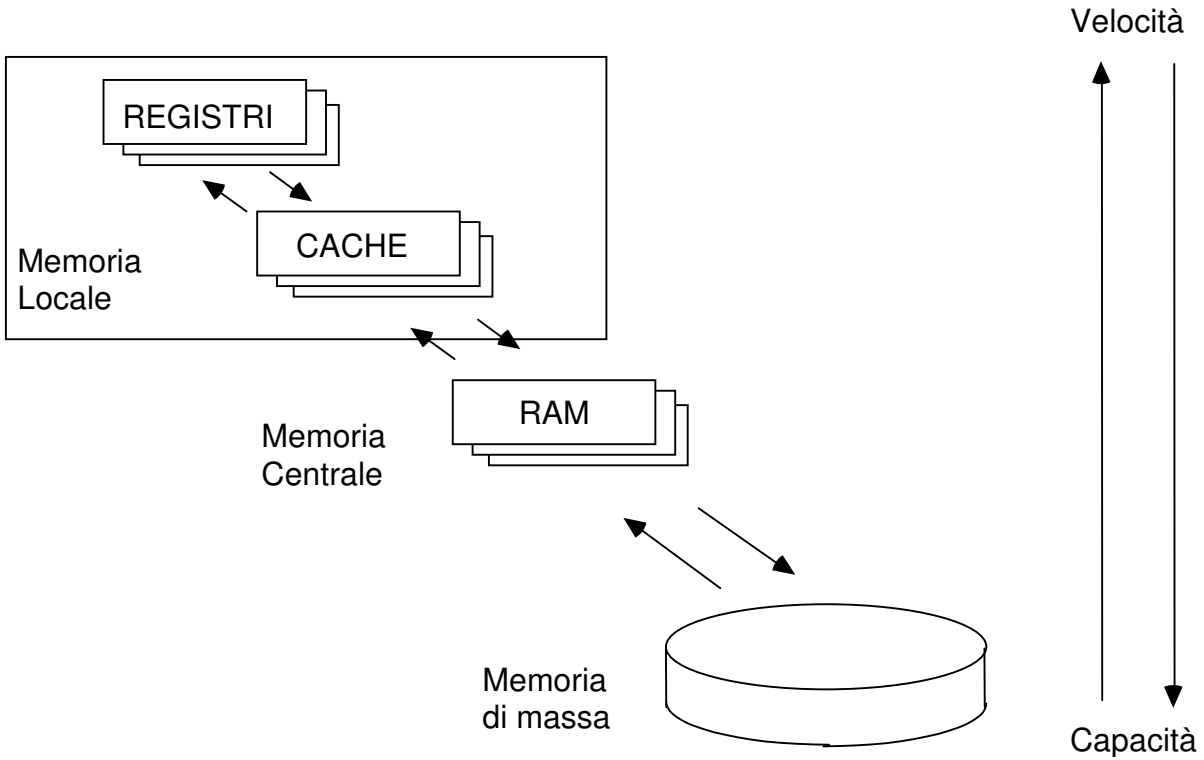
GERARCHIA DELLE MEMORIE

PROBLEMA:

Sebbene la RAM sia veloce, ***non è abbastanza veloce*** per i moderni processori

CONSEGUENZA:

il processore ***perde tempo ad aspettare l'arrivo dei dati dalla RAM***



MEMORIE CACHE (1)

SOLUZIONE:

Inserire **tra processore e RAM** una ***memoria particolarmente veloce*** **(SRAM)** dove tenere i dati usati più spesso **(*memoria cache*)**

In questo modo,

- **la prima volta** che il microprocessore carica dei dati dalla memoria centrale, tali dati vengono caricati *anche sulla cache*
- **le volte successive**, i dati possono essere *letti dalla cache (veloce)* invece che dalla memoria centrale (più lenta)

MEMORIE CACHE (2)

DUBBIO:

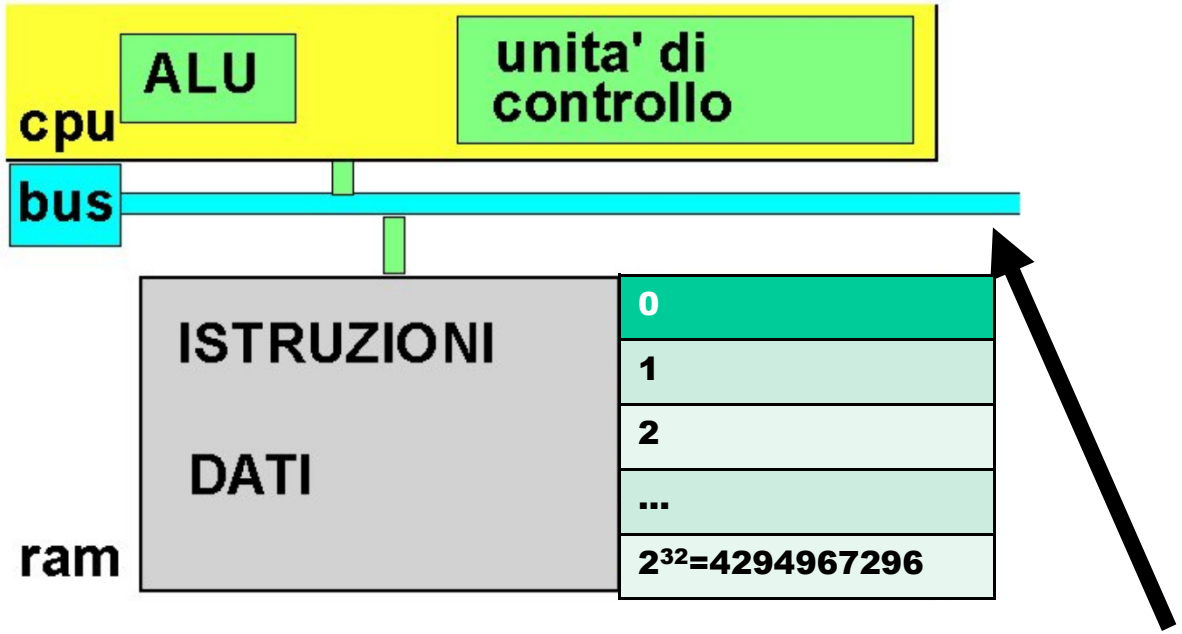
Ma se abbiamo memorie così veloci,
perché non le usiamo per costruire <u>tutta</u> la RAM?

Semplice...
perché costano molto!

OGGI, la cache è tipicamente già integrata dentro al processore **(cache di I/II livello)**, e altra può essere aggiunta (**cache di II/III livello**)

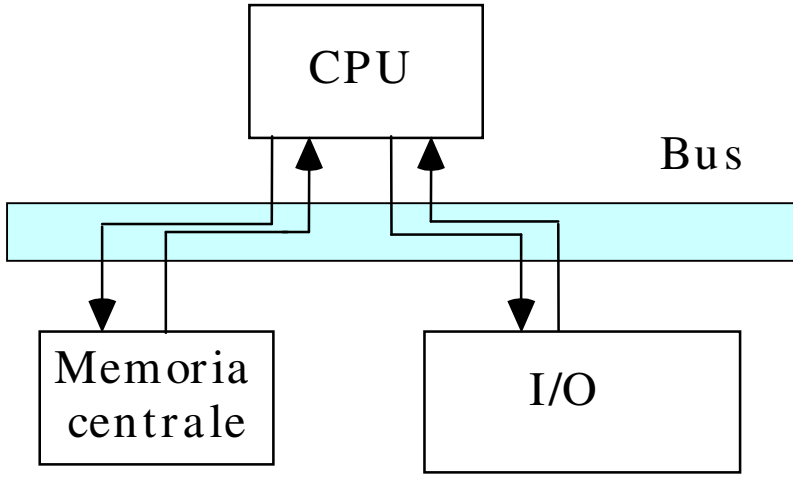
Nei processori con architettura *multi-core* ogni singolo *core* ha la propria cache di I livello, mentre la cache di II livello (sempre integrata) è normalmente condivisa fra i processori

BUS DI SISTEMA



Il Bus di Sistema interconnette la CPU, la memorie e le interfacce verso dispositivi periferici (I/O, memoria di massa, etc.)

BUS DI SISTEMA (2)

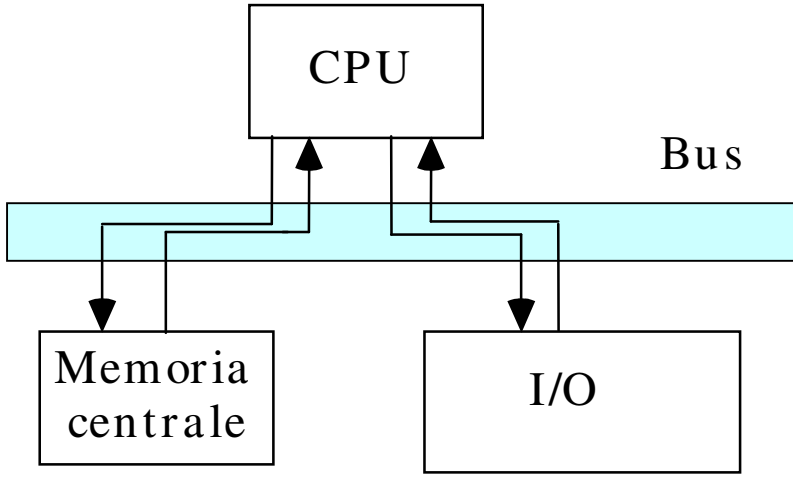


Il Bus collega ***due unità funzionali alla volta***:

- **una trasmette**…
- ... e **l'altra riceve**

Il trasferimento dei dati avviene usualmente ***sotto il controllo della CPU*** (funzionamento base)

BUS DI SISTEMA (3)



Il Bus è in realtà **un insieme di linee diverse:**

- **bus dati (*data bus*)**
- **bus indirizzi (*address bus*)**
- **bus comandi (*command bus*)**

BUS DI SISTEMA (4)

BUS DATI

- **bidirezionale**
- serve per trasmettere dati ***dalla memoria o viceversa***

BUS INDIRIZZI

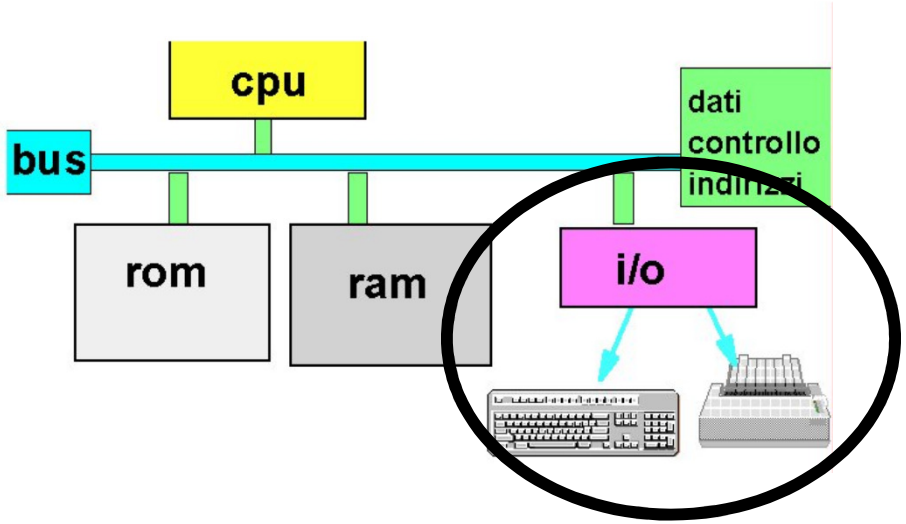
- **unidirezionale**
- serve per trasmettere ***il contenuto del registro indirizzi alla memoria***
  (si seleziona una specifica cella su cui viene eseguita o un'operazione di lettura o una operazione di scrittura)

BUS DI SISTEMA (5)

BUS COMANDI

- **bidirezionale**
- tipicamente usato per ***inviare comandi verso la memoria*** (es: lettura o scrittura) o ***verso una periferica*** (es. stampa verso la stampante → interfaccia)
- può essere usato in alcuni casi per ***inviare comandi verso il processore***

INTERFACCE DI I/O



Una **interfaccia** è un dispositivo che consente all'elaboratore di **comunicare con una periferica** (tastiere, mouse, dischi, terminali, stampanti, ...)

Le interfacce sono molto diverse tra loro e dipendono dal tipo di unità periferica da connettere

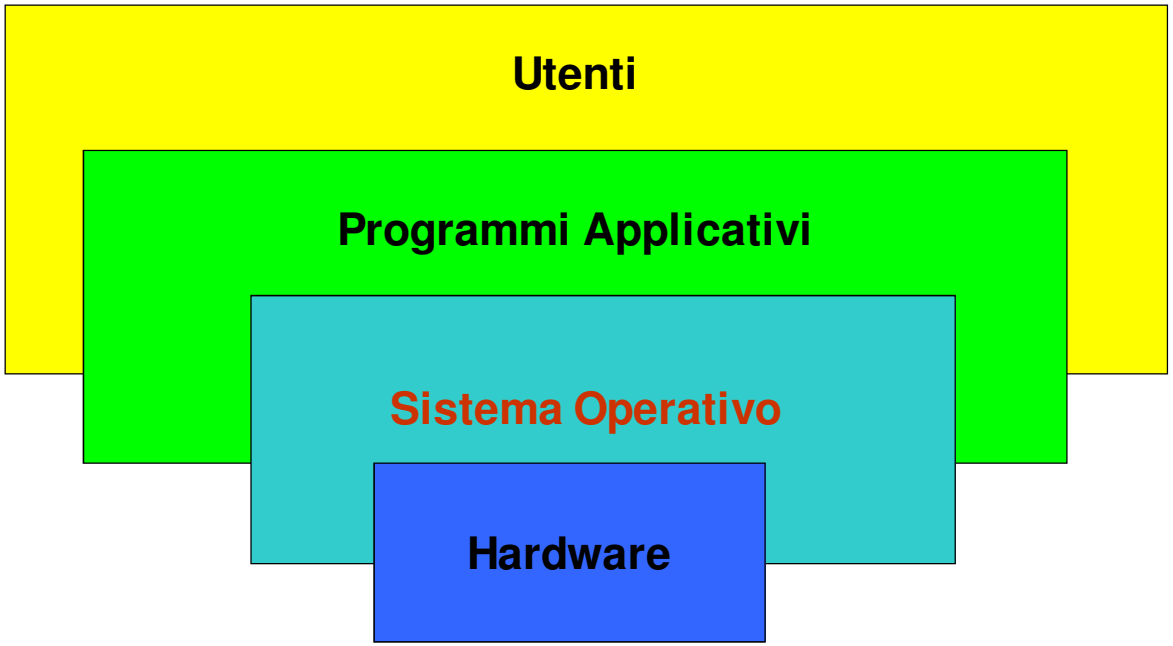
OLTRE la macchina di Von Neumann

- **Problema:** nella Macchina di Von Neumann le operazioni sono ***strettamente sequenziali***
- Altre soluzioni introducono forme di ***parallelismo***
  - **processori dedicati** (*coprocessori*) al calcolo numerico, alla gestione della grafica, ***all'I/O***
  - **esecuzione *in parallelo*** delle varie fasi di un'istruzione: mentre se ne esegue una, si acquisiscono e decodificano le istruzioni successive (***pipeline***)
  - ***architetture completamente diverse***: sistemi multi-processore, macchine dataflow, reti neurali, …

Che cos'è un SISTEMA OPERATIVO (SO)?

È un ***programma*** (o un insieme di programmi) che agisce come ***intermediario tra l'utente e l'hardware*** del computer:

- fornisce un **ambiente di sviluppo e di esecuzione** per i programmi applicativi
- fornisce una **visione astratta** dell'HW
- **gestisce** efficientemente le risorse del sistema di calcolo



SO e HARDWARE

– SO interfaccia programmi applicativi o di sistema con le risorse HW:

- CPU **- memoria** volatile e persistente

- dispositivi di I/O - **connessione di rete**

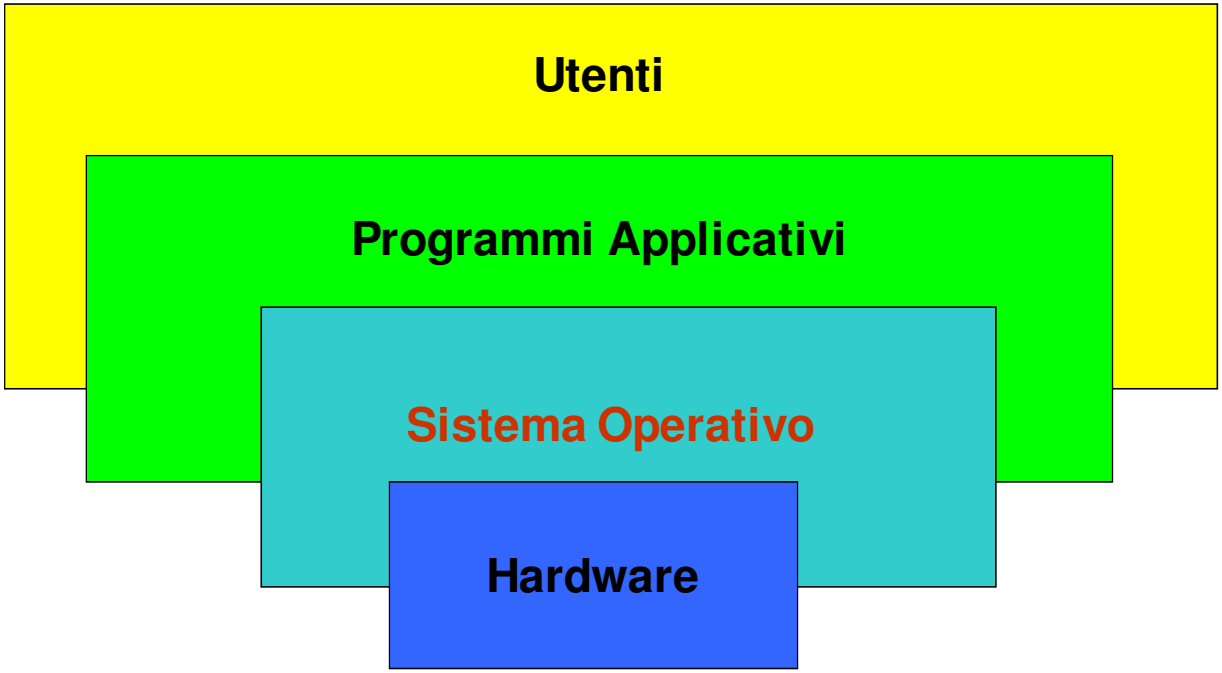
- **dispositivi di comunicazione** **- …**

– SO *mappa* le risorse HW in **risorse logiche**, accessibili attraverso interfacce ben definite:

- ***processi*** (CPU)
- ***file system*** (dischi)
- ***memoria virtuale*** (memoria), ...

Che cos'è un SO?

- Un programma che ***gestisce risorse*** del sistema di calcolo in modo ***corretto ed efficiente*** e le ***alloca*** ai programmi/utenti
- Un programma che innalza il ***livello di astrazione*** con cui utilizzare le ***risorse logiche*** a disposizione



EVOLUZIONE SO

Prima generazione (anni ‘50)

- linguaggio macchina
- dati e programmi su schede perforate

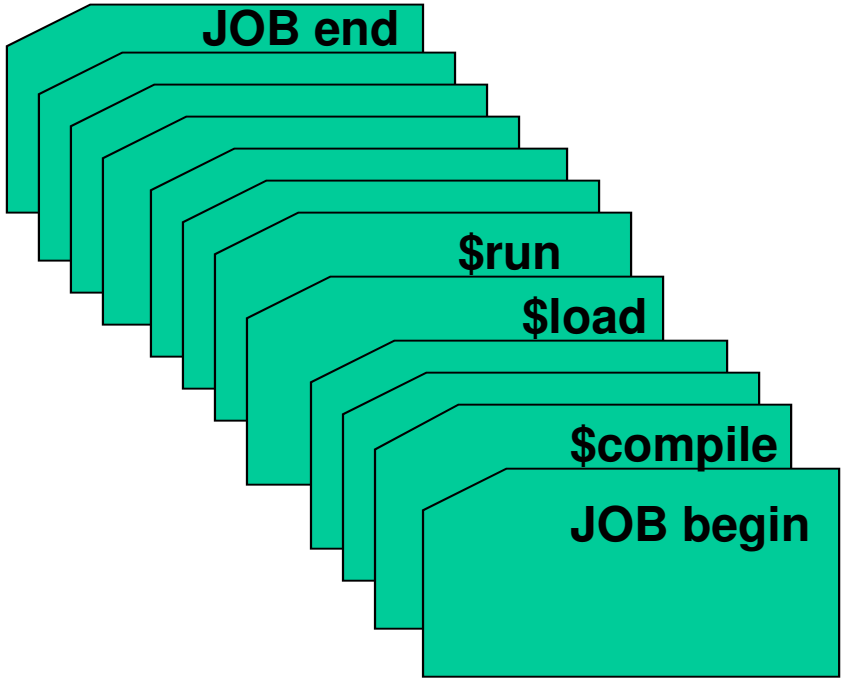
Seconda generazione (‘55-’65):

sistemi batch semplici

- linguaggio di alto livello (fortran)
- input mediante schede perforate
- aggregazione di programmi in **lotti** (batch) con esigenze simili

SISTEMI BATCH SEMPLICI (1)

Batch: insieme di programmi (*job*) da eseguire **in modo sequenziale**



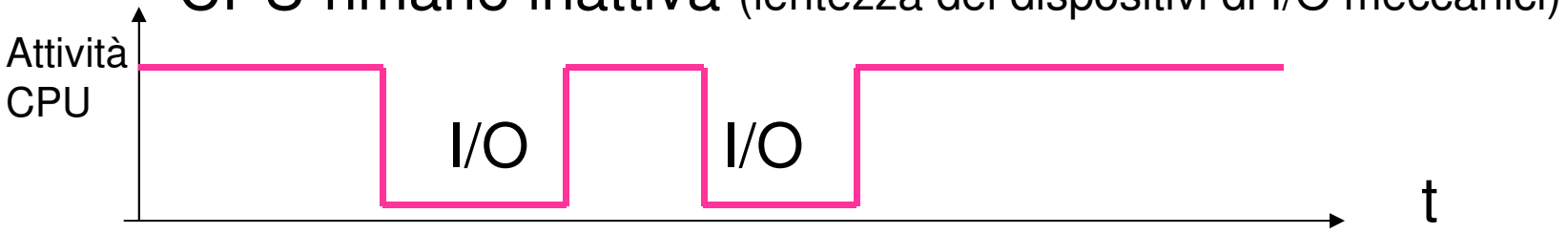
SISTEMI BATCH SEMPLICI (2)

Compito di SO (*monitor*):

trasferimento di controllo da un job (appena terminato) al prossimo da eseguire

Caratteristiche dei sistemi batch semplici:

- SO ***residente in memoria*** (monitor)
- ***assenza di interazione*** tra utente e job
- ***scarsa efficienza***: durante l'I/O del job corrente, la CPU rimane inattiva (lentezza dei dispositivi di I/O meccanici)



SISTEMI BATCH SEMPLICI (3)

In memoria centrale, ad ogni istante, è ***caricato (al più) un solo job*:**

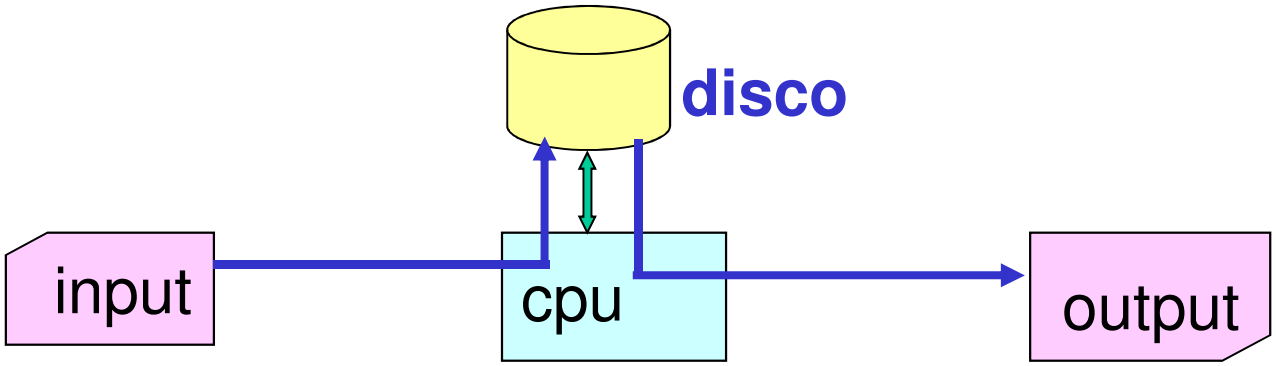
Configurazione della memoria centrale in sistemi batch semplici

SISTEMI BATCH SEMPLICI (4)

Spooling (Simultaneous Peripheral Operation On Line): simultaneità di I/O e attività di CPU

disco viene impiegato come **buffer** molto ampio, dove

- ***leggere*** in anticipo i dati
- ***memorizzare*** temporaneamente i risultati (in attesa che il dispositivo di output sia pronto)
- caricare ***codice e dati del job successivo***: -> possibilità di **sovrapporre I/O** di un job **con elaborazione** di un altro job



SISTEMI BATCH SEMPLICI (5)

Problemi:

- finché il job corrente non è terminato, il ***successivo non può iniziare l'esecuzione***
- se un job si ***sospende*** in attesa di un evento, la CPU rimane ***inattiva***
- ***non c'è interazione*** con l'utente

SISTEMI BATCH MULTIPROGRAMMATI (1)

Sistemi batch semplici: ***l'attesa*** di un ***evento*** causa inattività della CPU. Per evitare il problema

$\Rightarrow$ **Multiprogrammazione**

Pool di job contemporaneamente presenti su disco:

- SO seleziona un ***sottoinsieme dei job*** appartenenti al pool da ***caricare in memoria centrale***
- mentre un job è in ***attesa di un evento***, il sistema operativo ***assegna CPU a un altro job***

SISTEMI BATCH MULTIPROGRAMMATI (2)

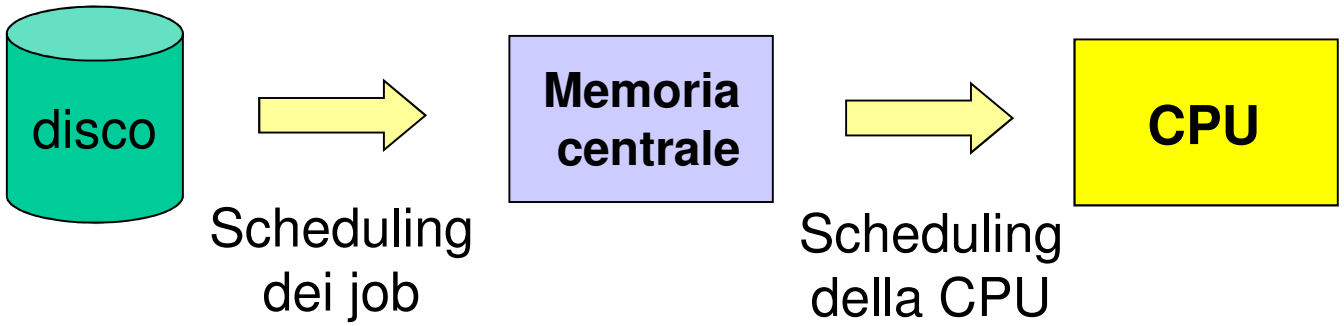
SO è in grado di ***portare avanti*** l'esecuzione di più job ***contemporaneamente***

- Ad ogni istante:
  - **un solo job** utilizza la CPU
  - **più job,** appartenenti al pool selezionato e caricati in memoria centrale, attendono di acquisire la CPU
- Quando il job che sta utilizzando la CPU si ***sospende in attesa di un evento***:
  - SO ***decide*** a quale job assegnare la CPU ed effettua lo scambio (***scheduling***)

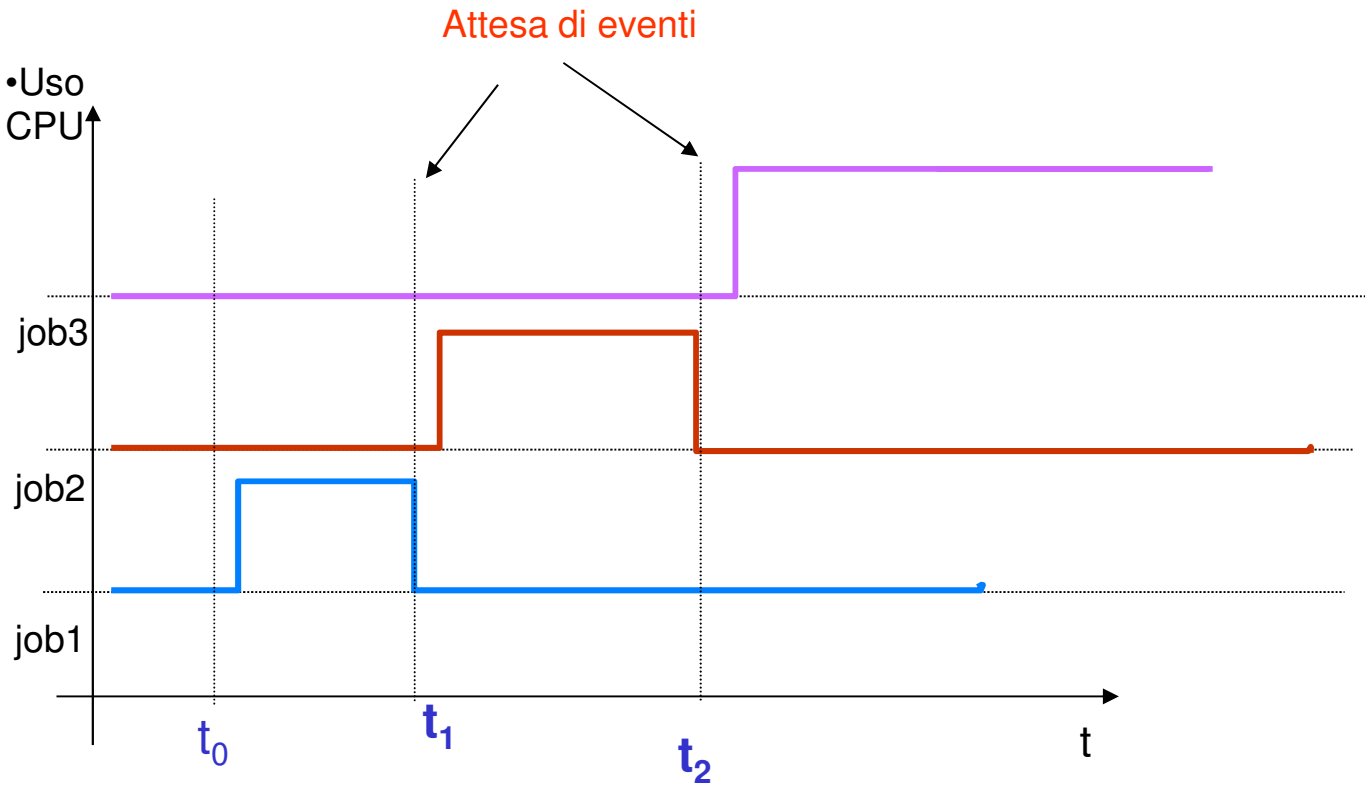
BATCH MULTIPROG.: SCHEDULING (1)

SO effettua delle scelte tra tutti i job

- quali job caricare in memoria centrale: **scheduling dei job** (*long-term scheduling*)
- a quale job assegnare la CPU: **scheduling della CPU** o (*short-term scheduling*)

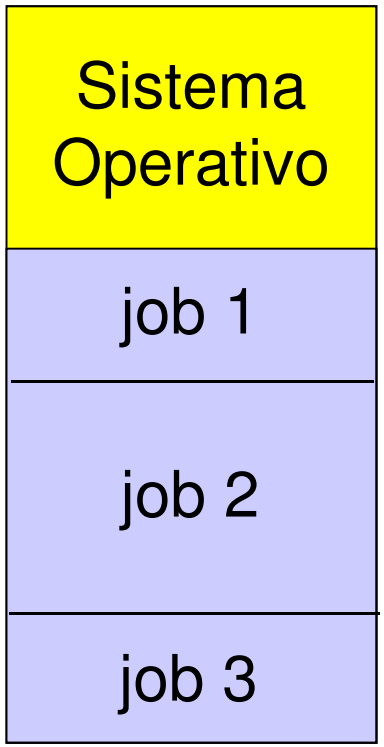


BATCH MULTIPROG.: SCHEDULING (2)



SISTEMI BATCH MULTIPROGRAMMATI

In memoria centrale, ad ogni istante, possono essere caricati più job:



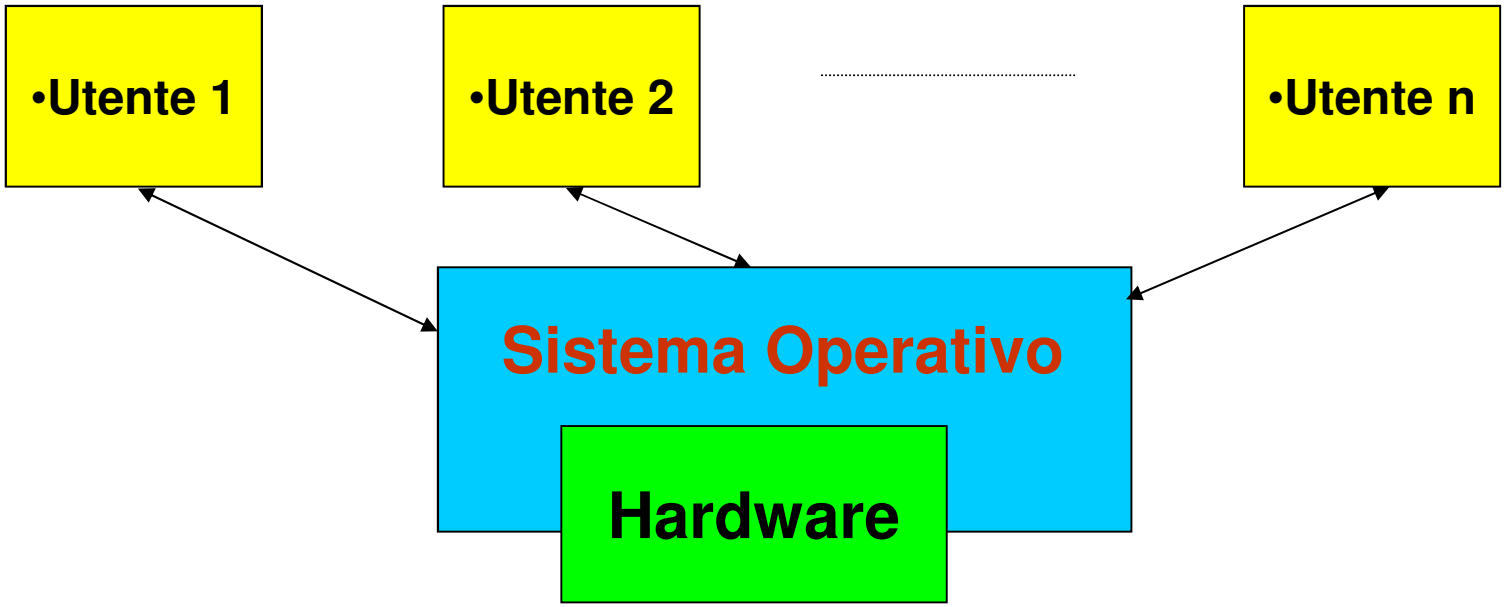
Configurazione della ***memoria centrale*** in sistemi batch multiprogrammati

Necessità di protezione

SISTEMI TIME-SHARING (Multics, 1965)

Nascono dalla necessità di:

- ***interattività*** con l'utente
- ***multi-utenza*****:** più utenti interagiscono contemporaneamente con SO



SISTEMI TIME-SHARING (1)

Multiutenza: il sistema presenta ad ogni utente una ***macchina virtuale completamente dedicata*** in termini di

- utilizzo della CPU
- utilizzo di altre risorse, ad es. file system

Interattività: per garantire un'accettabile velocità di "reazione" alle richieste dei singoli utenti, SO ***interrompe l'esecuzione*** di ogni job dopo un intervallo di tempo prefissato (***quanto di tempo***, o ***time slice***), assegnando la CPU a un altro job

SISTEMI TIME-SHARING (2)

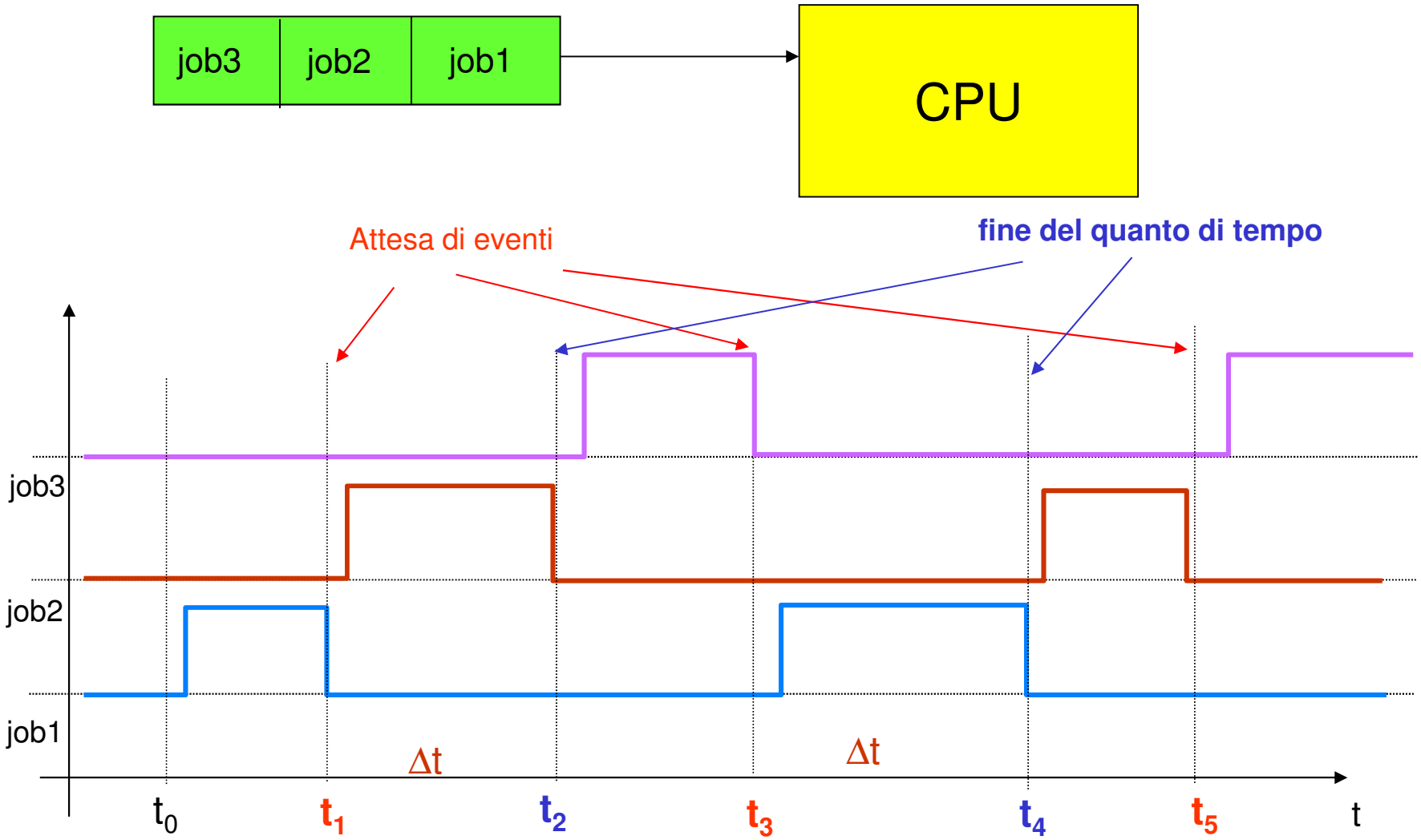
Sono sistemi in cui:

- attività della ***CPU è dedicata a job diversi*** che si alternano ***ciclicamente*** nell'uso della risorsa
- frequenza di commutazione della CPU è tale da fornire l'illusione ai vari utenti di una macchina completamente dedicata (***macchina virtuale***)

Cambio di contesto (***context switch***):

operazione di trasferimento del controllo da un job al successivo ⇨ costo aggiuntivo (*overhead*)

SISTEMI TIME-SHARING (3)



TIME-SHARING: REQUISITI

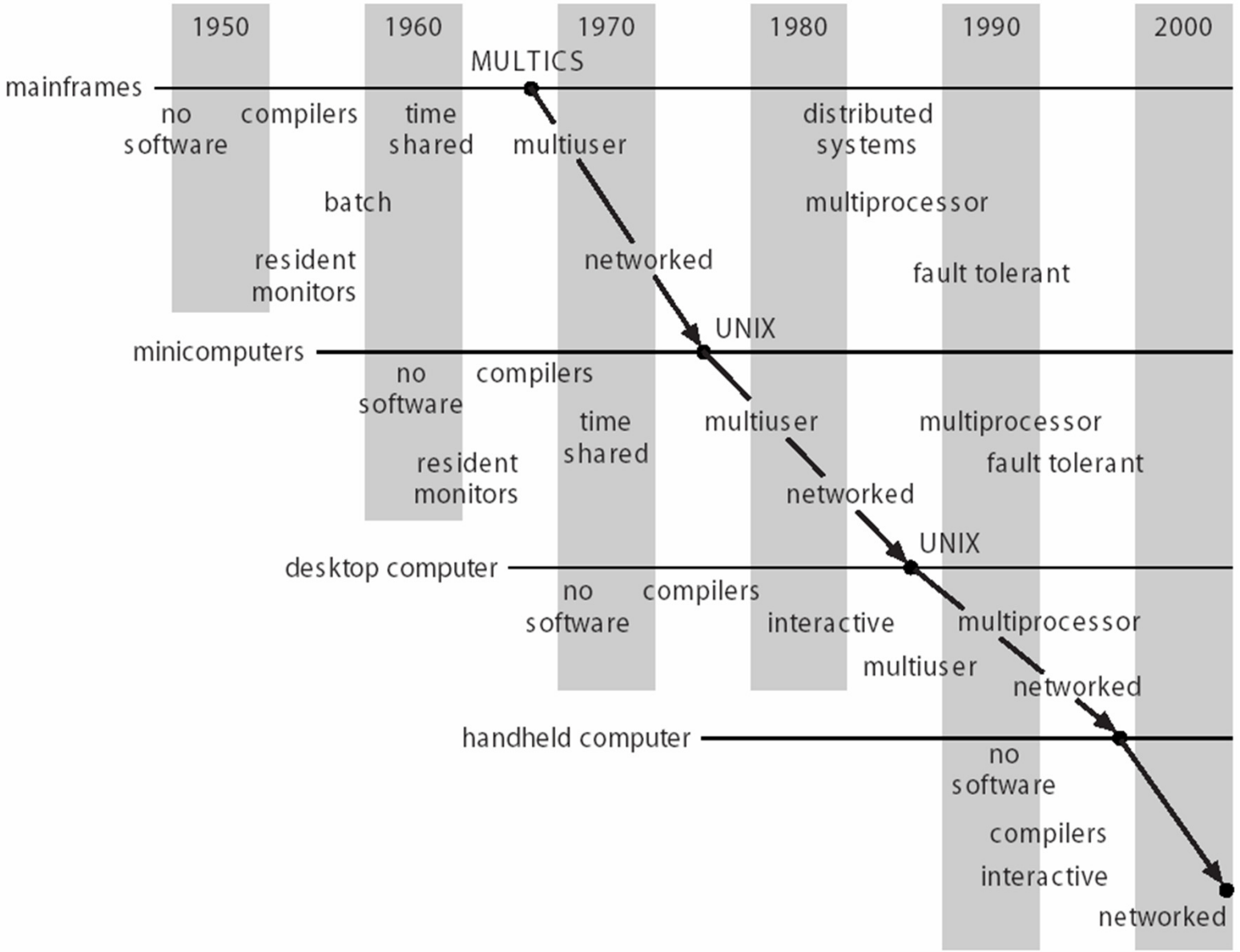
- **Gestione/protezione** della memoria:
  - trasferimenti memoria-disco
  - ***separazione degli spazi*** assegnati ai diversi job
  - molteplicità job + limitatezza della memoria

    ⇒ ***memoria virtuale***
- **Scheduling** CPU
- **Sincronizzazione/comunicazione** tra job:
  - interazione
  - prevenzione/trattamento di blocchi critici (***deadlock***)
- **Interattività**: ***accesso on-line*** al ***file system*** per permettere agli utenti di accedere semplicemente a codice e dati

ESEMPI di SO ATTUALI

- **MSDOS**: monoprogrammato, monoutente
- **Windows 95/98, molti SO attuali per dispositivi portabili (Symbian, PalmOS)**: multiprogrammato (time sharing), tipicamente monoutente
- **Windows NT/2000/XP**: multiprogrammato, “multiutente”
- **MacOSX**: multiprogrammato, multiutente
- **UNIX/Linux**: multiprogrammato, multiutente

EVOLUZIONE CONCETTI nei SO



PRINCIPALI FUNZIONI di SO

Quali sono le ***componenti*** di un SO?

- gestione dei ***processi***
- gestione della ***memoria centrale***
- gestione di ***memoria secondaria*** e ***file system***
- gestione dell'***I/O***
- ***protezione e sicurezza***
- interfaccia utente/programmatore

Quali sono le ***relazioni mutue*** tra le componenti?

PROCESSI

Processo = programma in esecuzione

- il ***programma*** è ***un'entità passiva*** (un insieme di byte contenente le istruzioni che dovranno essere eseguite)
- ***il processo è un'entità attiva:***
  - è ***l'unità di lavoro/esecuzione*** all'interno del sistema. ***Ogni attività all'interno del SO è rappresentata da un processo***
  - ***è l'istanza di un programma in esecuzione***

Processo = programma +
contesto di esecuzione (PC, registri, …)

GESTIONE dei PROCESSI

In un sistema multiprogrammato: più processi possono essere ***simultaneamente presenti*** nel sistema

Compito cruciale del SO

- ***creazione/terminazione*** dei processi
- ***sospensione/ripristino*** dei processi
- ***sincronizzazione/comunicazione*** dei processi
- ***gestione del blocco critico (deadlock)*** di processi

GESTIONE della MEMORIA CENTRALE

HW di sistema di elaborazione è equipaggiato con ***un unico spazio di memoria*** accessibile direttamente da CPU e dispositivi

Compito cruciale di SO

- ***separare gli spazi di indirizzi*** associati ai processi
- ***allocare/deallocare memoria*** ai processi
- ***memoria virtuale*** **-** gestire ***spazi logici di indirizzi*** di dimensioni complessivamente ***superiori allo spazio fisico***
- realizzare i collegamenti (***binding***) tra ***memoria logica e memoria fisica***

GESTIONE DISPOSITIVI I/O

Gestione dell'I/O rappresenta una parte importante di SO:

- ***interfaccia*** tra programmi e dispositivi
- per ogni dispositivo: ***device driver***
  - ***routine per l'interazione con un particolare dispositivo***
  - contiene ***conoscenza specifica*** sul dispositivo (ad es., routine di gestione delle interruzioni)

GESTIONE MEMORIA SECONDARIA

Tra tutti i dispositivi, la ***memoria secondaria*** riveste un ruolo particolarmente importante**:**

- ***allocazione/deallocazione*** di spazio
- gestione dello ***spazio libero***
- ***scheduling*** delle operazioni sul disco

Di solito:

- la **gestione dei file** usa i meccanismi di gestione della memoria secondaria
- la **gestione della memoria secondaria** è indipendente dalla gestione dei file

GESTIONE FILE SYSTEM (1)

Ogni sistema di elaborazione dispone di uno o più dispositivi per la memorizzazione persistente delle informazioni (***memoria secondaria***)

Compito di SO

fornire una ***visione logica uniforme della memoria secondaria*** (indipendente dal tipo e dal numero dei dispositivi):

- realizzare il ***concetto astratto di file***, come unità di memorizzazione logica
- fornire una struttura astratta per ***l'organizzazione*** dei file (***direttorio***)

GESTIONE FILE SYSTEM (2)

Inoltre, SO si deve occupare di:

- creazione/cancellazione di file e direttori
- manipolazione di file/direttori
- associazione tra file e dispositivi di memorizzazione secondaria

Spesso file, direttori e dispositivi di I/O vengono ***presentati*** a utenti/programmi ***in modo uniforme***

PROTEZIONE e SICUREZZA (1)

In un sistema multiprogrammato, più entità (processi o utenti) possono utilizzare le risorse del sistema contemporaneamente:
necessità di protezione

Protezione: controllo dell'accesso alle risorse del sistema da parte di processi (e utenti) mediante

- ***autorizzazioni***
- ***modalità di accesso***

Risorse da proteggere:

- ❑ memoria
- ❑ processi
- ❑ file
- ❑ dispositivi

PROTEZIONE e SICUREZZA (2)

Sicurezza:

se il sistema appartiene a una rete, la ***sicurezza misura l'affidabilità del sistema nei confronti di accessi (attacchi) dal mondo esterno***

Non ce ne occuperemo all'interno di questo corso…

INTERFACCIA UTENTE

SO presenta un'interfaccia che consente l'interazione con l'utente

- **interprete comandi** (***shell***): l'interazione avviene mediante una linea di comando
- **interfaccia grafica** (graphical user interface, ***GUI***): l'interazione avviene mediante ***interazione*** mouse-elementi grafici su desktop; di solito è organizzata a finestre