Fondamenti di Informatica

(L-Z)

Corso di Laurea in Ingegneria Gestionale

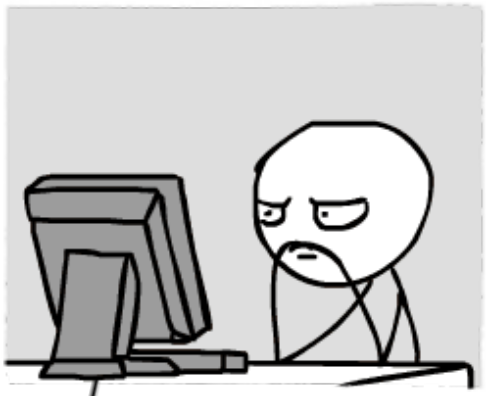
Introduzione all'Informatica

Prof. Stefano Mariani

Dott. Alket Cecaj

Indice

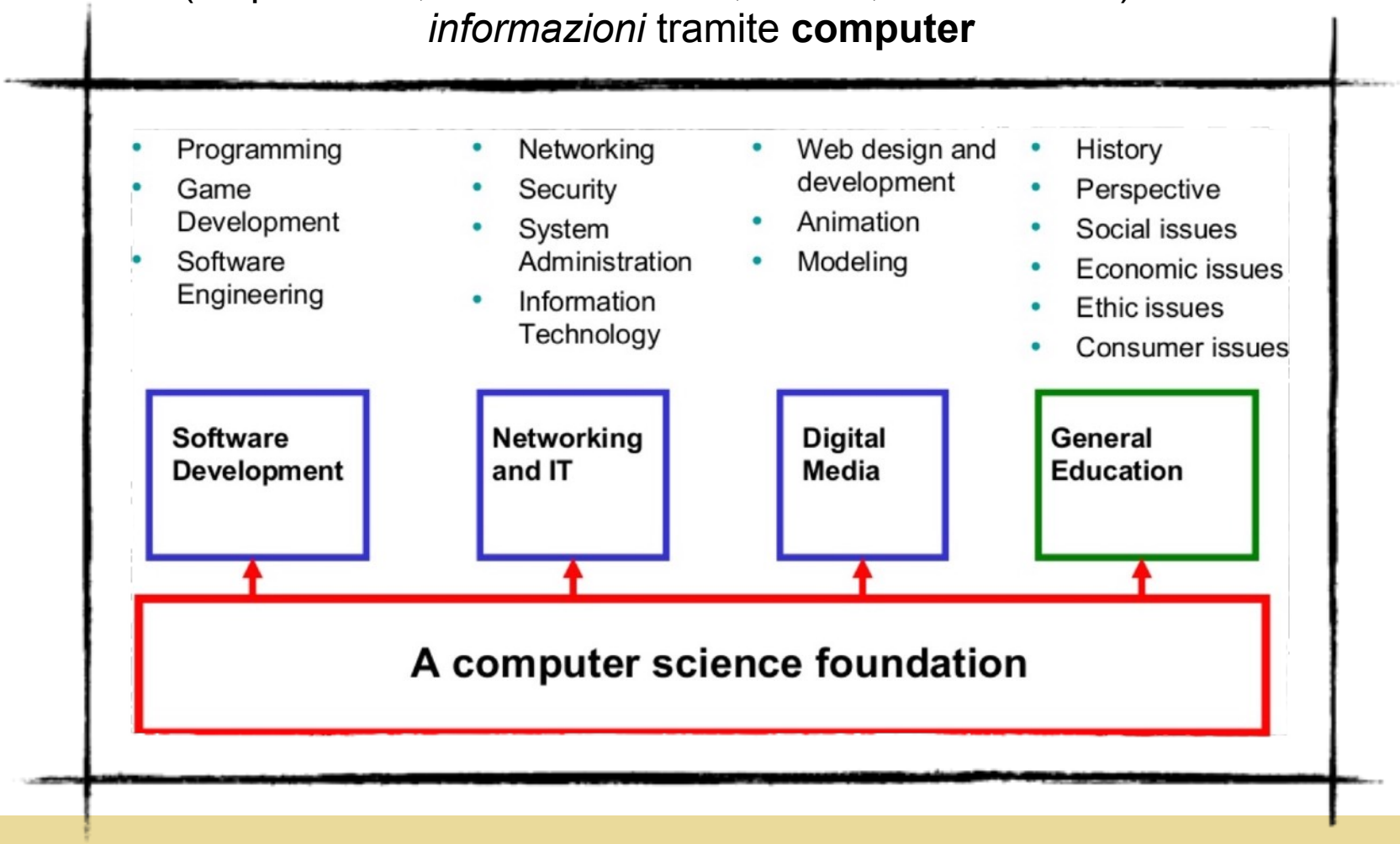
- Cos’è l’informatica?
- Com'è fatto un computer?
- Cos’è un sistema operativo?
- Cos’è “ l’ internet ”??



L' INFORMATICA

Definizione

Informatica (**computer science**) è la *scienza* che studia i *metodi* e le *tecniche* per il *trattamento* (acquisizione, memorizzazione, analisi, trasmissione) *automatico* delle *informazioni* tramite **computer**



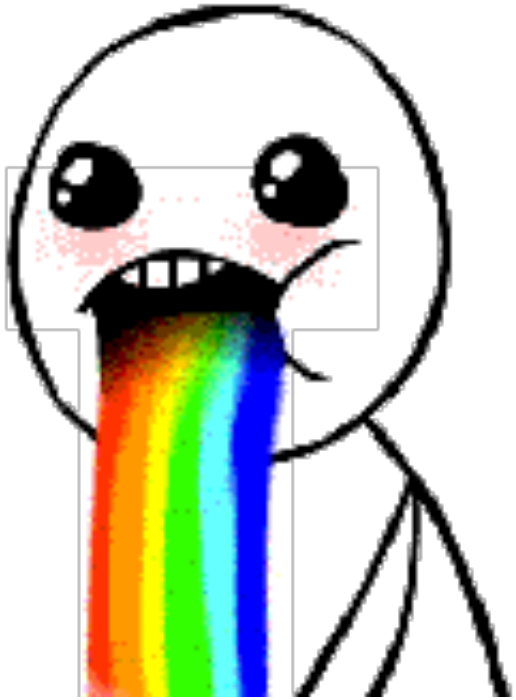
Laurea in Ing. Informatica

Laurea Magistrale in Ing. Inf.

Insomma…

- *L'informatica è ovunque* e tratta di un sacco di argomenti apparentemente diversi
- La nozione stessa di computer comprende *dispositivi largamente eterogenei*

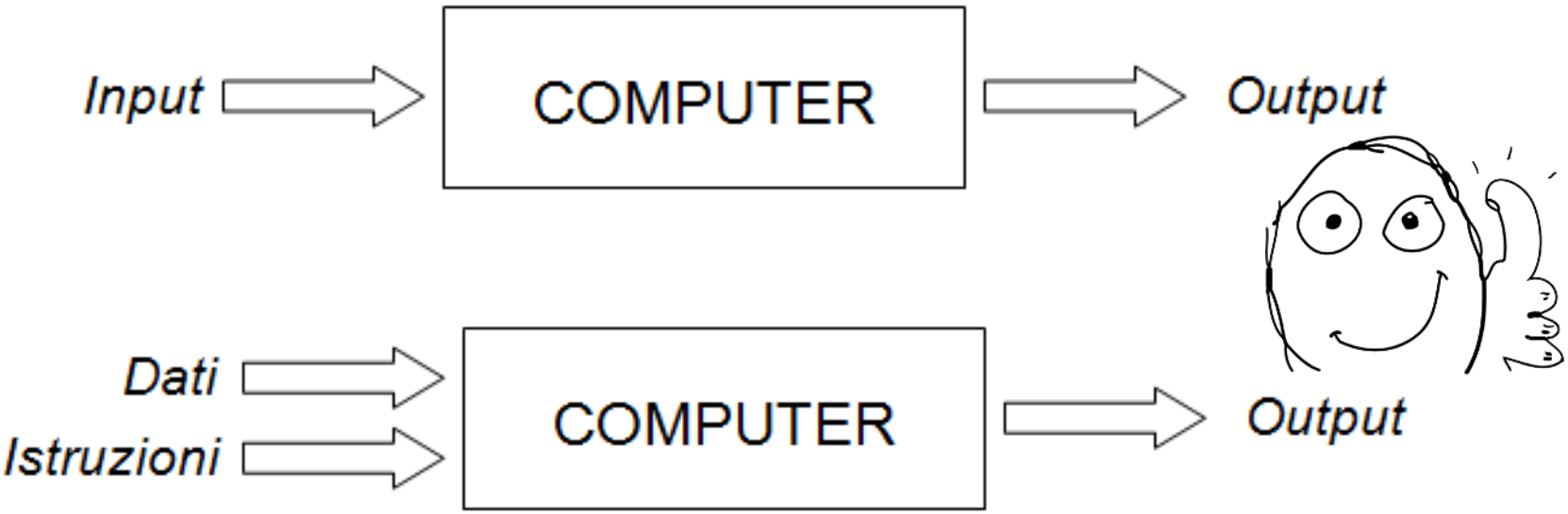
Da dove cominciamo?



ARCHITETTURA DI UN COMPUTER

Computer

Un **computer** (o calcolatore elettronico) è una macchina che *produce* dati in uscita (**output**) sulla base delle informazioni che *riceve* in ingresso (**input**) elaborate secondo precise *istruzioni* (**programma**)

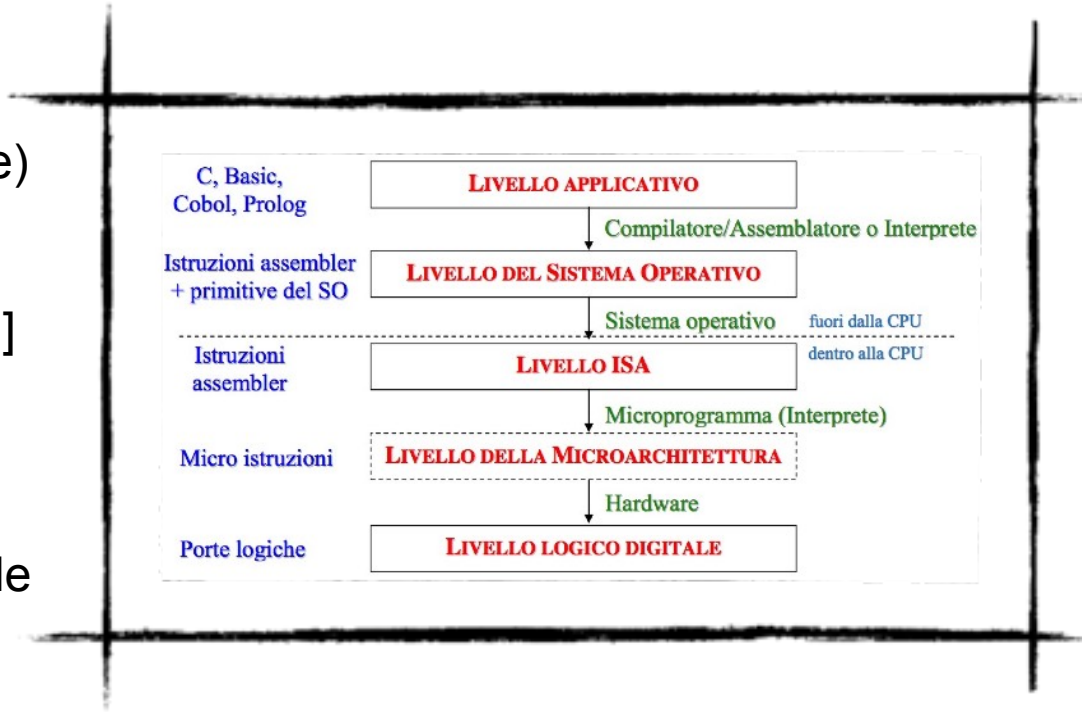


Programma

- **Programmare** significa scrivere la *sequenza* di istruzioni (**algoritmo**) necessarie a risolvere un problema, in un *linguaggio comprensibile al computer*
- Il linguaggio comprensibile al computer, formato dalle *istruzioni direttamente eseguibili dall'hardware*, è il **linguaggio macchina** (o **assembly**)
  - l'assembly non è adatto ai programmatori
    - numero di istruzioni estremamente limitato
    - bassissimo "livello di astrazione" (concetti per pensare alla soluzione)
  - si crea un *linguaggio di più alto livello* L[1], eseguibile da un **compilatore/interprete**, che traduce istruzioni L[1] in istruzioni assembly L[0]
    - questo processo di *astrazione* è solitamente iterato più volte

Linguaggio di programmazione

- La traduzione da un livello al sottostante può avvenire secondo due tecniche:
  - **compilazione**, per cui il traduttore (detto compilatore) *produce un programma* in linguaggio L[i-1] dal programma in linguaggio L[i]
  - **interpretazione**, per cui il traduttore (detto interprete) *esegue direttamente* istruzioni in L[i-1] derivandole dal programma scritto in L[i]

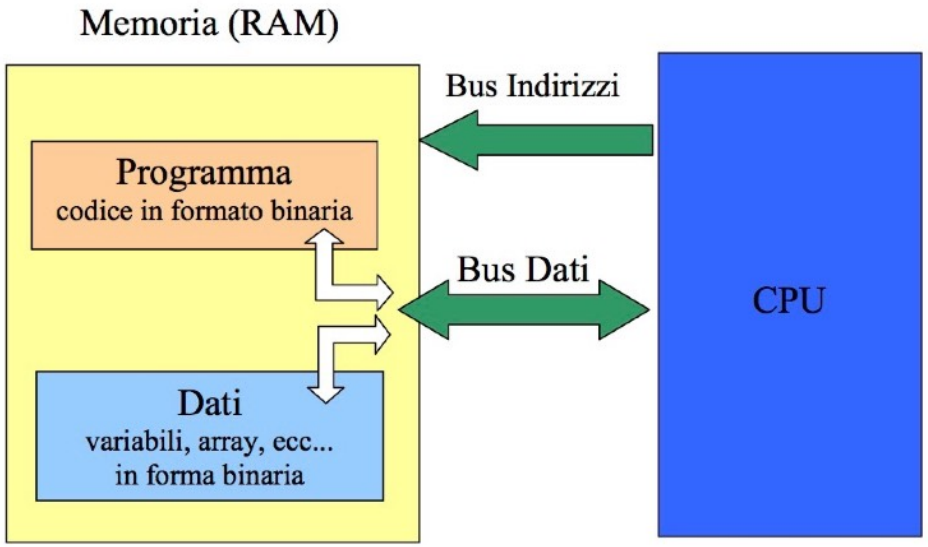


Classificazione calcolatori

- Sistemi embedded
- SmartPhone e Tablet
- Console
- Personal Computer (PC)
- Server e Workstation
- Cluster
- Supercomputer

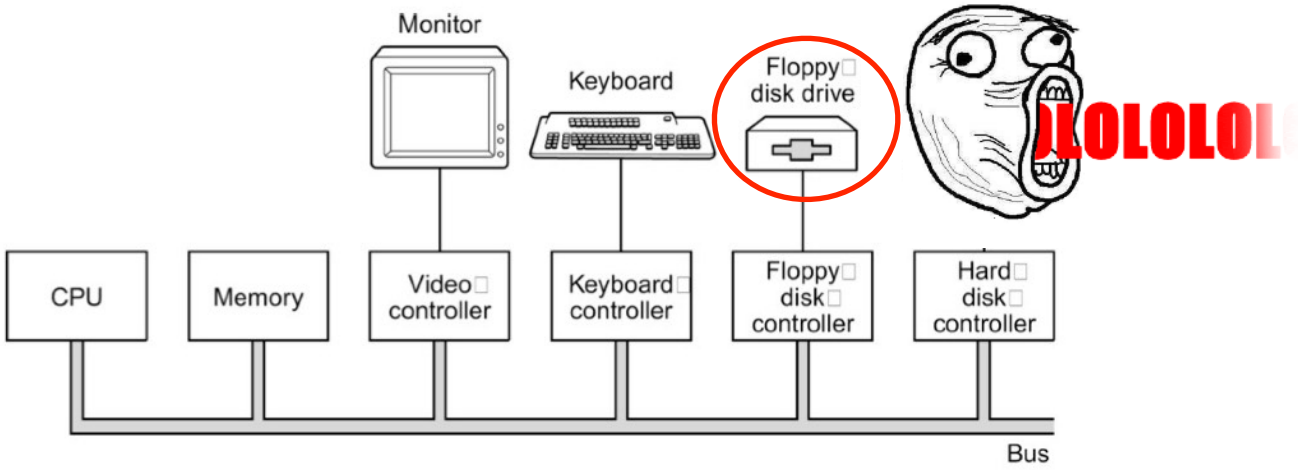
Architettura di Von Neumann

- Alla base della *maggior parte* dei computer moderni sta l'architettura di **Von Neumann** (1946)
  - si utilizza la *stessa memoria* sia per i dati di input che per i programmi
  - il **bus dati** trasferisce sia dati che istruzioni
  - il **bus indirizzi** indica alla memoria dove sono i dati e le istruzioni richieste



Periferiche

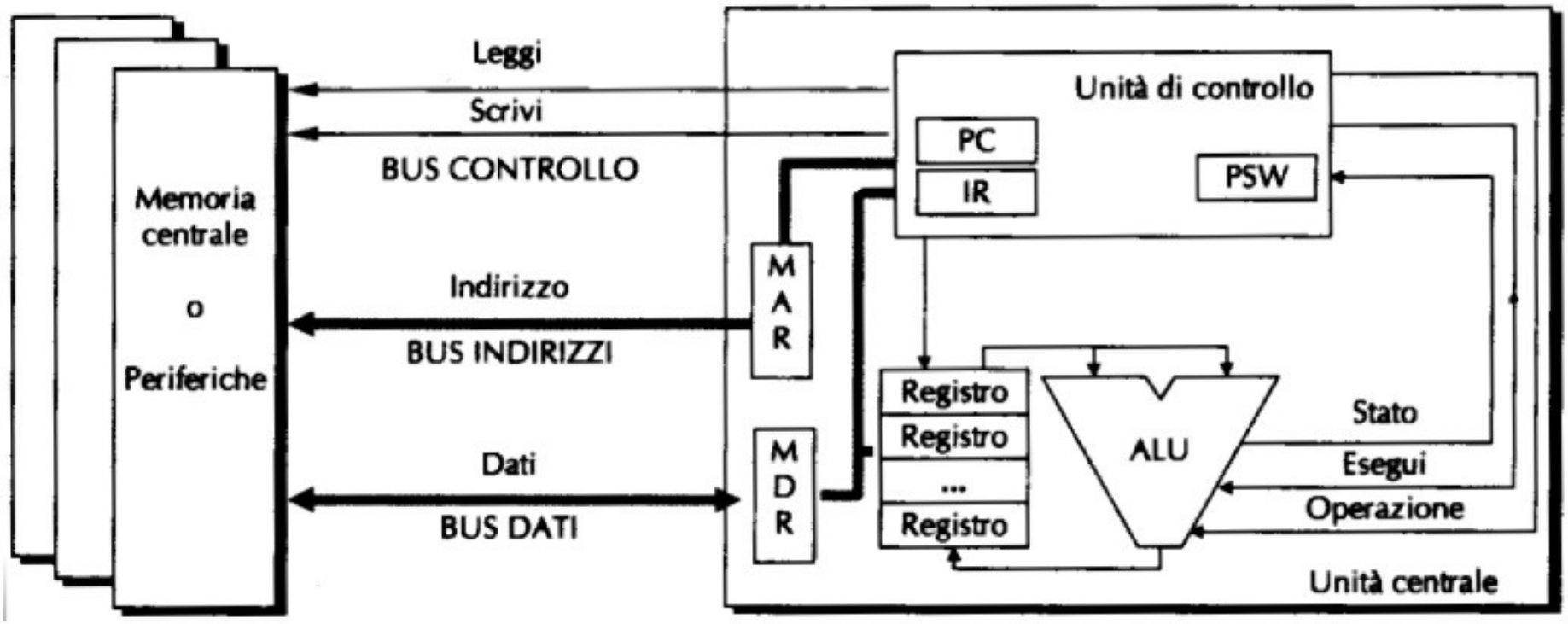
- Oltre a **CPU**(s) e **memoria**/e*, un computer è composto da dispositivi di input/output (**periferiche**)



- Il **bus** è un insieme di connessioni elettriche che trasportano informazioni tra i componenti collegati

*Nei computer moderni ci sono sempre più CPUs/cores e più memorie

CPU


Leggi
Scrivi
BUS CONTROLLO
Memoria centrale o Periferiche
Indirizzo
BUS INDIRIZZI
Dati
BUS DATI
Unità di controllo
PC
IR
PSW
MAR
MDR
Registro
Registro
...
Registro
ALU
Stato
Esegui
Operazione
Unità centrale

CPU

- La **CPU** (Central Processing Unit) esegue i programmi memorizzati nella *memoria centrale* leggendo le loro istruzioni ed eseguendole *in sequenza*
- La CPU è composta da:
  - **Unità di controllo**, legge le istruzioni e ne determina il tipo
  - **ALU** (Arithmetic Logic Unit), esegue le operazioni necessarie alle istruzioni (addizione binaria, AND, OR, …)
  - **Registri**, piccole memorie interne velocissima per risultati intermedi e informazioni di controllo
  - **Program Counter** (PC), indica la prossima istruzione da eseguire
  - **Instruction Register** (IR), memorizza l'istruzione da eseguire

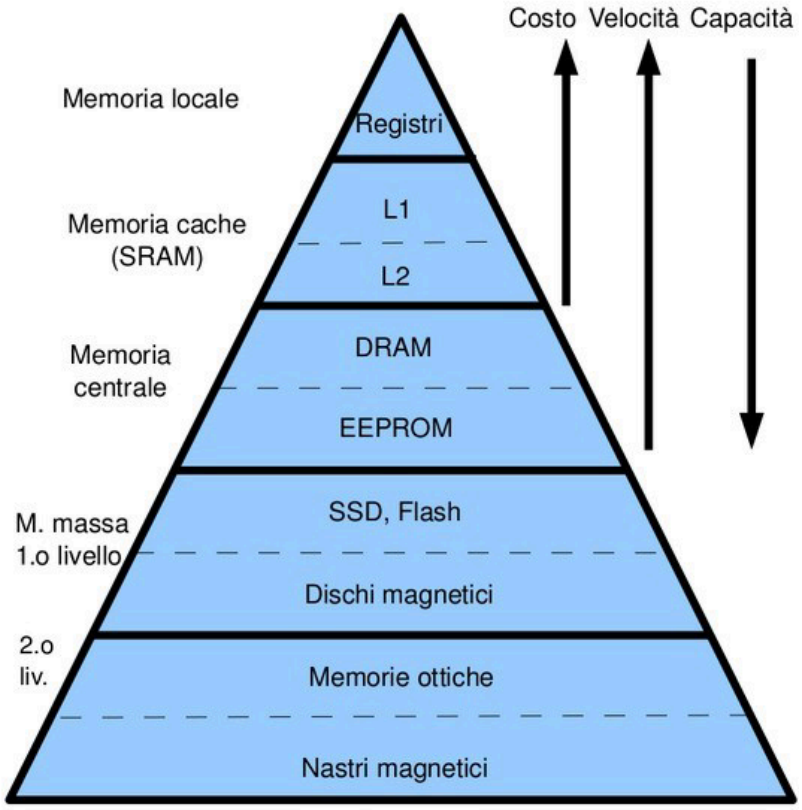
Esecuzione istruzioni

- La CPU opera in modo ciclico, ripetendo i passi seguenti fino al termine del programma
  - Caricamento (**fetch**): acquisizione dalla memoria di un'istruzione
  - Decodifica (**decode**): identificazione del tipo di operazione da eseguire
  - Esecuzione (**execute**): messa in atto delle operazioni implicate dall'istruzione

Memoria

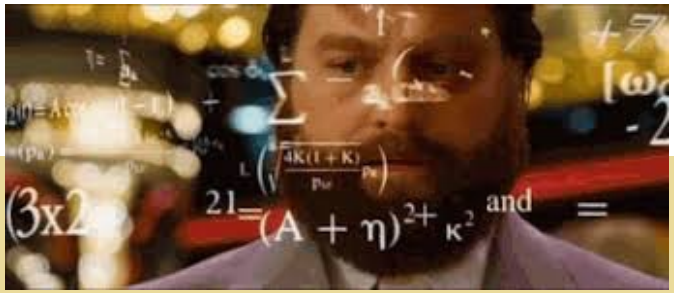
- Le **memorie** sono le componenti del computer in grado di memorizzare informazioni (e.g. *dati e programmi*)

- Diversi tipi per diversi scopi:
  - *volatile*, l'informazione si perde a computer spento
  - *persistente*, l'informazione si mantiene a computer spento
  - *on-line*, informazioni sempre accessibili (e.g. hard disk)
  - *off-line*, il supporto deve essere "montato" (e.g. dvd)

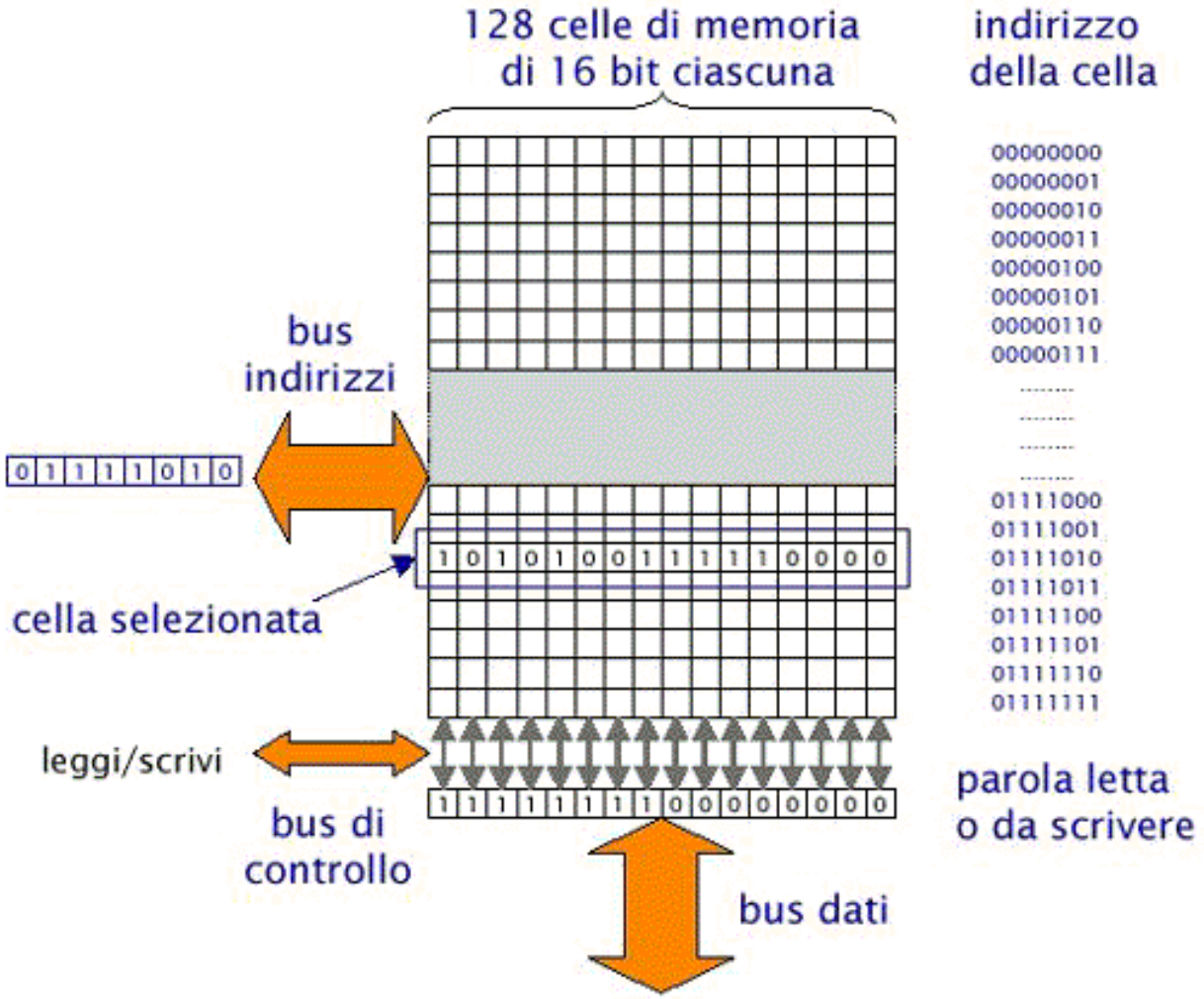


Memoria principale (RAM)

- La **memoria principale** è preposta a memorizzare i *programmi* in esecuzione e i relativi *dati*
- Ogni tipo di memoria è composta da un certo numero di **celle** (o locazioni) in grado di memorizzare una parte di informazione (parola)
  - nei computer moderni, 32 o *64 bit*
- Ogni cella è associata a un numero (**indirizzo**) che la identifica *univocamente*
  - gli indirizzi sono rappresentati da **numeri binari**
  - se un indirizzo ha M bit il numero massimo di celle indirizzabili sarà 2^M



Memoria principale (RAM)



RAPPRESENTAZIONE DIGITALE

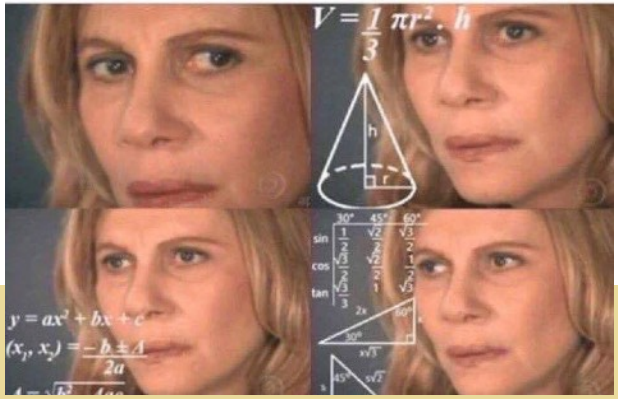
Analogico vs. digitale

- Segnale **analogico**: rappresentato da un numero *infinito* di valori *continui*
- Segnale **digitale**: rappresentato da un numero *finito* di valori *discreti*
- I computer utilizzano soltanto 2 valori discreti: 0 e 1 (logica **binaria**)

Ogni informazione memorizzata in un computer è internamente rappresentata in logica binaria!

Notazione posizionale

- I sistemi di *numerazione posizionale* associano alle cifre un diverso valore a seconda della posizione della cifra nel numero
  - la *base* (o radice) definisce il sistema di numerazione
  - una base *B* richiede *B* simboli
- Sistema **decimale**: B = 10 = {0 … 9}
  - e.g. $33 = 3*10^1 + 3*10^0 =$ **33**
- Sistema **binario**: B = 2 = {0 1}
  - e.g. $33 = 1*2^5 + 0*2^4 + 0*2^3 + 0*2^2 + 0*2^1 + 1*2^0 =$ **10 0001**



Unità di misura binarie

- Una singola cifra del sistema binario è chiamata **bit**
  - con *N* bit si possono rappresentare 2^n valori diversi
- Il bit è l'unità di misura *più piccola* utilizzabile dal computer
- Solitamente però, il computer maneggia *multipli* del bit:
  - Byte (B) = 8 bit
  - Kilobyte (KB) = 1024 Byte (non 1000!*)
  - **Megabyte** (MB) = 1024 KB
  - **Gigabyte** (GB) = 1024 MB
  - Terabyte (TB) = 1024 GB
  - Petabyte, Exabyte, Zettabyte, Yottabyte, …

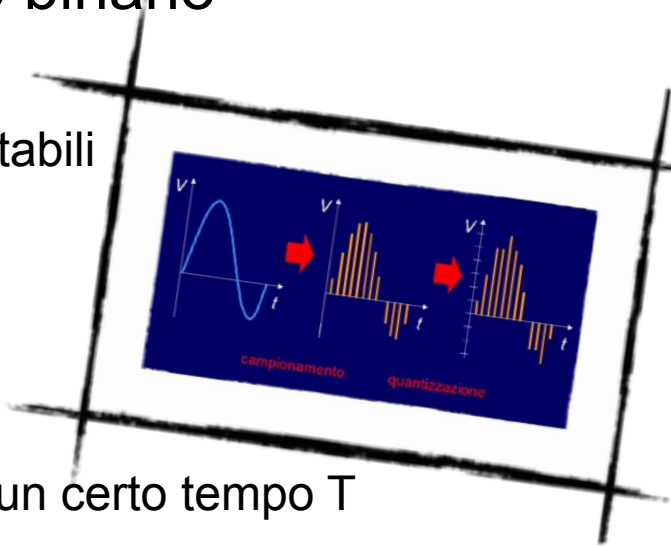
*Si ragiona in termini di multipli di 2, essendo nel sistema binario

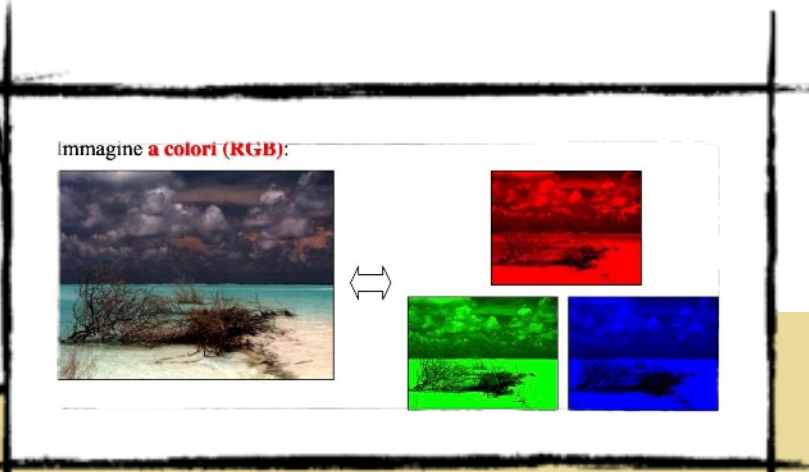
Rappresentazione numeri

- I numeri a **precisione finita** sono rappresentati da un numero finito di cifre (e.g. bit)
  - sono dunque un insieme *limitato*
- Le operazioni sull'insieme possono causare errori:
  - *underflow (overflow)*: il risultato è < (>) del più piccolo valore rappresentabile
  - *non appartenenza*: il risultato non è rappresentabile nell'insieme
- Per quanto un linguaggio sia "sofisticato" (e.g. Java vs C), non può sfuggire ai *limiti fisici* del calcolatore
  - programmando in Java può capitare 5.1 + 0.1 = 5.1999999 =)
- A meno che il numero non sia un'*esatta* potenza di 2, o una somma di potenze di 2, la rappresentazione conterrà *sempre* un errore
  - spesso però, l'errore è trascurabile =)

Caratteri, suoni, immagini

- Ogni **carattere** che digitate al computer è rappresentato internamente al computer come numero binario
  - e.g. *ASCII*: 7 bit, 128 caratteri rappresentabili
  - e.g. *UNICODE*: 16 bit, 65 536 caratteri rappresentabili
- Fisicamente, un **suono** è un'onda
  - è dunque un *segnale analogico*
- Per codificarlo in digitale occorre:
  - *campionare* l'onda, ovvero osservarne il valore a un certo tempo T
  - *quantizzarla*, codificare il valore (analogico) campionato in digitale
- Ogni **immagine** sul display è rappresentata da una matrice di *pixels* che memorizzano
  - intensità luminosa
  - colore
  - altro

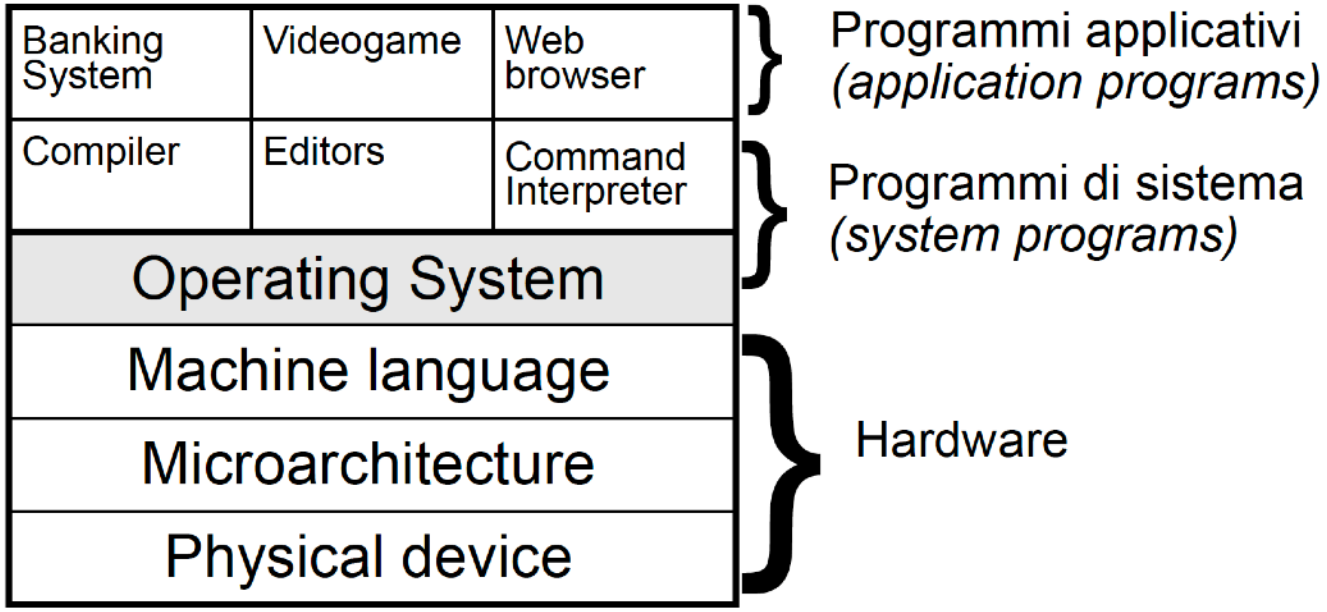




IL SISTEMA OPERATIVO

Definizione

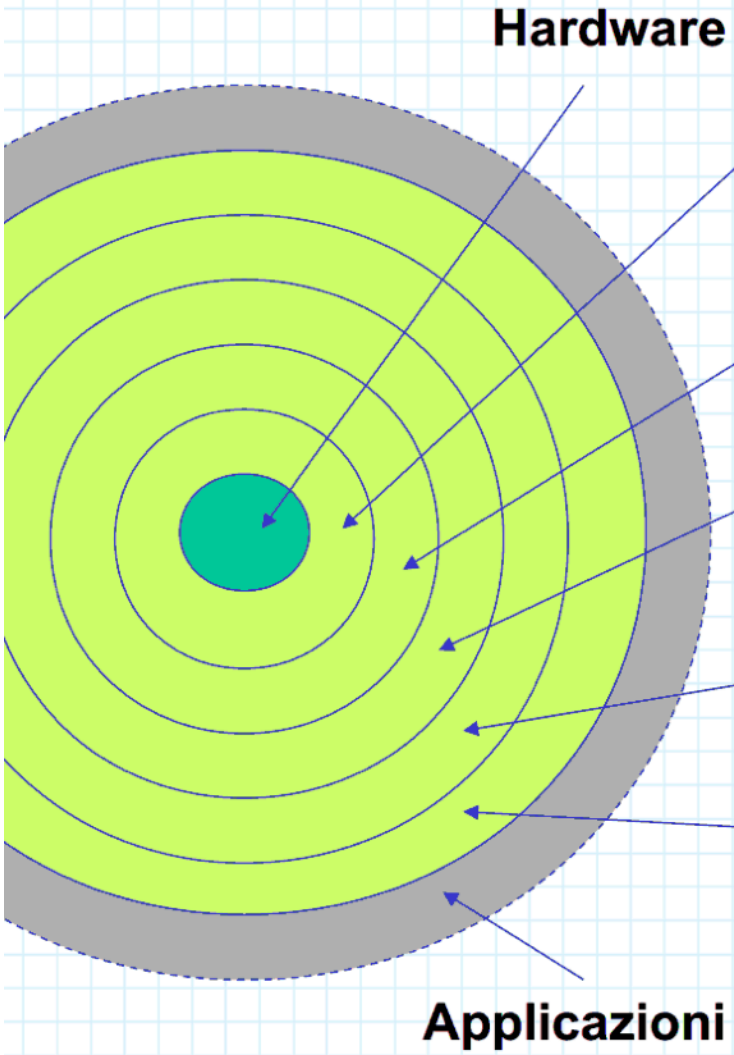
Il **sistema operativo** (S.O.) – in inglese operating system (O.S.) – è quella parte *software* di un sistema di elaborazione che *controlla* l'esecuzione dei programmi applicativi e funge da **intermediario** fra questi e la macchina fisica (*hardware*)



Obiettivi principali

- **Astrazione**, controllo, protezione
  - fornisce un'*interfaccia di programmazione* (**API**: Application Programming Interface) comune che facilita lo *sviluppo* e aumenta la *portabilità* dei programmi
  - *controlla* l'esecuzione dei programmi
  - *protegge* l'hardware e il software
- **Ottimizzazione**
  - rende *efficace* ed *efficiente* l'utilizzo delle risorse hardware
- **Coordinazione**
  - più *applicazioni* con concorrentemente
  - più *utenti* concorrentemente

Funzioni principali



Gestore processi (o nucleo): esecuzione programmi sulla CPU

Gestore memoria: allocazione memoria tra i vari programmi

Gestore periferiche: operazioni di I/O che coinvolgono le periferiche

Gestore file: archivi in memoria di massa

Interprete comandi (o shell): attivazione programmi da parte dell'utente

Gestione rete

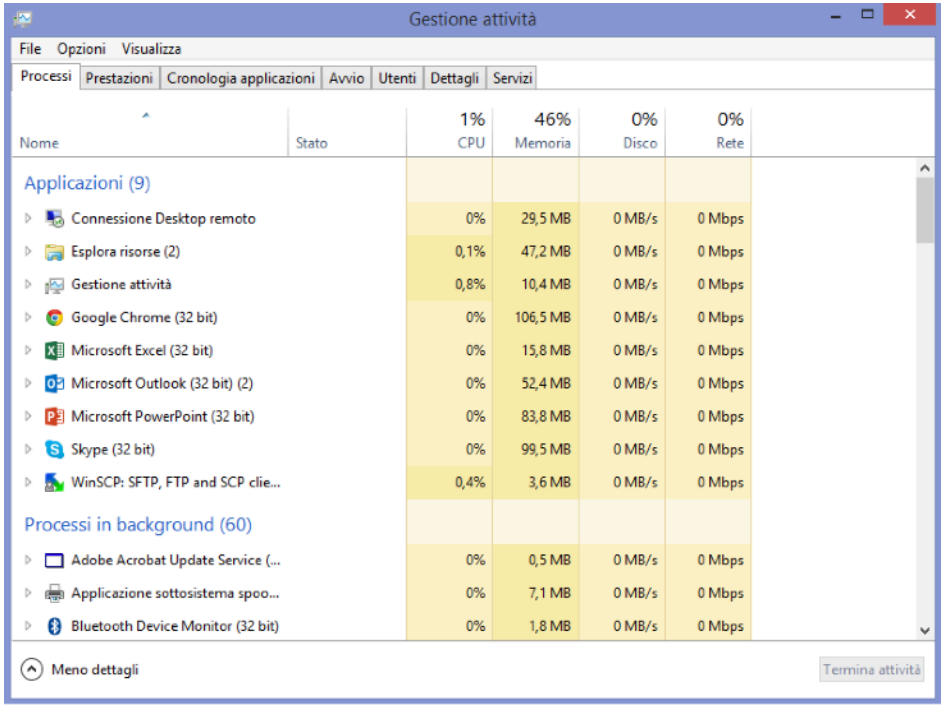
Protezione e sicurezza

Gestione utenti

Gestione processi

- L'astrazione con cui si definisce e identifica un *programma in esecuzione* prende il nome di **processo**

- S.O. **multi-tasking**: esegue più programmi *in contemporanea*
  - crea e distrugge i processi
  - decide a quale processo assegnare la CPU
  - sospende e riattiva i processi
- Inoltre offre meccanismi per:
  - *sincronizzazione* e gestione risorse *condivise*
  - *comunicazione* fra processi
  - comunicazione con periferiche



Gestione memoria

- Memoria **principale**
  - tenere *traccia* di quali zone di memoria sono attualmente usate e da quale processo
  - decidere quali *processi* caricare in memoria centrale quando c'è spazio disponibile
  - *allocare* / *deallocare* memoria a seconda delle richieste
- Memoria **secondaria**
  - gestione della *memoria virtuale*
    - memoria secondaria trattata *come se fosse* memoria principale*
  - *swapping* memoria principale – memoria virtuale
    - da memoria virtuale a memoria principale (e viceversa)

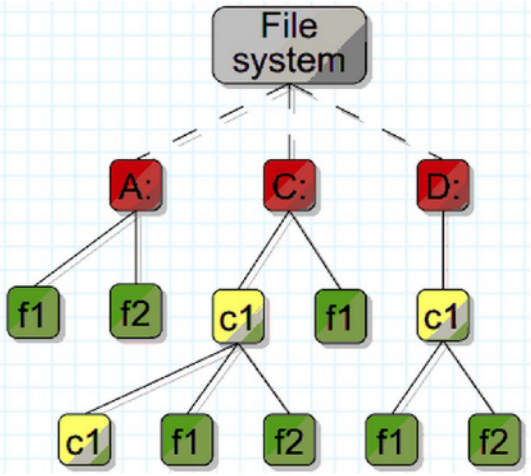
*Più o meno: è un argomento troppo dettagliato, chi vuole approfondire mi contatti

Gestione periferiche

- Il S.O. permette l'interazione tra i programmi in esecuzione e le **periferiche**
  - fornisce procedure I/O ad alto livello *mascherando* i dettagli hardware
- **BIOS** (Basic I/O System): software "sotto" al livello S.O. (firmware) per accesso a diversi tipi di periferica
- **Driver**: software che permette l'accesso ad una specifica periferica

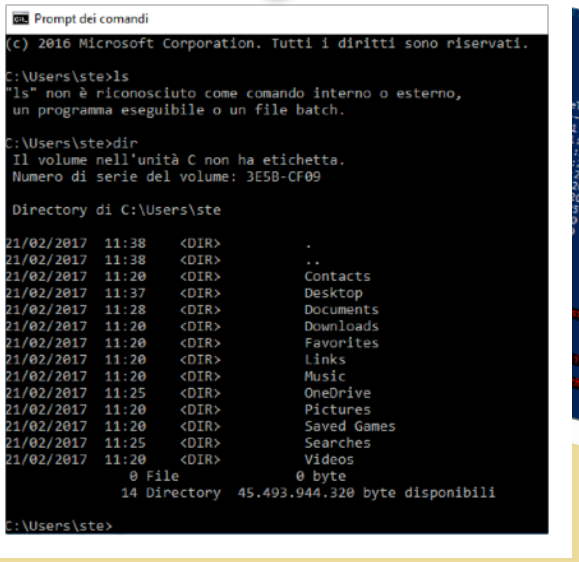
Gestione file(s)

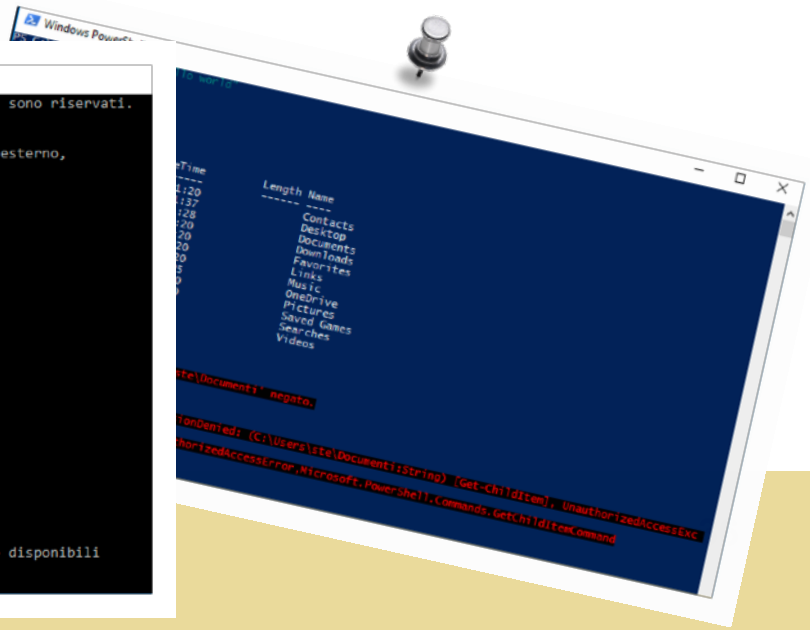
- Un **file** è una *collezione* di informazioni
  - programmi (in forma di sorgenti e binari)
  - puri dati (testo, immagini, video, etc.)
- Una **directory** è un file che funge da *contenitore (logico)* di file e altre directory
  - consentono di organizzare il file system in modo *gerarchico*
- Per **file system** si intende il sistema adottato per gestire i file su un dispositivo di memoria di massa
  - vari tipi di file system: FAT32, NTFS, ex-FAT, UFS, …
- La gestione dei file offre servizi per:
  - creazione e cancellazione di file / directory
  - manipolazione di file
  - mapping dei file in memoria secondaria



Interprete comandi

- Programma che permette all'utente di interagire con il sistema mediante comandi impartiti in *modalità testuale* (non grafica), via **linea di comando**
  - navigazione file system
  - interazione / manipolazione file
  - esecuzione programmi
  - controllo processi (programmi in esecuzione)
  - comandi per ispezionare/controllare lo stato del sistema





Gestione rete

- Un sistema di rete (**distribuito**) è una *collezione* di processori che *non condividono memoria*
  - ogni processore ha la propria memoria locale
- I processori in un sistema distribuito sono connessi mediante una **rete** di comunicazione
  - *protocolli* di comunicazione definiscono le *regole* con cui avviene la comunicazione
- Un sistema distribuito fornisce l'accesso ad un utente alle varie *risorse distribuite* per i nodi, consentendo
  - speed-up della computazione
  - migliore *availability* (disponibilità dei dati)
  - migliore *reliability* (affidabilità di accesso)

Principali S.O.

- Ambito desktop
  - **Microsoft** Windows (da NT)
    - Windows XP, Vista, 7, 8, 10
  - **Linux** e altri sistemi famiglia UNIX
    - Solaris, FreeBSD, Debian, Ubuntu, etc.
  - **Apple**: macOS
    - Unix-based + S.O. di ricerca sviluppato alla Carnegie Mellon University
- Ambito mobile (tablet, smartphone)
  - **iOS** (famiglia Mac OS X)
  - **Windows Phone** 7/8 (famiglia Microsoft)
  - Google **Android**
    - basata su Linux ed esecuzione programmi *Java-like*

INTERNET E IL WEB

Un po' di storia

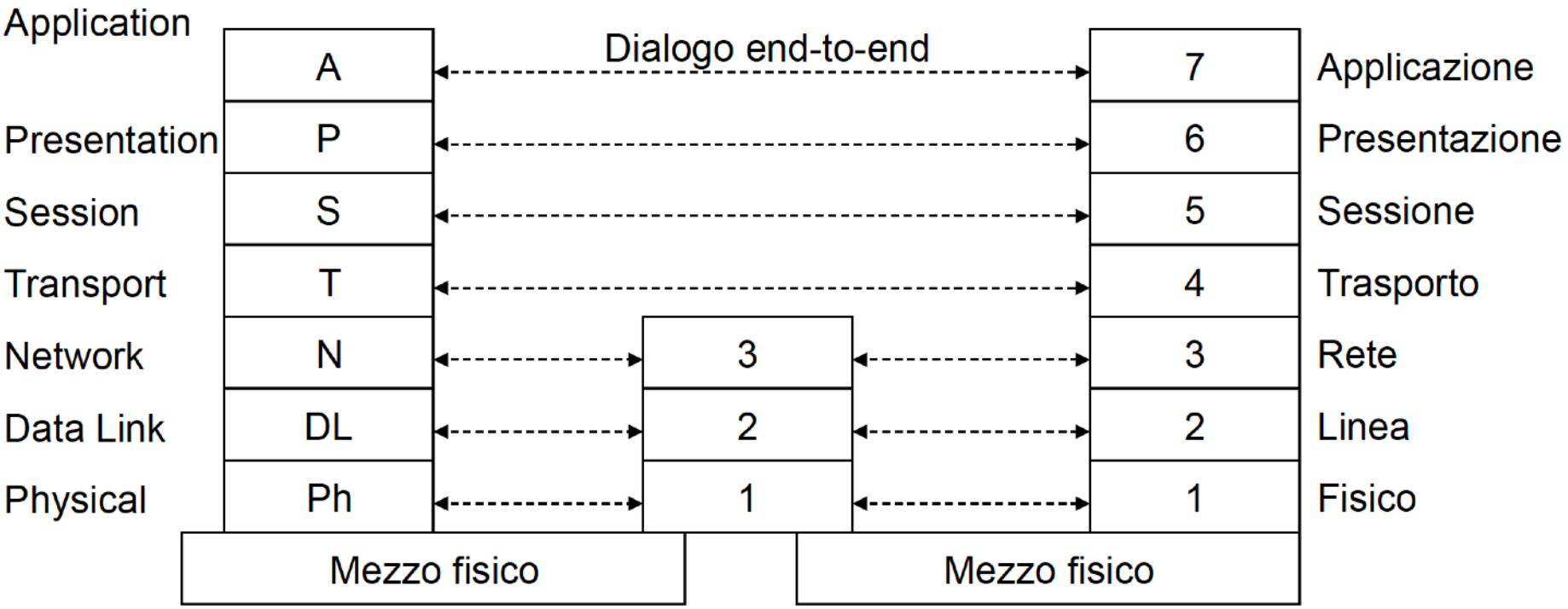
- **1969**: il dipartimento della difesa USA, attraverso l'Agenzia per i Progetti di Ricerca Avanzati (ARPA), finanzia la sperimentazione di una rete di calcolatori (**ARPANet**) fra: UCLA, Stanford, UCSB, Università dello Utah
- **1982**: ARPANet adotta come standard i protocolli **TCP/IP**
  - nasce la prima definizione di **Internet** come insieme di reti interconnesse tramite TCP/IP
- **1989**: l'Italia si connette a Internet
- **1992**: viene rilasciato dal CERN il servizio World Wide Web (**www**)

Un po' di storia

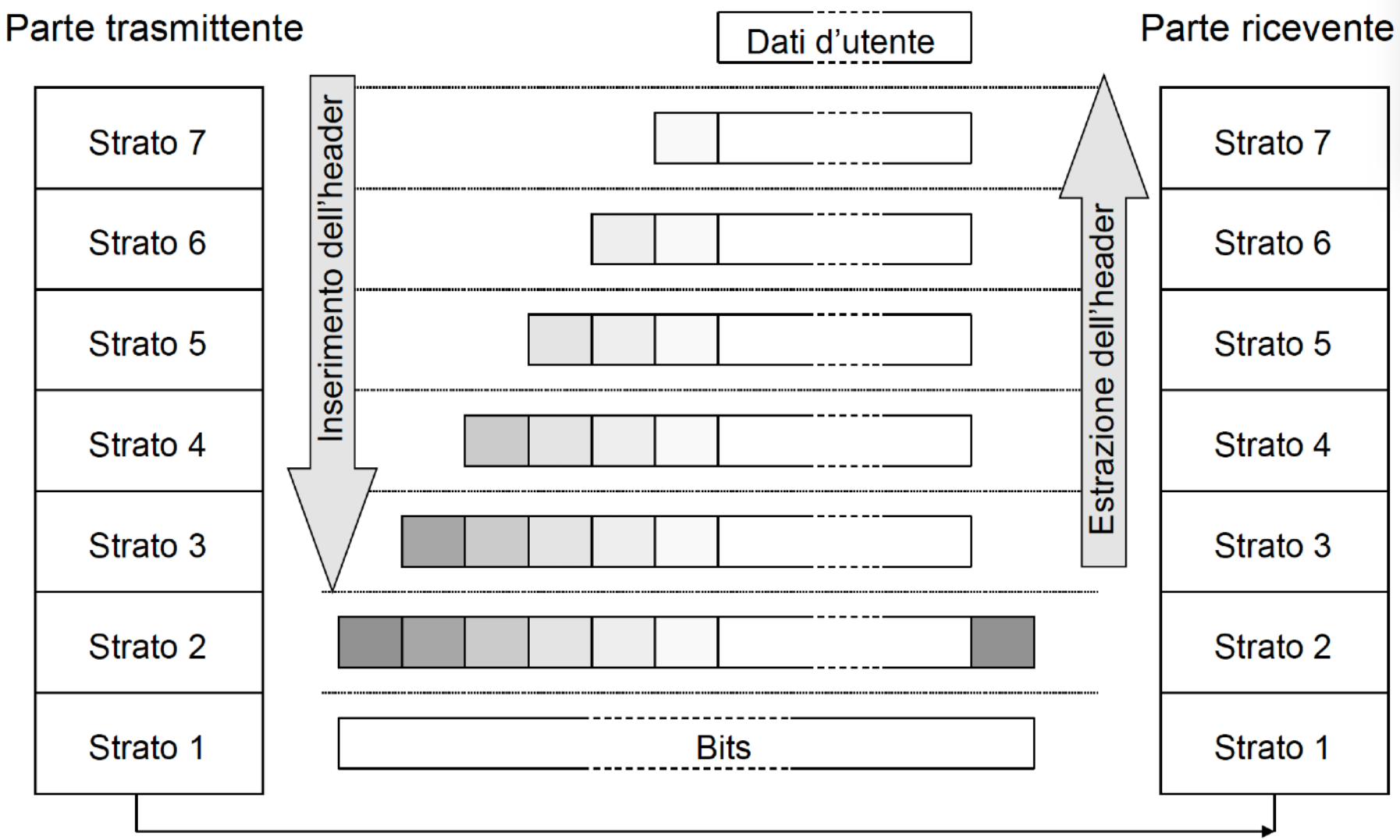
- A partire dal 1976 la **ISO*** ha dato il via a lavori per giungere ad una serie di **standard** unificati per la realizzazione di reti di calcolatori **aperte**
- Per prima cosa ha proposto un *modello di riferimento*
  - **OSI-RM** (Open System Interconnection Reference Model)
  - basato sul concetto centrale di *architettura a strati*
    - *scompone* il problema in sotto-problemi più semplici da trattare
    - rende i vari livelli *indipendenti*
    - definendo solamente servizi e *interfacce*

*International Standards Organisation

Il modello OSI-RM



Il modello OSI-RM



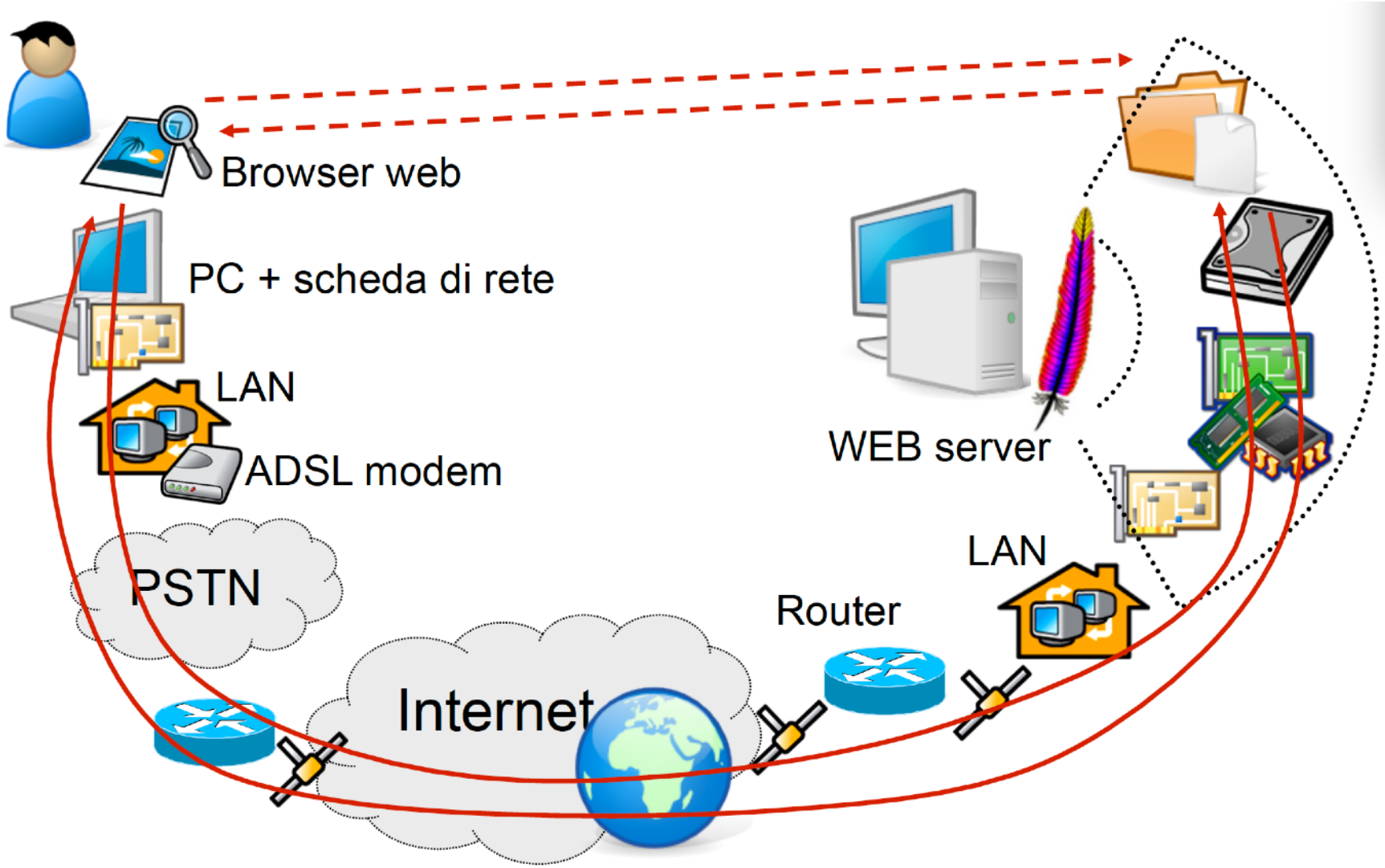
OSI vs. TCP/IP

OSI	TCP/IP	Protocolli
Application	Application	HTTP, TELNET, FTP, SMTP, POP, DNS, SNMP
Presentation		
Session		
Transport	Transport	TCP, UDP
Network	Network	IP, ICMP, IGMP, ARP, RARP
Data Link	Link	ETHERNET, IEEE 802, HDLC, PPP
Physical		

Client-Server

- Un utente (U) è interessato a reperire una **risorsa** (R) sul web
  - R = pagina web, immagine, video, file, …
- Apre il proprio **Browser** Web (B)
  - MS Edge, Safari, Chrome, Firefox, …
  - B è un'applicazione "**client**" agli occhi della rete
  - B deve sapere come cercare e reperire R
- R risiede su un **Server** Web (S)
  - S è un'applicazione "**server**" per la rete
  - S deve sapere come rispondere a B e consegnare R
- Ottenuta R il browser B la mostra a U

Client-Server



Indirizzamento

- R è *univocamente* identificata da un indirizzo chiamato *Uniform Resource Locator* (**URL**)
  - un indirizzo complesso che riflette l'*organizzazione a livelli* della rete

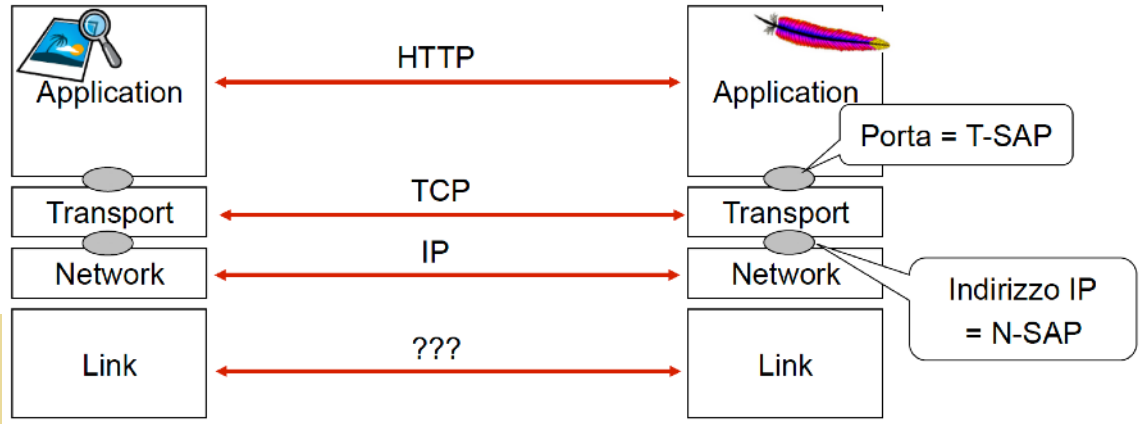
http://deisnet.deis.unibo.it:8080/prova/prova.html

Protocollo di dialogo con l'applicazione

Nome simbolico del server web: Indirizzo di rete (N-SAP)

Numero di porta: Indirizzo dell'interfaccia trasporto/ applicazione (T-SAP)

Percorso al documento nel file system del server

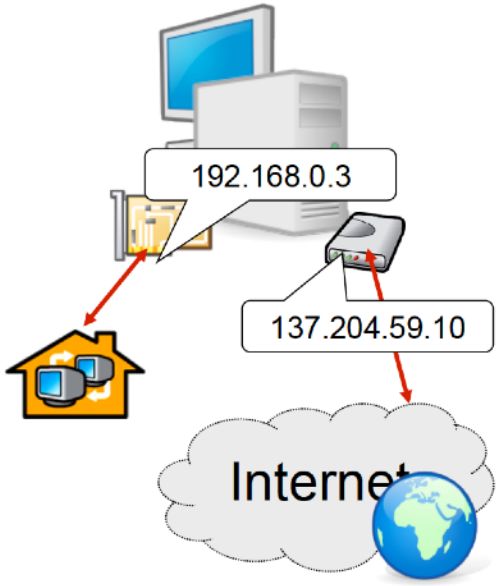


Indirizzo IP

- Indirizzi di lunghezza fissa pari a **32 bit**
  - scritti convenzionalmente come sequenza di 4 numeri decimali, con valori da 0 a 255, separati da punto (rappresentazione *dotted decimal*)

e.g. 155.185.28.83

- Numero *teorico* max. di indirizzi 2^{32} = 4.294.967.296*

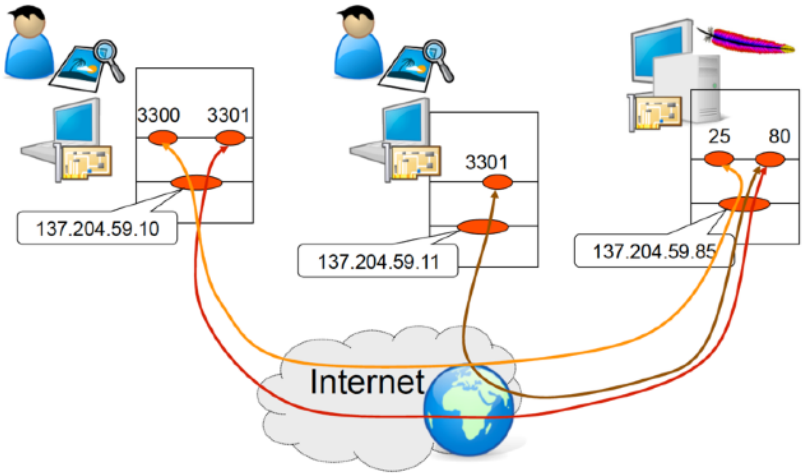


*vi faccio notare che ad oggi sarebbero già finiti, si usano tecniche particolari quali il *NAT* per evitarlo

Indirizzo TCP

- Gli standard non specificano come gli applicativi debbano interagire con i protocolli
- L'interfaccia fra applicazione e TCP
  - non è standardizzata
  - dipende dall'implementazione del sistema operativo
  - viene comunemente chiamata **Socket**
- Indirizzo della Socket (*Socket Address*) = numero di **porta** (concatenato ad un indirizzo di rete)

e.g. 155.185.28.83:80

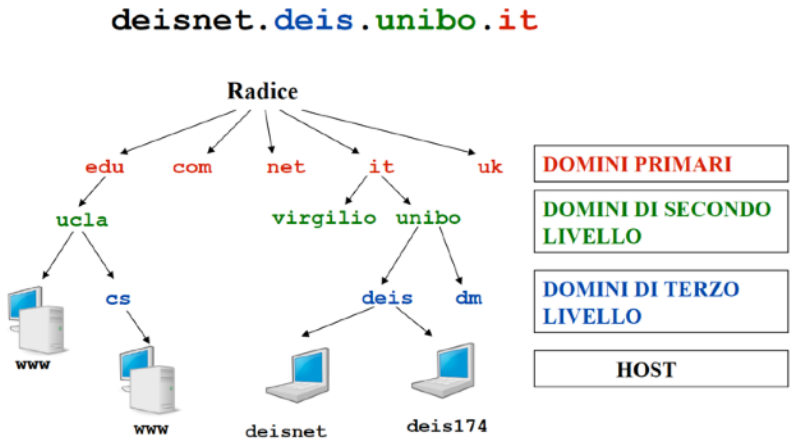


DNS

- Per comodità degli utenti agli indirizzi IP sono associati dei *nomi simbolici*
  - Sequenza di stringhe alfanumeriche separate da punti

    e.g. dolly.ingre.unimore.it

- Le componenti del nome riflettono un'organizzazione gerarchica in *domini* e *sotto-domini*
- *Domain Name System* (**DNS**) è un *database distribuito* che associa ad ogni nome il relativo indirizzo di rete
  - La consultazione del DNS avviene tramite opportuni server DNS in maniera *trasparente* (il browser sa come fare)



Fondamenti di Informatica

(L-Z)

Corso di Laurea in Ingegneria Gestionale

Introduzione all'Informatica

Prof. Stefano Mariani

Dott. Alket Cecaj