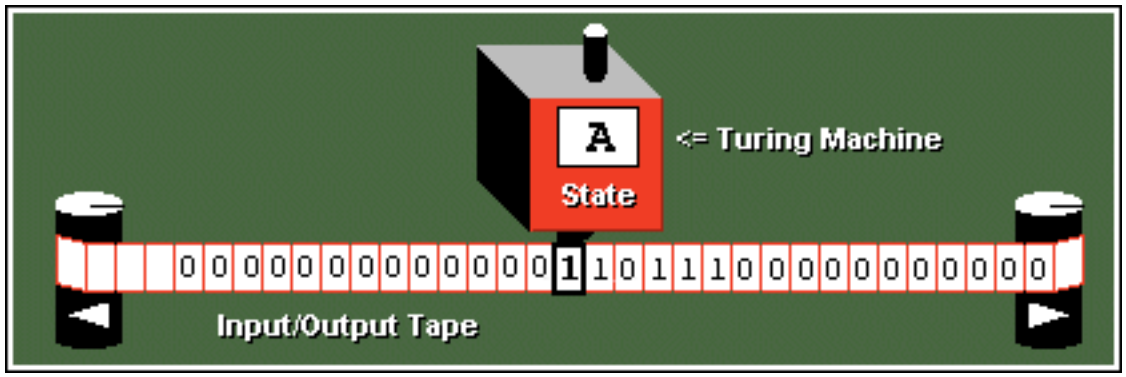
Fondamenti di Informatica e Laboratorio

Informatica e computer

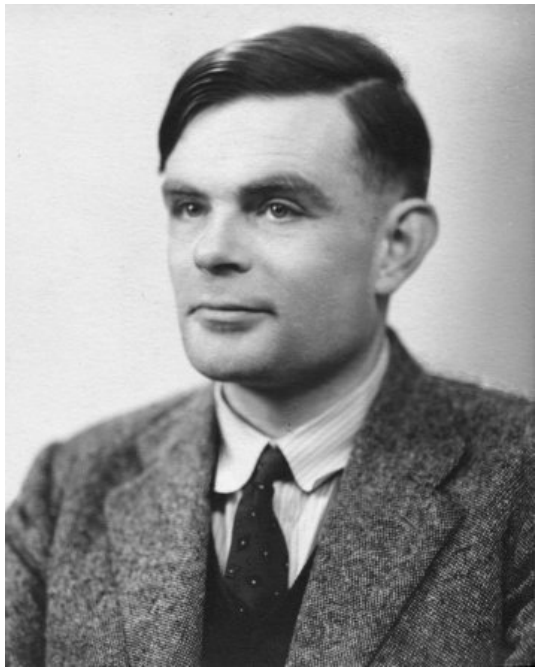
"L'informatica non riguarda i computer più di quanto l'astronomia riguardi i telescopi." (E. W. Dijkstra)

"Il computer non è una macchina intelligente che aiuta le persone stupide, anzi, è una macchina stupida che funziona solo nelle mani delle persone intelligenti." (U. Eco)

Informatica e computer



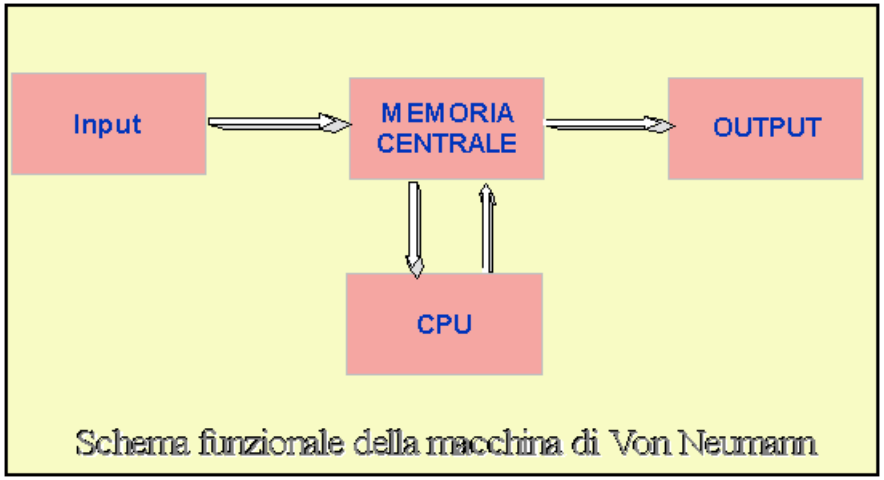
Cosa significa “calcolare”?
Quali problemi sono “risolvibili” tramite procedimenti di calcolo?



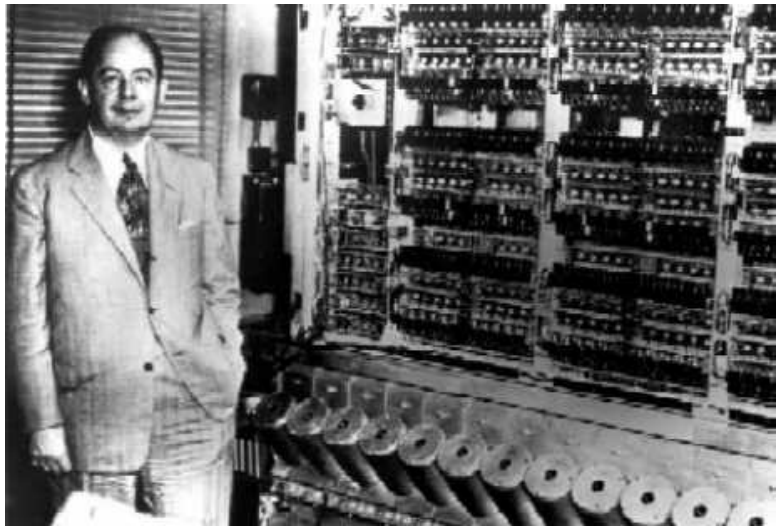
Alan Turing, anni 30’

Informatica e computer

Propone un modello di organizzazione di una macchina a programma memorizzato



Schema funzionale della macchina di Von Neumann



John Von Neumann, 1946

Problemi, algoritmi e programmi

Problema: descrizione della relazione tra i dati che rappresentano la situazione di partenza (input) e i dati che rappresentano la situazione desiderata (output).

Es. numeri disordinati (input) ⟶ numeri ordinati (output)

Algoritmo: sequenza di passi computazionali che trasforma l'input nell'output desiderato

Programma: codifica di un algoritmo in un linguaggio "comprensibile ed eseguibile" da un elaboratore

Obiettivi del corso

- Comprensione del mondo digitale
  - concetti di base
  - tecnologie
  - problematiche tecniche, sociali ed etiche

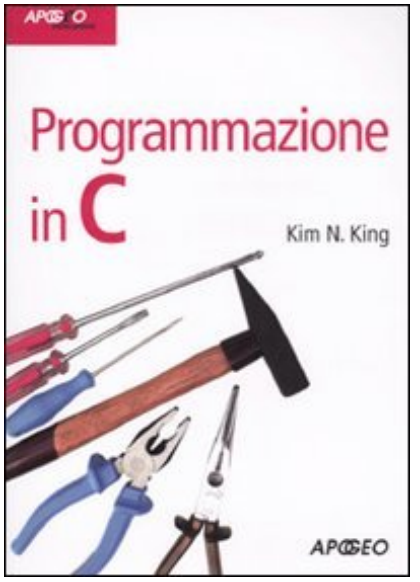
Obiettivi del corso

- Programmazione in linguaggio C

1. Fondamenti di C
2. Input/Output
3. Espressioni
4. Istruzioni condizionali
5. Cicli
6. Tipi di base
7. Array
8. Funzioni
9. Organizzazione programma
10. Puntatori
11. Puntatori e array
12. Stringhe
13. Preprocessore
14. Strutture
15. Uso avanzato dei puntatori
16. I/O e gestione dei file

Libri di testo

Informatica
Orientarsi nel mondo digitale
B.W. Kerninghan
Egea
ISBN: 9768823822733
276 pagine
Luglio 2019
traduzione in italiano

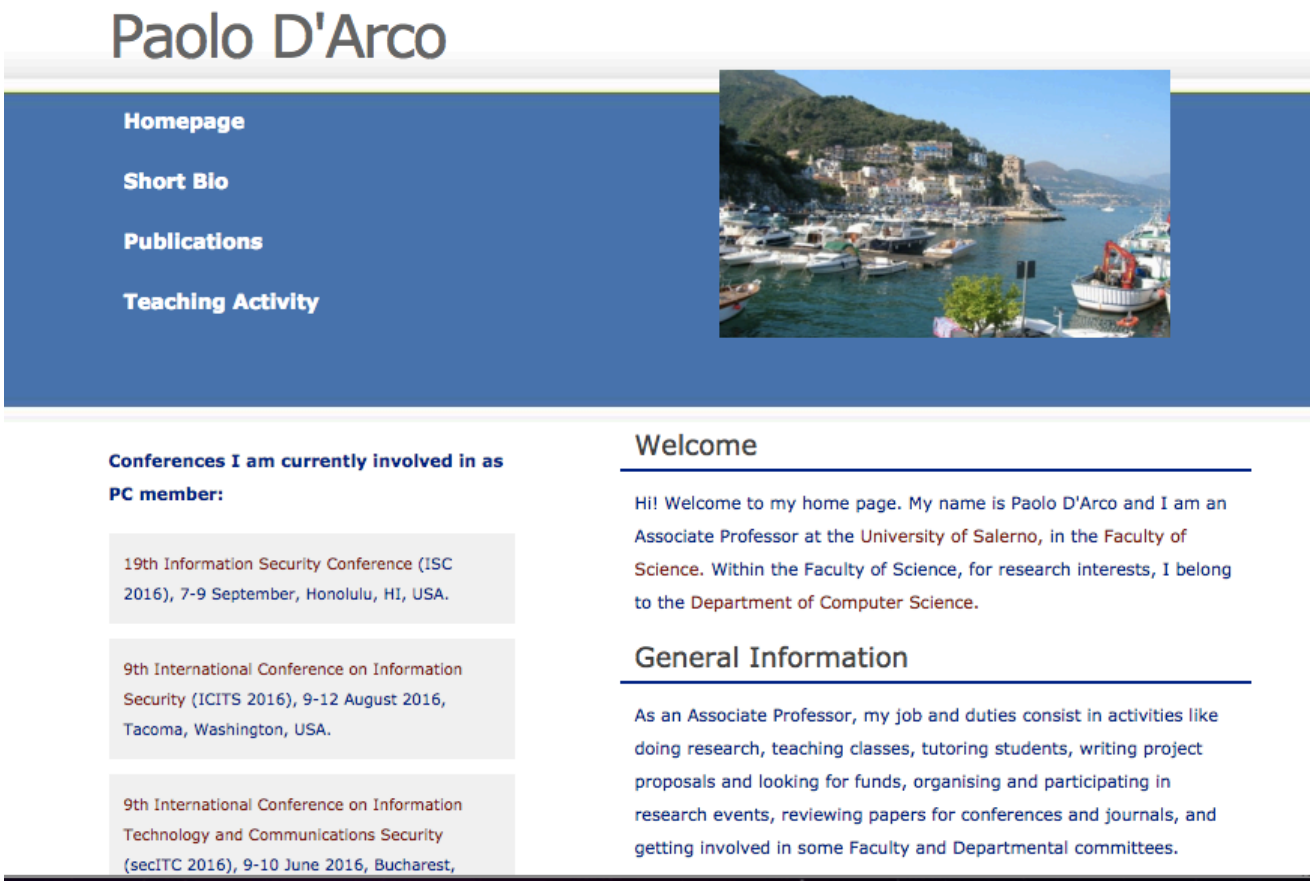


Programmazione in C
K.N. King
Apogeo
ISBN: 9788850328697
786 pagine
Settembre 2009
traduzione in italiano

Sito web

- Materiale didattico
  - Codice
  - Slide
- Informazioni
  - Programma
  - Libri
  - Annunci
- Esami
  - Date
  - Risultati

http://www.di.unisa.it/~paodar/



Informazioni

- Prof. Paolo D'Arco
  - studio: Edificio F, 4° piano, numero 45
    - Dipartimento di Informatica
  - email: pdarco@unisa.it
- Orario di ricevimento (virtuale)
  - Martedì 11:00-12:30
  - Giovedì 11:00-12:30

Orario delle lezioni (on-line, in futuro aula F3):

- Martedì, 9:00 - 11:00
- Venerdì, 11:00 - 13:00

Prove d'esame

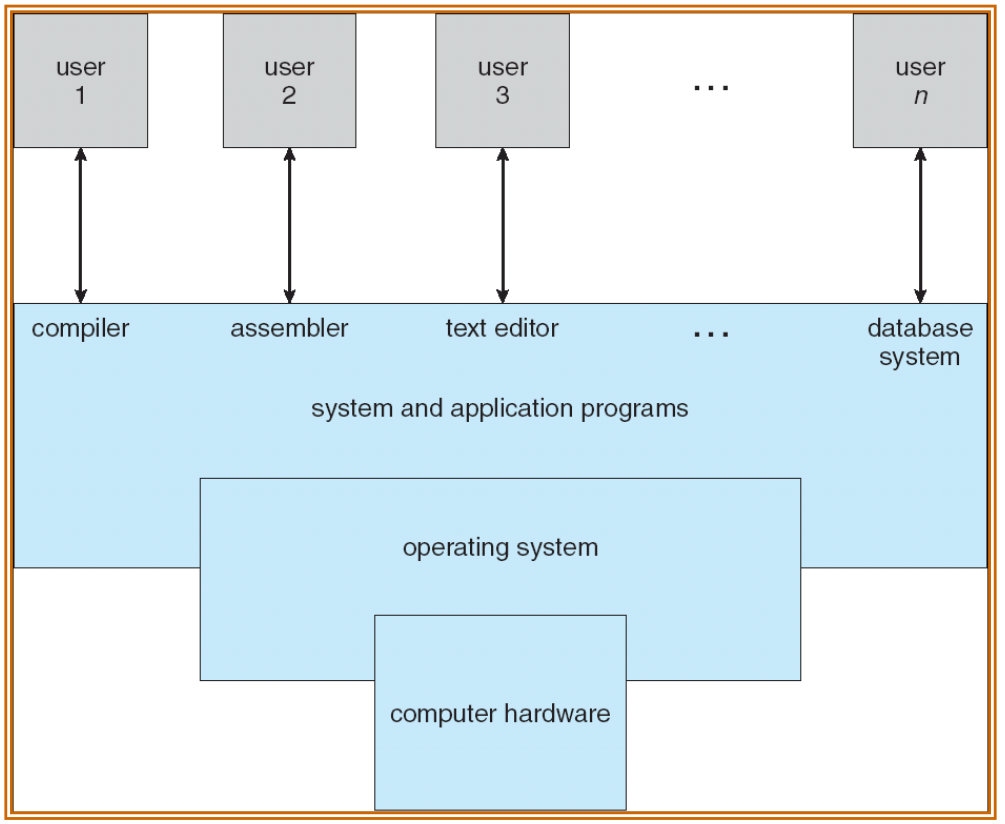
- Prova scritta / prova pratica
  - Sviluppo di un programma in C
- Prova orale
  - Discussione del programma sviluppato e domande di cultura informatica

48 ore insieme …

Cominciamo con l'acquisire familiarità con un sistema di calcolo e le sue componenti

Elementi di un sistema di calcolo

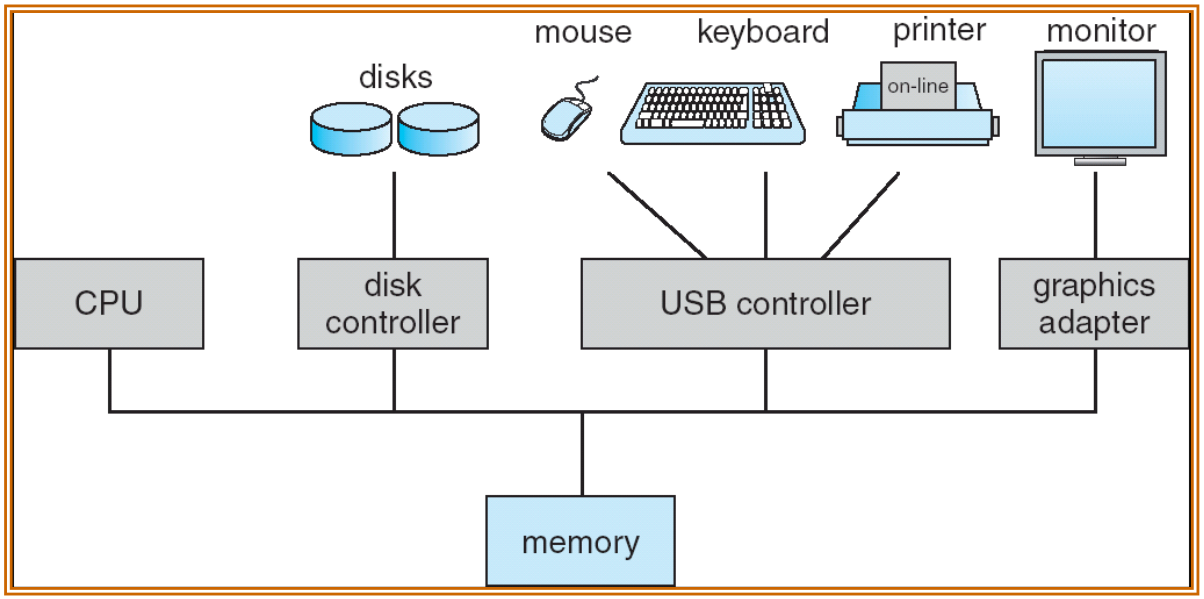
- **Hardware** – fornisce risorse computazionali di base
  - CPU, memoria, I/O device
- **Sistema Operativo**
  - Controlla e coordina l'uso dell'hardware tra applicazioni e utenti
- **Programmi** – definiscono i modi attraverso i quali le risorse del sistema vengono usate per risolvere problemi computazionali degli utenti
  - word processor, compiler, web browser, database, video game
- **Utenti**
  - Persone, dispositi, altri computer



Organizzazione di un elaboratore
Componenti e meccanismi

Struttura di un elaboratore

- Modalità di funzionamento
  - Le CPU e controller di dispositivi sono connessi ad un bus comune che fornisce accesso alla memoria condivisa
  - Le CPU e i controller dei dispositivi competono per ottenere cicli di accesso alla memoria



Struttura di un elaboratore

- I dispositivi di I/O e la CPU lavorano concorrentemente.
- Ciascun controller gestisce un particolare tipo di dispositivo
- Ogni controller possiede un buffer locale
- La CPU trasferisce dati dalla/alla memoria in/da buffer locali
- I/O avviene tra i dispositivi e i buffer locali dei controller
- I controller dei dispositivi informano la CPU che hanno finito il proprio lavoro generando un *interrupt*.

Bootstrap

- **Un programma di bootstrap** viene caricato quando il computer viene accesso o viene riavviato
  - Tipicamente è memorizzato in una ROM o in una EEPROM (**firmware)**
  - Inizializza tutte le funzioni principali del sistema, dai registri della CPU ai controller della memoria
  - Carica il kernel del sistema operativo e comincia l'esecuzione

Il kernel aspetta che si verifichino **eventi** o richieste degli utenti da eseguire

Interrupt driven

Gli eventi sono segnalati da **interrupt** o da eccezioni (**trap**).

- Per ogni tipo di interrupt, segmenti separati del codice del SO (**routine di gestione dell'interrupt**) determinano le azioni da intraprendere per gestire l'evento.

- Una **trap** è un interrupt generato dal software, causato o da un *errore* durante la computazione o da una *richiesta specifica* dell'utente (chiamata di sistema).

Struttura della memoria centrale

•La memoria centrale (RAM) è una **sequenza di parole** a cui il processore può accedere direttamente attraverso il bus.

•Ogni parola ha un proprio **indirizzo**.

•L'interazione avviene tramite istruzioni **load** e **store**.

•La memoria contiene istruzioni e dati, ma l'unica cosa che vede è un **flusso** di indirizzi.

•È volatile.

Memoria Centrale
Random Access Memory

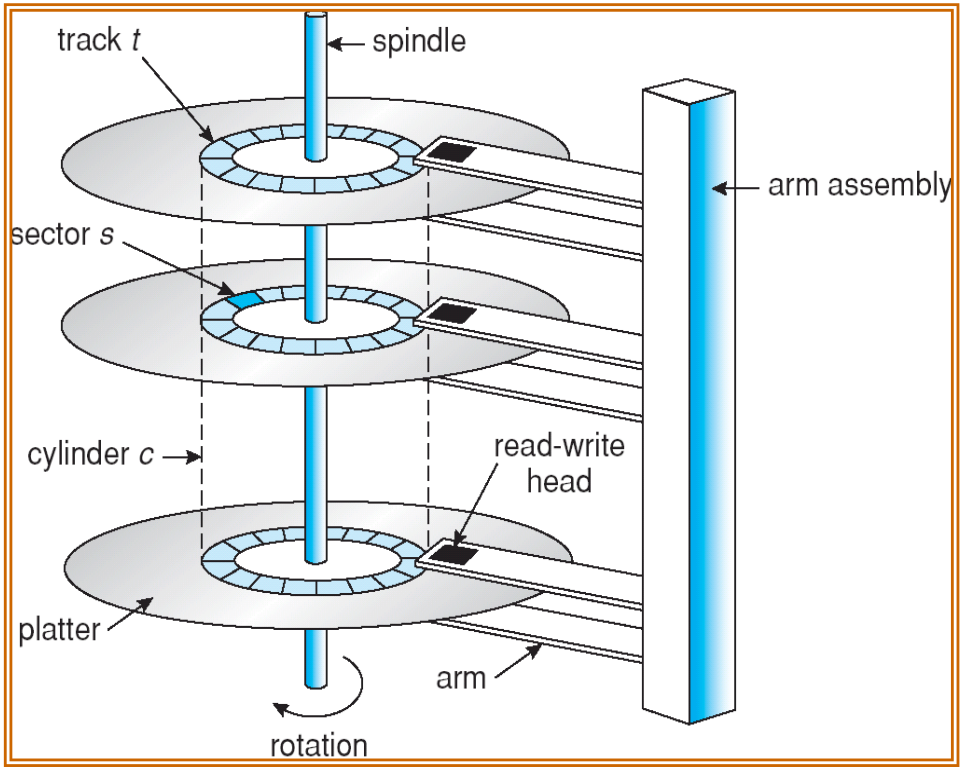
Struttura della memoria secondaria

Memoria secondaria. Disco magnetico

Un disco è composto da piatti

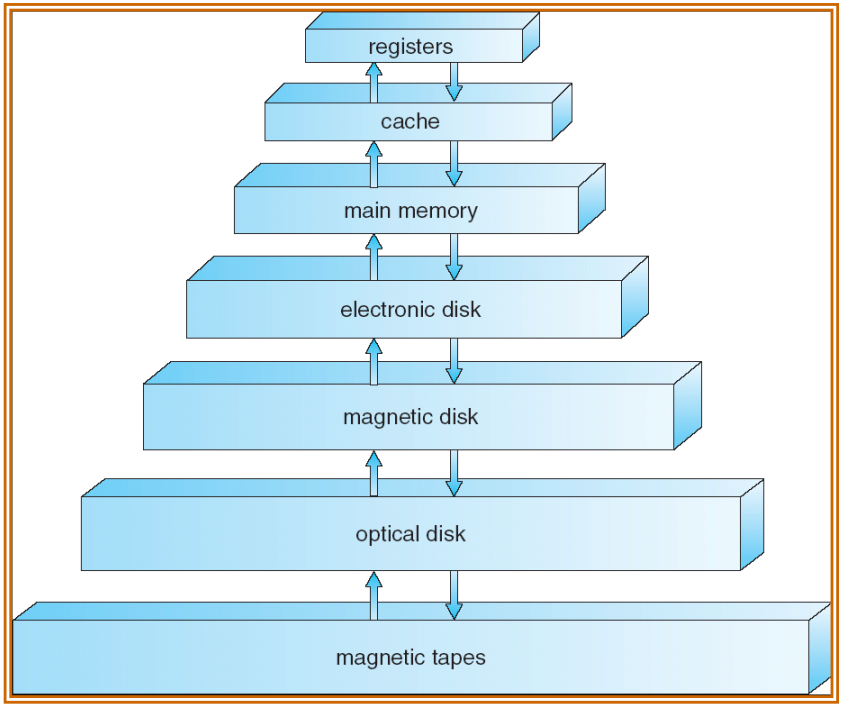
Una testina di lettura - scrittura sfiora ogni piatto

Ogni piatto è diviso in **tracce**. Ogni traccia in **settori**. L'insieme delle tracce che si trova sotto un braccio forma un **cilindro**.



Gerarchia di memorizzazione

- I sistemi di memorizzazione sono organizzati *gerarchicamente*.
  - Velocità
  - Costo
  - Volatilità



Unità di misura

- **b** bit, valore 0/1
- **B** byte = 8 bit
- **Kb** Kilobit = 1024 bit
- **Mb** Megabit = 1024^2 bit
- **Gb** Gigabit = 1024^3 bit
- **KB** Kilobyte = 1024 byte
- **MB** Megabyte = 1024^2 byte
- **GB** Gigabyte = 1024^3 byte
- **TB** Terabyte = 1024^4 byte

- **ms** millisecondo = 10^{-3} sec
- **µs** microsecondo = 10^{-6} sec
- **ns** nanosecondo = 10^{-9} sec

Architetture

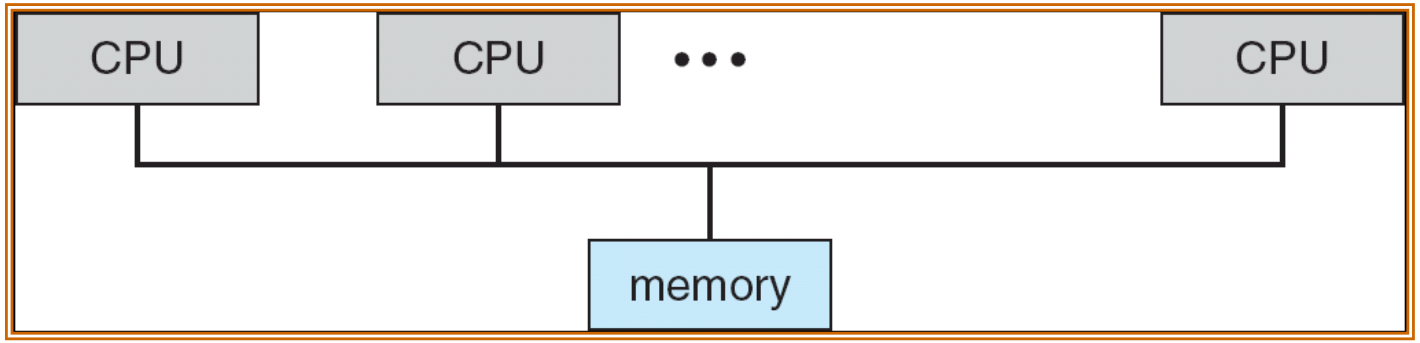
Architetture a singolo processore

Tali sistemi sono dotati di un singolo processore che esegue un set di istruzioni general-purpose

Spesso usano anche processori special purpose, quali processori di I/O, che muovono velocemente dati tra le componenti (e.g., disk-controller, processori associati alle tastiere).

A volte la CPU principale comunica con questi processori. Altre volte, essi sono totalmente autonomi.

Architetture multiprocessore



Sistemi multiprocessore

Questi sistemi, anche detti paralleli o con processori strettamente accoppiati, posseggono più processori che **condividono il bus del computer, il clock, la memoria e le periferiche**.

- Maggiore quantità di elaborazione effettuata
- Economia di scala
- Aumento affidabilità

Esistono due tipi di sistema multiprocessore:

- **sistema multiprocessore asimmetrico** – un processore principale (master) organizza e gestisce il lavoro per gli altri (slave)

- **sistema multiprocessore simmetrico** (SMP) – ogni processore esegue una copia del sistema operativo e, tali copie, comunicano tra loro

CPU Multi-core e Server Blade

Un microprocessore **multi-core** combina due o più processori indipendenti su un singolo supporto, spesso un singolo circuito integrato

- Un **server blade** (a lama) è essenzialmente un alloggiamento per schede madri, ciascuna contenente uno o più processori, memoria centrale, e connessioni di rete, che condividono il sistema di alimentazione e di raffreddamento dell'intera infrastruttura e le memorie di massa

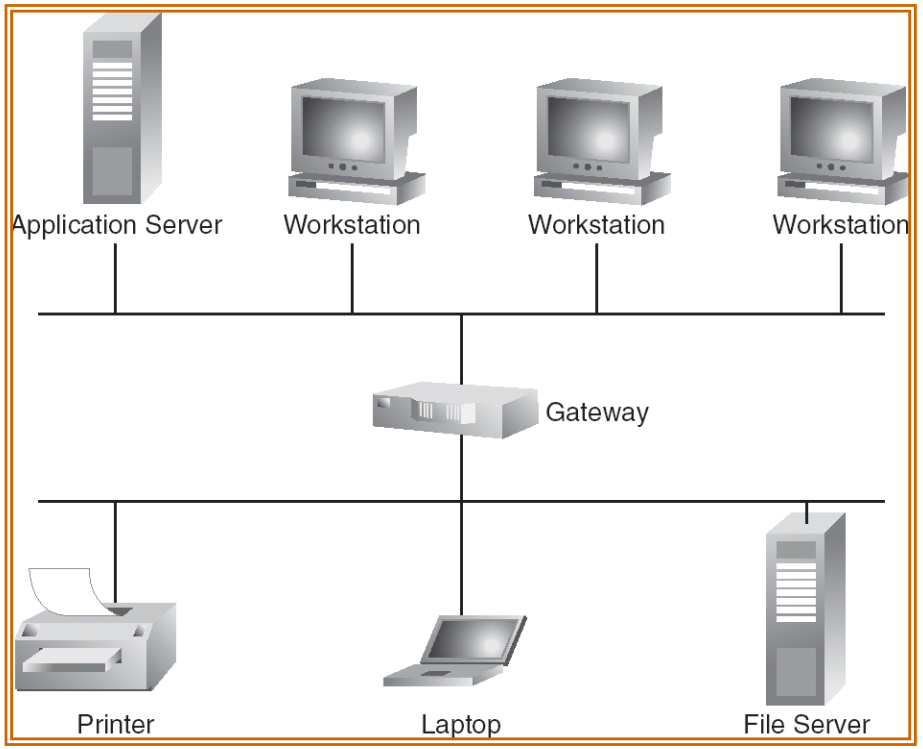
Struttura di rete

Reti locali (LAN)

Le velocità di comunicazione variano da 1 Mb a 10 Gb

Comuni sono:

10BaseT Ethernet	10Mb
100BaseT Ethernet	100Mb



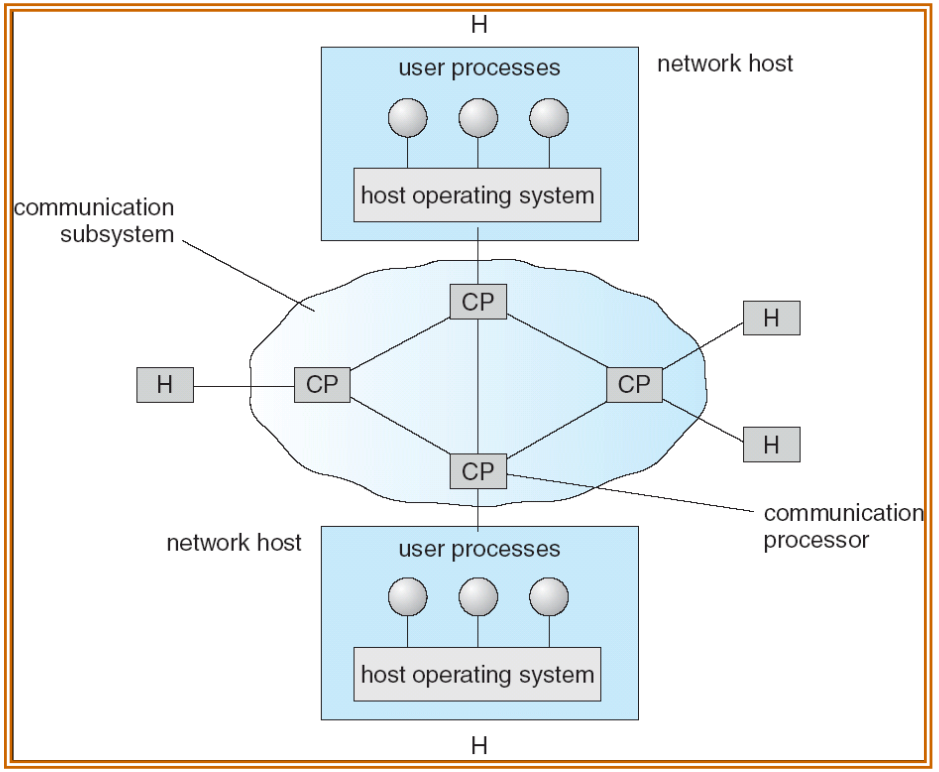
Struttura di rete

Reti geografiche (WAN)

Le velocità di comunicazione variano da 56Kb a diversi Mb

Le connessioni sono controllate da CP, communication processor

Internet: router, linee T1. Linee telefoniche, modem, ADSL.



Che cos'è un sistema operativo?

È un *programma* che opera da *intermediario* tra l'utente e l'hardware del computer

Assicura che il computer operi *correttamente* e che le risorse siano usate *efficientemente*

Esegue i programmi degli utenti e *facilita* i loro compiti offrendo un ambiente d'uso *conveniente*

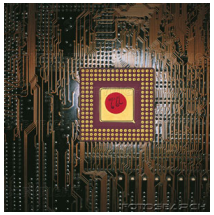
Ruolo del Sistema Operativo

PC. Il sistema operativo è progettato principalmente per facilitare l'uso del computer.

Mainframe e Minicomputer. Occorre massimizzare l'uso delle risorse.

Workstation. Compromesso ottimale tra uso risorse individuali e risorse condivise.

Palmari e simili. Progettati per l'uso individuale con attenzione alle prestazioni della batteria



Sistemi Embedded. Concepiti per funzionare senza l'intervento dell'utente

Visione del sistema

Il **S**istema **O**perativo (**SO** in breve) è il programma più intimamente connesso con l'hardware. Quindi, è:

- **allocatore di risorse**: di fronte a richieste conflittuali, decide come assegnare *equamente ed efficientemente* le risorse ai programmi

- **programma di controllo**: garantisce l'esecuzione dei programmi senza errori e usi impropri del computer

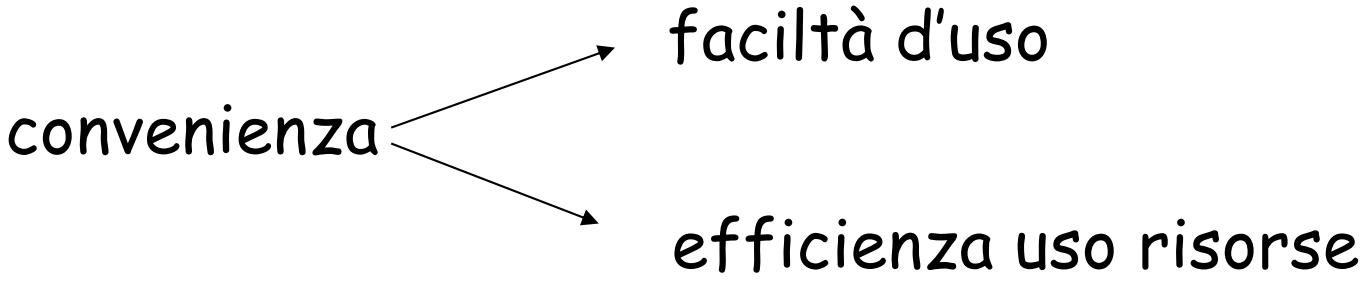
- **esecutore di funzioni comuni**: esegue funzioni di utilità generale comuni ai diversi programmi (e.g. routine di I/O)

Definizione di Sistema Operativo

- Non esiste una definizione universalmente accettata
- "Tutto ciò che il venditore ti invia quando ordini un sistema operativo" - è una buona approssimazione ma varia ampiamente
- "Il programma che è sempre in esecuzione sul computer" - **kernel del SO**. Tutto il resto è o un programma di sistema o un programma applicativo.

Sistema Operativo:cos'è e cosa fa?

I sistemi operativi esistono perché forniscono agli utenti uno strumento *conveniente* per l'uso di un sistema di calcolo



Esempi di sistemi operativi: Unix, Windows, Linux, OS X (Mac) …

Struttura di un sistema operativo

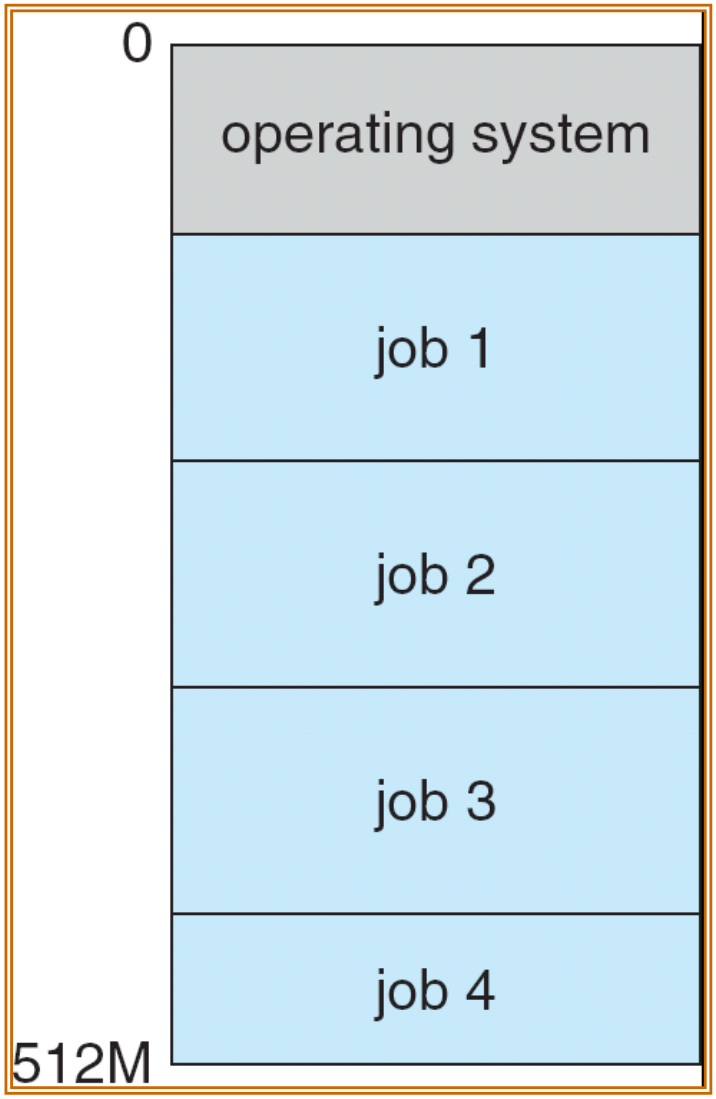
Idee di base e componenti principali

Concetti chiave

Terminologia: job – processo - task

- **Multiprogrammazione** - necessaria per efficienza
  - Un solo job non può tenere CPU e dispositivi I/O occupati per tutto il tempo
  - Esegue più job e la CPU è sempre impegnata con uno di essi
  - Un sottoinsieme dei job si trova in memoria centrale
  - Un job viene selezionato (**job scheduling)** ed eseguito
  - Quando attende (e.g., operazione di I/O), il SO esegue un altro job

Immagine della memoria in un sistema multiprogrammato



Interfaccia utente CLI

Le interfacce a linea di comando CLI permettono l'invio diretto di comandi

- Caratteristiche multiple e opzioni – shell
  - A volte i comandi sono built-in (implementati nel kernel), altre sono semplici nomi di programmi
  - Con il secondo approccio, l'aggiunta di nuove caratteristiche non richiede la modifica della shell

Unix: Bourne Shell, C Shell, Bourne-again Shell, Korn Shell

Interfaccia utente GUI

- Le interfacce grafiche forniscono desktop amichevoli
  - I comandi sono forniti tramite mouse, tastiera, monitor
  - Icone rappresentano file, programmi, azioni …
  - La pressione dei tasti del mouse sugli oggetti dell'interfaccia causa varie azioni (recupero informazioni, opzioni, esecuzioni di funzioni, apertura di directory)
  - Inventate a Xerox PARC, comuni con Apple Mac

- Molti sistemi oggi includono interfacce sia CLI che GUI
  - Microsoft Windows offre una GUI ed una CLI
  - Apple Mac OS X offre una GUI che poggia su un kernel UNIX, e mette a disposizione le shell UNIX
  - Solaris offre una CLI e opzionalmente GUI (Java Desktop, KDE)

Sviluppo di codice

- Sistema operativo: Linux
- Compilatore C

Alternative?

Windows?

- Dev-C++ [IDE non consigliati…]

Capitolo 1

Introduzione al C

Linguaggio di programmazione

- Le "parole" sono delle istruzioni
- Le "frasi" hanno un significato
  - rispettano regole sintattiche
- Permettono di creare i ***programmi***
- Sufficientemente di "alto livello"
  - facili da usare per i programmatori
  - i programmi devono essere tradotti in "linguaggio macchina" per poter essere eseguiti

Un esempio di programma C

```
#include "stdio.h"

int main() {
     printf("Ciao mondo!\n");
     return 0;
}
```

Ogni CPU offre un insieme di istruzioni

Per esempio l'operazione di somma dei registri 1 e 2 con memorizzazione del risultato nel registro 6

```
[  op   |  rs |  rt |  rd |shamt| funct]
    0      1     2     6     0     32      forma decimale
 000000 00001 00010 00110 00000 100000    forma binaria
```

Il codice 000000 corrisponde all'istruzione di somma.
Può essere rappresentato simbolicamente con ADD.

I codici delle istruzioni costituiscono il "linguaggio macchina".
I codici simbolici costituiscono il "linguaggio assembly".

Un programma scritto in assembly deve essere "tradotto" in linguaggio macchina per poter essere eseguito.

```
MODEL SMALL
STACK 100H
.DATA
    HW      DB      "hello, world", 13, 10, '$'
.CODE
.STARTUP
    MOV AX, @data
    MOV DS, AX
    MOV DX, OFFSET HW
    MOV AH, 09H
    INT 21H
    MOV AX, 4C00H
    INT 21H
END
```

… in linguaggio assembly

… in linguaggio macchina 

```
00100111101111011111111111100000
10101111101111110000000000010100
10101111101001000000000000100000
10101111101001010000000000100100
10101111101000000000000000011000
10101111101000000000000000011100
10001111101011100000000000011100
10001111101110000000000000011000
00000001110011100000000000011001
00100101110010000000000000000001
00101001000000010000000001100101
10101111101010000000000000011100
00000000000000001111000000010010
00000011000011111100100000100001
00010100001000001111111111110111
10101111101110010000000000011000
00111100000001000001000000000000
10001111101001010000000000011000
00001100000100000000000011101100
00100100100001000000010000110000
10001111101111110000000000010100
00100111101111010000000000100000
00000011111000000000000000001000
00000000000000000001000000100001
```

Le origini del linguaggio C

- Il linguaggio C è un prodotto collaterale di UNIX
  - Sviluppato nei *Bell Laboratories* da Ken Thompson, Dennis Ritchie, e altri.
  - Sistema operativo per il computer DEC PDP-7 (1969)

- UNIX era stato scritto in *assembly*

Le origini del linguaggio C

- Programmi sviluppati in assembly difficili
  - da scrivere
  - da debuggare (controllo degli errori)
  - da migliorare
- Thompson decise che era necessario un linguaggio di livello superiore
  - ha progettato un linguaggio ridotto chiamato B.
- B era basato su BCPL
  - Basic Combined Programming Language
  - Linguaggio sviluppato negli anni 60
- Thompson riscrisse una parte di UNIX in B

Le origini del linguaggio C

- Nel 1971, Ritchie iniziò a sviluppare una versione estesa di B
  - per poter sfruttare un nuovo computer, PDP-11
- Inizialmente chiamò questa versione estesa NB ("New B")
- Poichè questa nuova versione cominciò ad essere abbastanza diversa da B, cambiò il nome in C.
- Nel 1973 il linguaggio C era diventato sufficientemente potente ed affidabile al punto da poter riscrivere l'intero sistema operativo UNIX in C

Standardizzazione del C

- *K&R* (Kernighan and Ritchie) *C*
  - *The C Programming Language* (1978)
  - De facto standard
- *C89/C90*
  - Standard ANSI X3.159-1989
  - Completato nel 1988; approvato formalmente nel 1989
  - Standard internazionale ISO/IEC 9899:1990
- *C99*
  - Standard internazionale ISO/IEC 9899:1999
    - es. di differenza: dichiarazioni non devono essere necessariamente fatte all'inizio di un blocco
- *C11*
  - Standard internazionale ISO/IEC 9899:2011
    - es di differenza: la funzione gets() è deprecata; sostituita da gets_s()

Linguaggi basati sul C

- ***C++***
  - Include tutte le caratteristiche del C e aggiunge il supporto per la programmazione ad oggetti
- ***Java***
  - Si basa su C++ e quindi eredita molte caratteristiche del C
- ***C#***
  - È un linguaggio recente basato su C++ e Java
- ***Perl***
  - Utilizza molte caratteristiche del C

Proprietà del linguaggio C

- Programmazione a basso livello
  - Programmazione di sistema
  - Controllo diretto della macchina
- Contenuto
  - Fornisce un insieme minimale di istruzioni e funzioni
  - Le librerie estendono il linguaggio
- Permissivo
  - Accesso diretto alla memoria
  - Il programmatore deve sapere cosa sta facendo

Aspetti positivi del C

- Efficiente
- Portabile
- Potente
- Flessibile
- Libreria Standard
  - Contiene centinaia di funzioni
    - I/O, stringhe, memoria, etc.
- Integrazione con UNIX

Debolezze del C

- È facile fare errori di programmazione
  - Il linguaggio è molto "permissivo"
- Può essere difficile capire i programmi
  - Stile di programmazione
- Può essere difficile modificare i programmi

Per un uso efficiente del C

- Evitare gli errori comuni
  - ne indicheremo molti durante il corso
- Sfruttare il codice già scritto per le librerie
- Adottare un insieme pratico e funzionale di convenzioni di programmazione
- Evitare "trucchi di programmazione" e codice eccessivamente complesso
- Attenersi allo standard (per la portabilità)

… arrivederci alla prossima lezione