Informatica Grafica

Corso di Laurea in Ingegneria Edile – Architettura

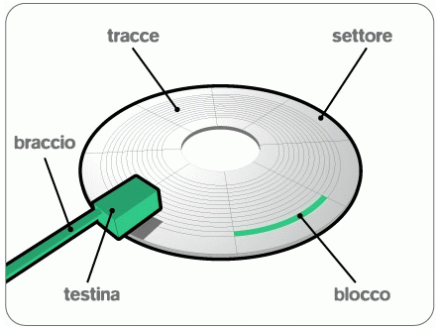
Architetture degli elaboratori e sistemi operativi

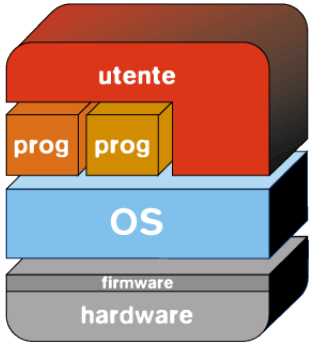
Paolo Torroni

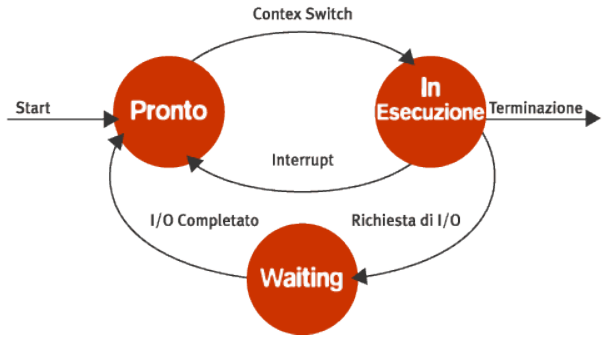
Dipartimento di Elettronica, Informatica e Sistemistica (DEIS)
Università degli Studi di Bologna

Anno Accademico 2011/2012

Architetture degli elaboratori e sistemi operativi





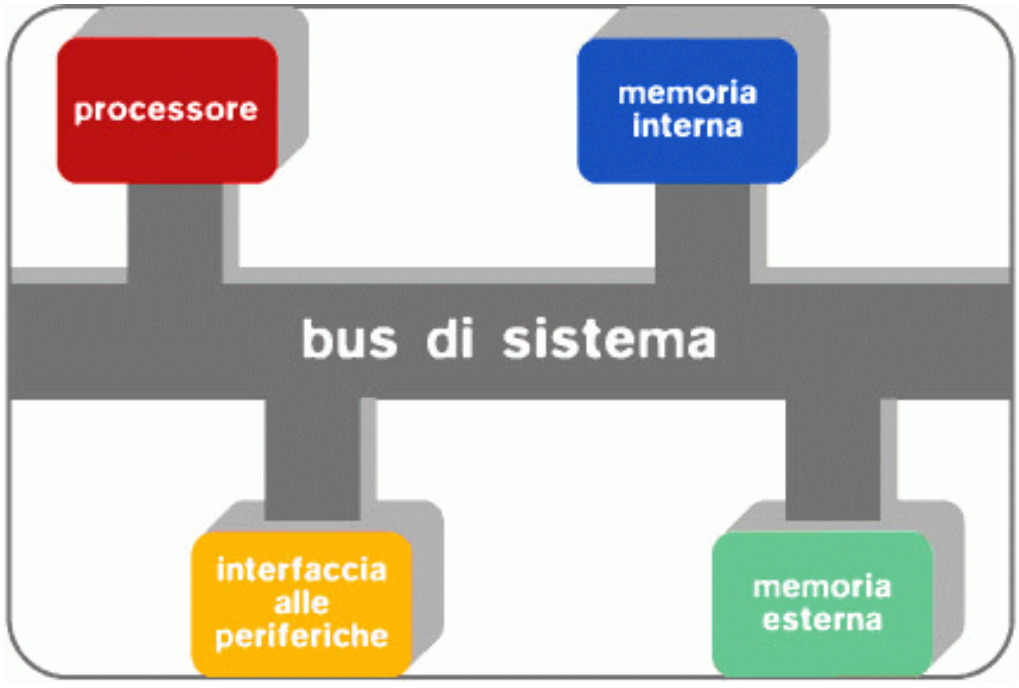


- **Architetture e sistemi operativi**
  - Com'è strutturato un computer?
  - Dispositivi principali
  - Evoluzione delle architetture
- **Il sistema operativo**
  - Concetti di base
  - Funzioni del sistema operativo
  - Sicurezza nei sistemi operativi
  - Strumenti di amministrazione (⌨ XP)

Parte I

Architetture degli elaboratori

La macchina di Von Neumann

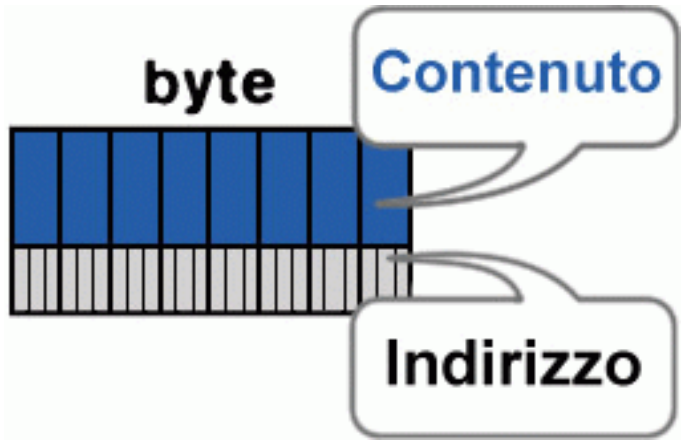


- 5 **moduli funzionali**
- Nel computer: motherboard, circuiti integrati, porte (interfacce), collegamenti anche wireless.

Funzionamento della CPU

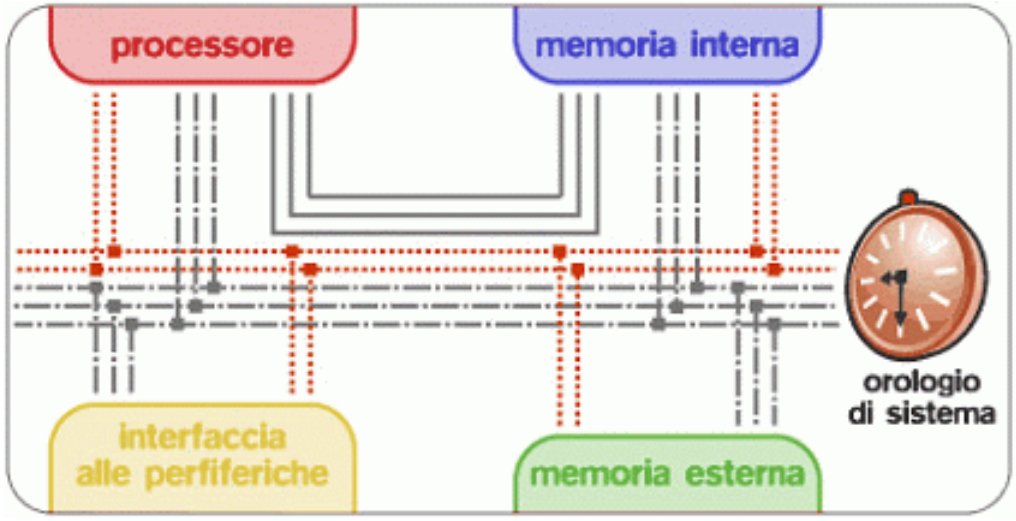
- Modello sequenziale.
- Sequenza dei passi scandita dal **clock** di sistema.

Funzionamento della RAM



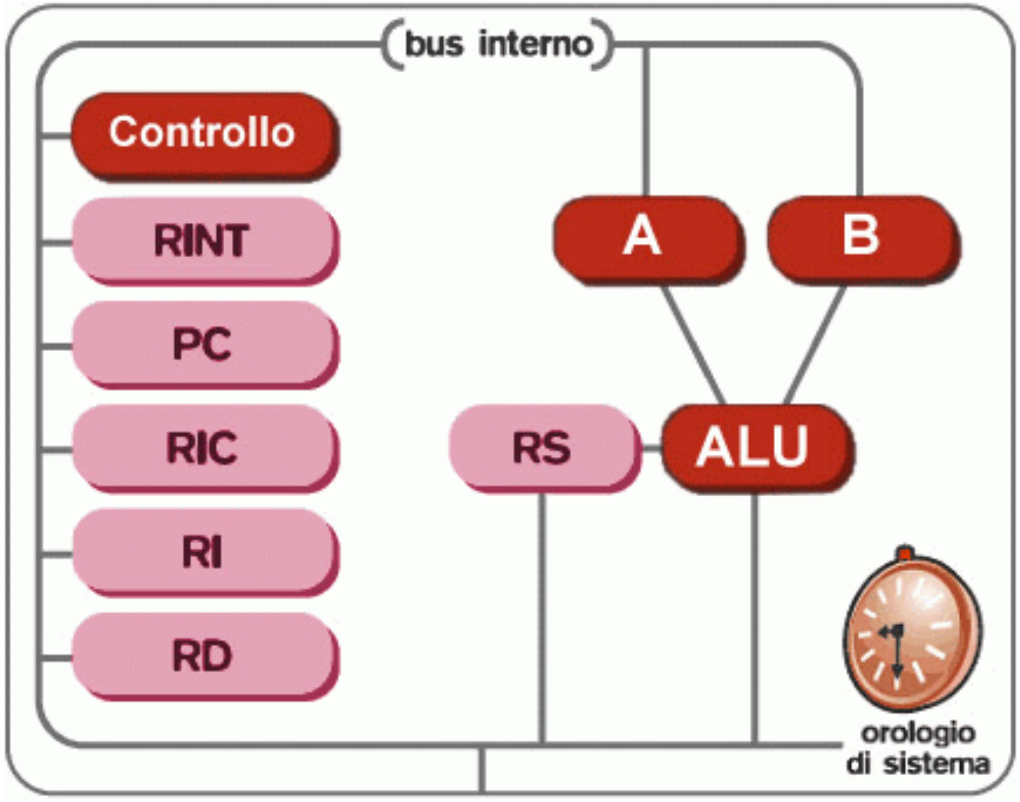
- Suddivisione in **parole** (es: 4 byte, 8 byte)
- identificate univocamente da un **indirizzo** (numero intero)
- Larghezza della parola (numero di byte) dipende dall'architettura del computer
- **Volatile**, e ad **accesso diretto** (Random Access Memory)
- Due operazioni possibili:
  - **LOAD** (lettura da RAM a CPU)
  - **STORE** (scrittura da CPU a RAM)

Funzionamento del BUS di sistema



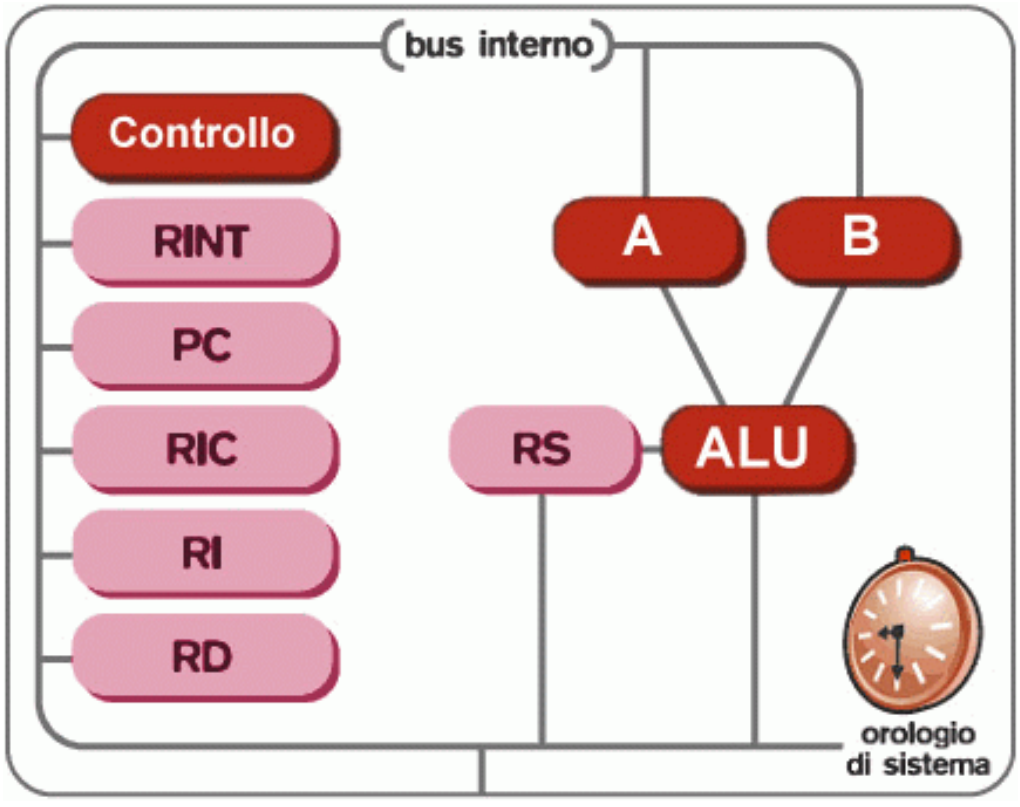
- Esistono **3 tipologie di connessione**
  - $\Leftrightarrow$ **bus dati**: fa transitare il **contenuto** del registro dati dalla CPU a una delle altre unità e viceversa;
  - $\Rightarrow$ **bus indirizzi**: fa transitare dalla CPU verso la memoria l'**indirizzo della parola** a cui si accede;
  - $\Leftrightarrow$ **bus controllo**: fa transitare
    - dalla CPU il codice della operazione da eseguire,
    - dalla unità funzionale il codice risultato dell'operazione
- Larghezza del bus dipende dall'architettura

La CPU



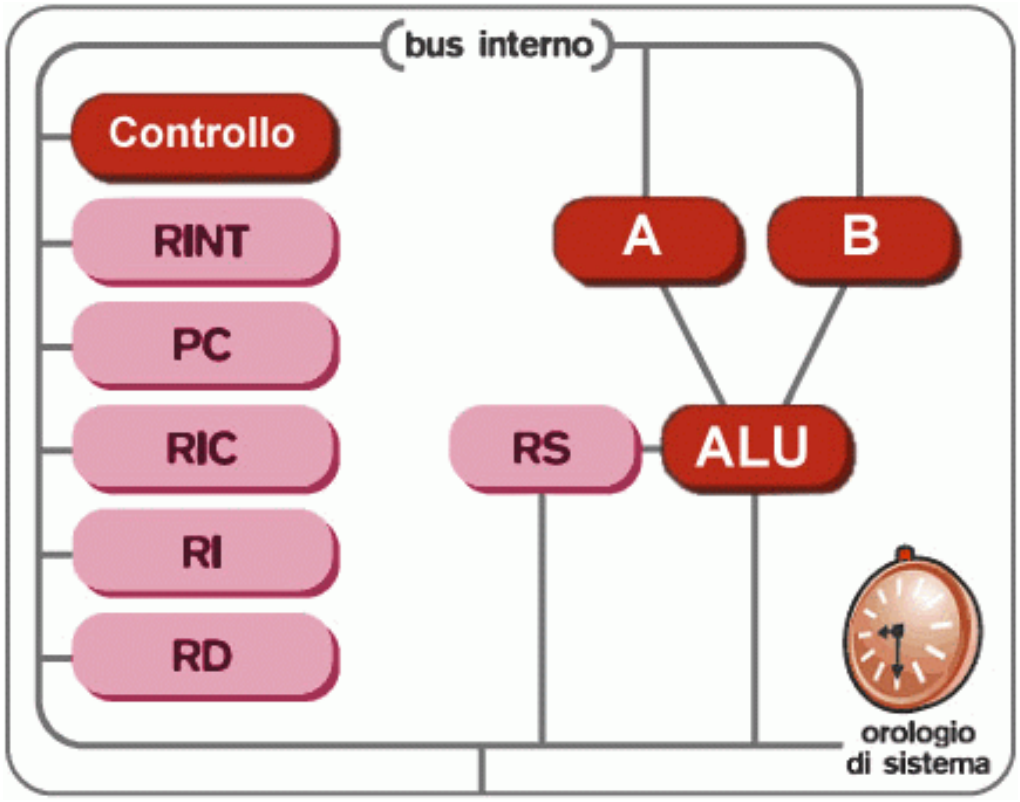
- **Control Unit**:
  - regola l'esecuzione del ciclo della CPU,
  - invia i segnali di controllo per i trasferimenti sul bus
  - decodifica le istruzioni
  - gestisce sia i trasferimenti all'interno della CPU sia quelli con il mondo esterno.

La CPU



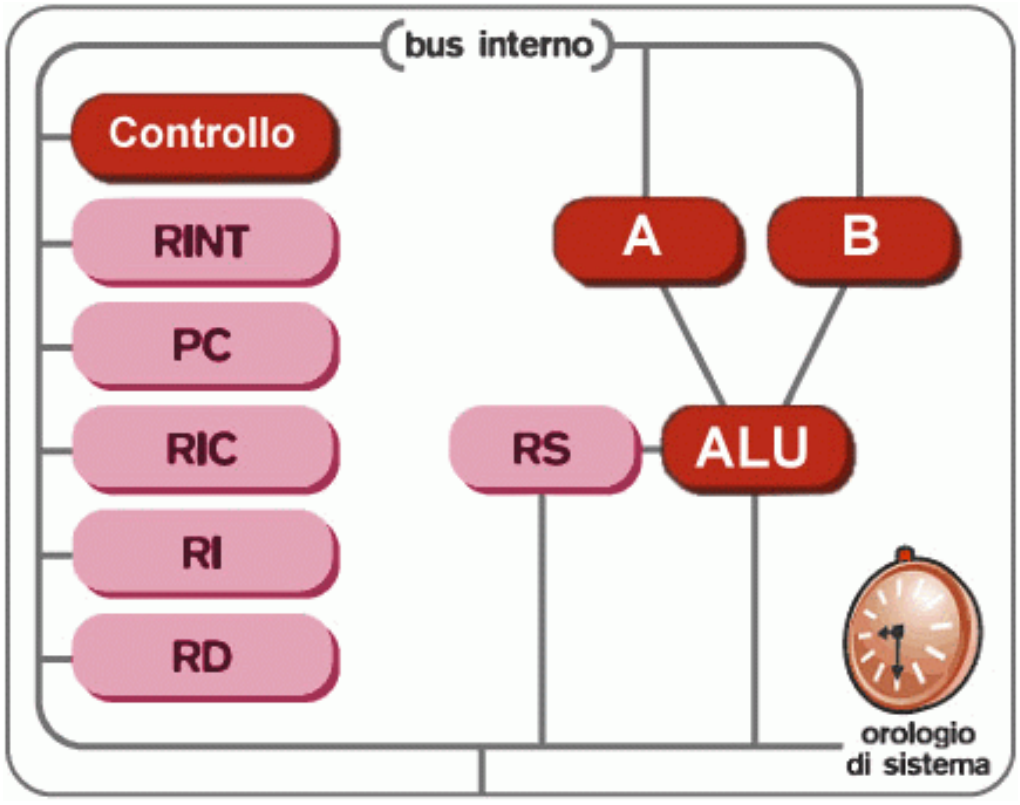
- **Clock**
  - determina la frequenza delle operazioni da eseguire

La CPU



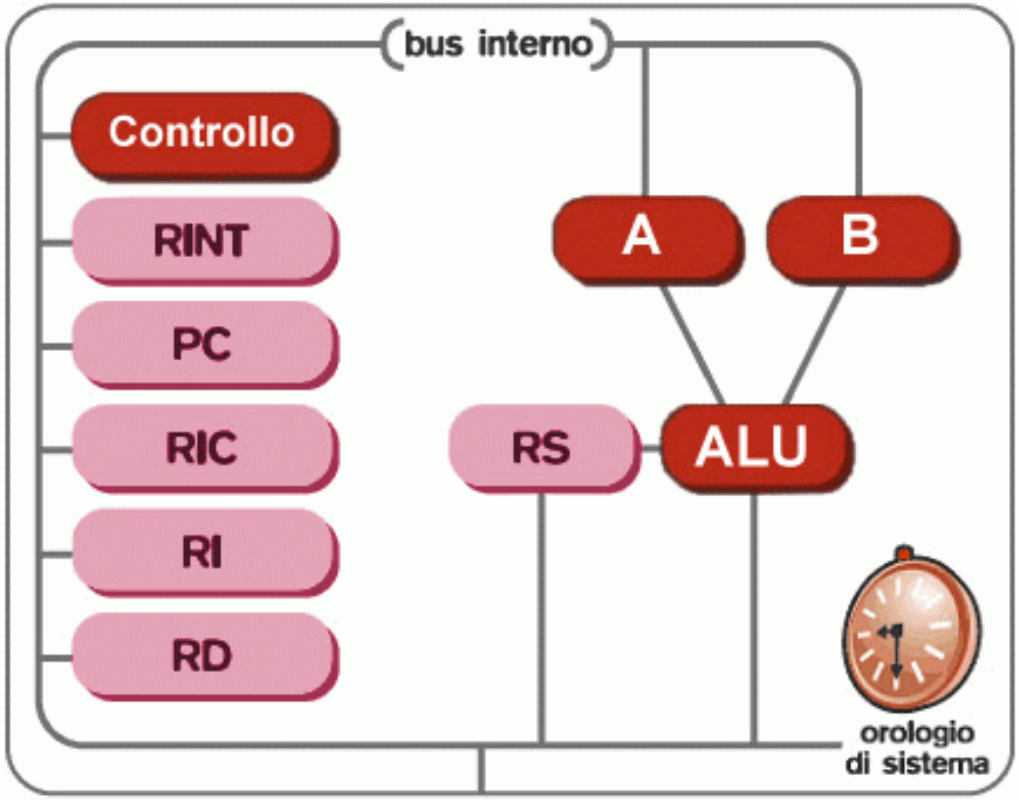
- **Arithmetic Logic Unit**
  - realizza le operazioni aritmetiche e quelle logiche
  - due operandi: A e B
  - risultato memorizzato in A

La CPU



- **Registri interni**
  - memorie locali che possono essere accedute alla velocità del clock della CPU
  - molto più veloci della RAM (clock del bus $<<$ clock CPU)

Registri della CPU



- **Dati (RD)**
- **Indirizzi (RI)**
- **Istruzione corrente (RIC)**
- **Contatore di programma (PC)**
- **Interruzioni (RINT)** (stato delle periferiche)
- **Operandi** (es: A e B)
- **Stato (RS)** (a volte: **flag**)

Ricordate CR?

Tabella: Possibile codifica delle istruzioni di CR

INC	001	**somma** 1 alla cella C_n
RESET	000	**memorizza** 0 nella cella C_n
STORE	010	**leggi** dall'esterno un numero, memorizzalo in C_n
WRITE	011	**mostra** il contenuto della cella C_n
HALT	111	**stop**
JMPEQ	100	**se** $C_n = C_m$ vai all'istruzione (X)

Codifica degli operandi per CR

- Gli operandi si riferiscono a celle di memoria
- Esempio: memoria a disposizione: 4GByte, suddivisa in parole di 64 bit
- Indirizzi distinti: $4GB/8B = 2^{32}/2^3 = 2^{32-3} = 2^{29}$

$\Rightarrow$ servono 29 bit per gli indirizzi

- Bastano invece 3 bit per codificare 6 istruzioni distinte

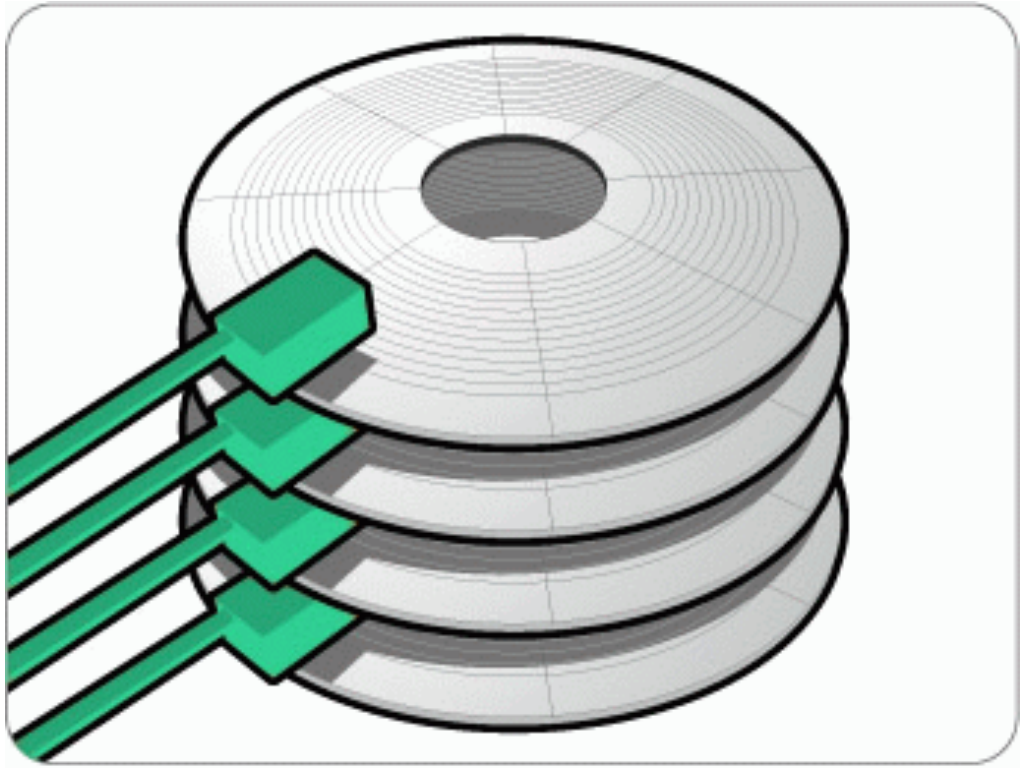
`INC` *cell_num*	`001 xx...x`
`RESET` *cell_num*	`000 xx...x`
`STORE` *cell_num*	`010 xx...x`
`WRITE` *cell_num*	`011 xx...x`
`HALT`	`111`
`JMPEQ` *c_1 c_2 (i)*	`100 xx...x xx...x xx...x`

- Lunghezza delle istruzioni:
  - 32 bit per `INC`, `RESET`, `STORE`, `WRITE`;
  - 3 bit per `HALT`,
  - 90 bit per `JMPEQ`

Parte II

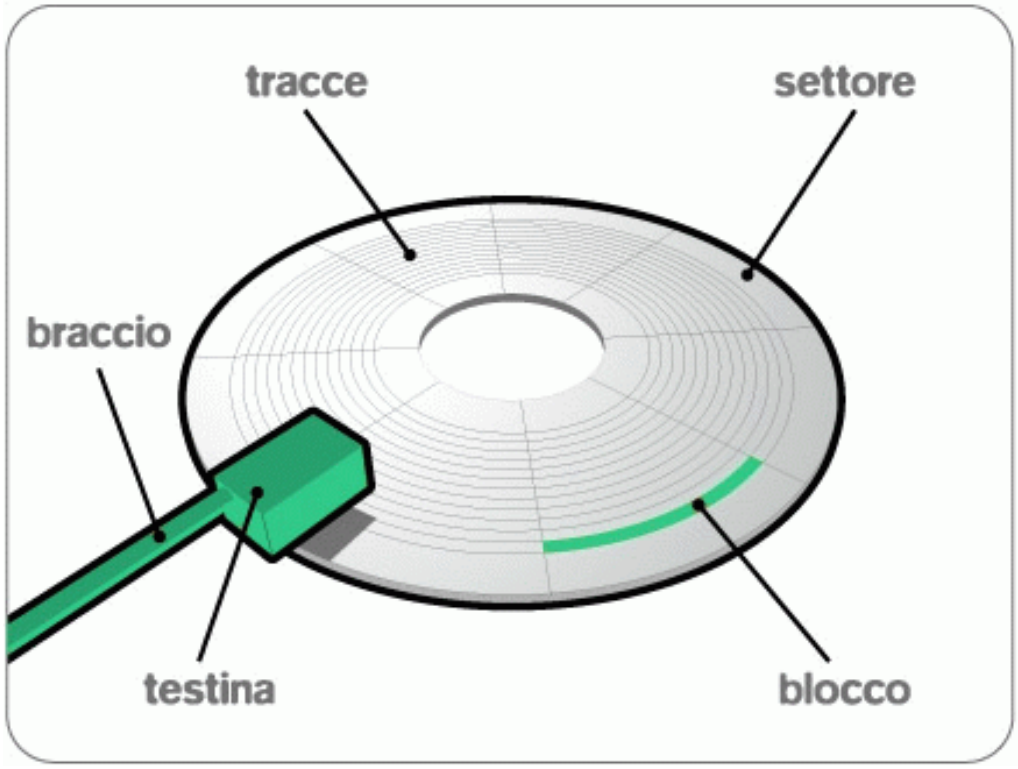
Dispositivi di memoria e di I/O

Memoria di massa



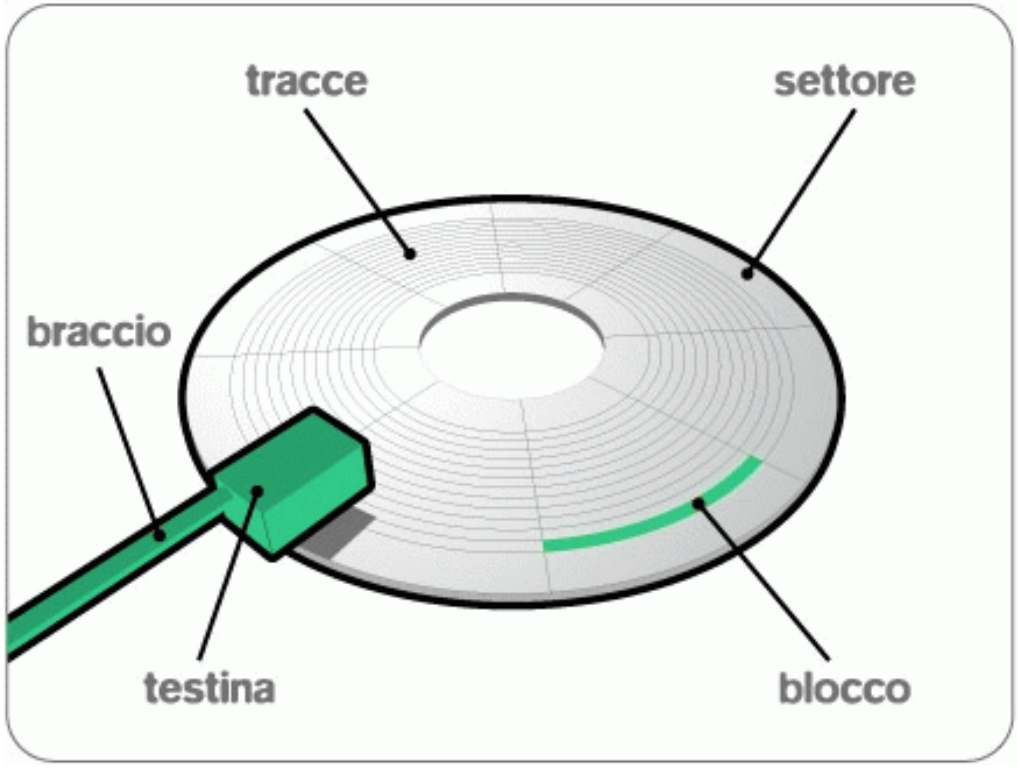
- **Hard disk**: dispositivo **magnetico**
  - Pila di dischi, ciascuno organizzato in **tracce** e **settori**

Memoria di massa



- **Hard disk**: dispositivo **magnetico**
  - Pila di dischi, ciascuno organizzato in **tracce** e **settori**

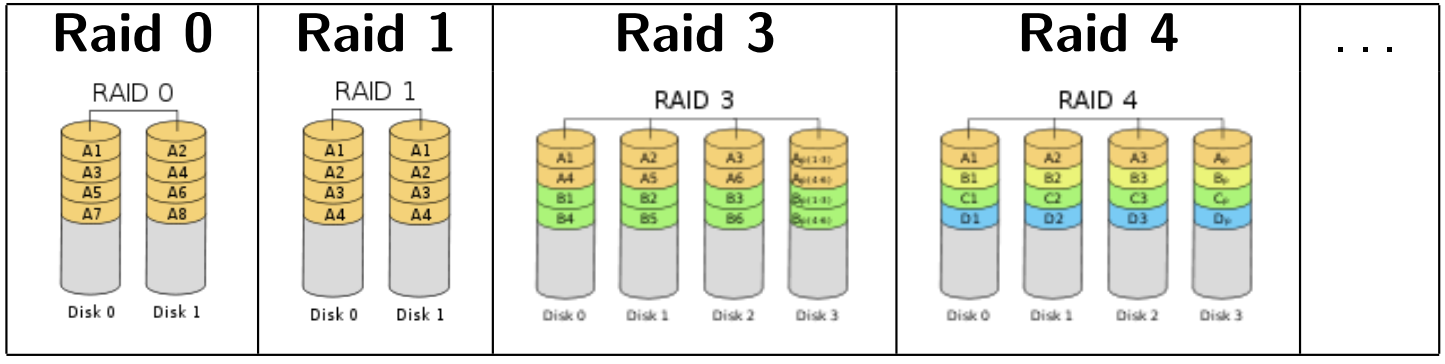
Memoria di massa



- **Hard disk**: dispositivo **magnetico**
  - Pila di dischi, ciascuno organizzato in **tracce** e **settori**
  - ciascun file occupa un numero intero di **blocchi**
  - **allocazione** dei file: tramite FAT (**File Allocation Table**)
  - operazioni di lettura/scrittura gestite da un **controller**

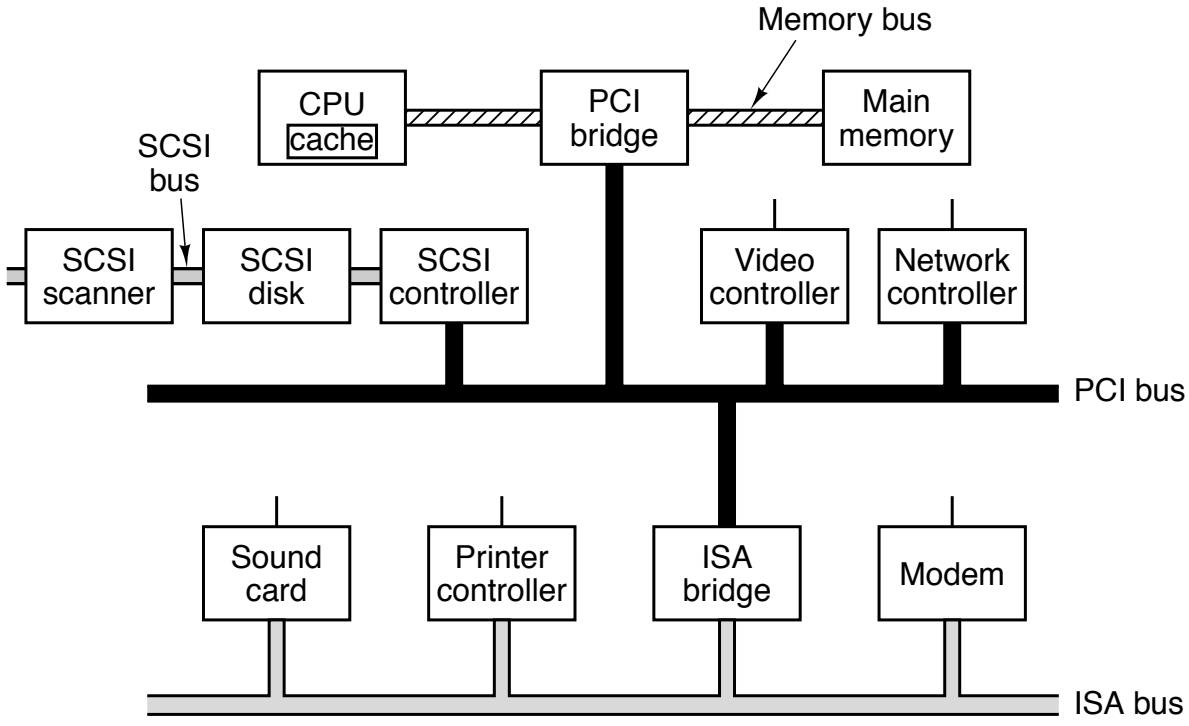
Altri dispositivi di memoria

- **RAID**
  - Gruppi di dischi rigidi che funzionano in parallelo
  - Varie modalità, per aumentare prestazioni e/o robustezza
    - Raid 1 (*striping*): migliori prestazioni
    - Raid 2 (*mirroring*): migliore robustezza, metà spazio
    - Raid 3, 4: almeno 3 dischi; 1 usato per garantire robustezza; migliori prestazioni
    - ...



- Dispositivi **ottici**: **CD, DVD, Blue Ray**
  - utili come sistemi di archiviazione
- Dispositivi **a stato solido**: **USB stick, Flash cards**
  - utili per trasporto e assenza di testine di lettura

Canali di comunicazione

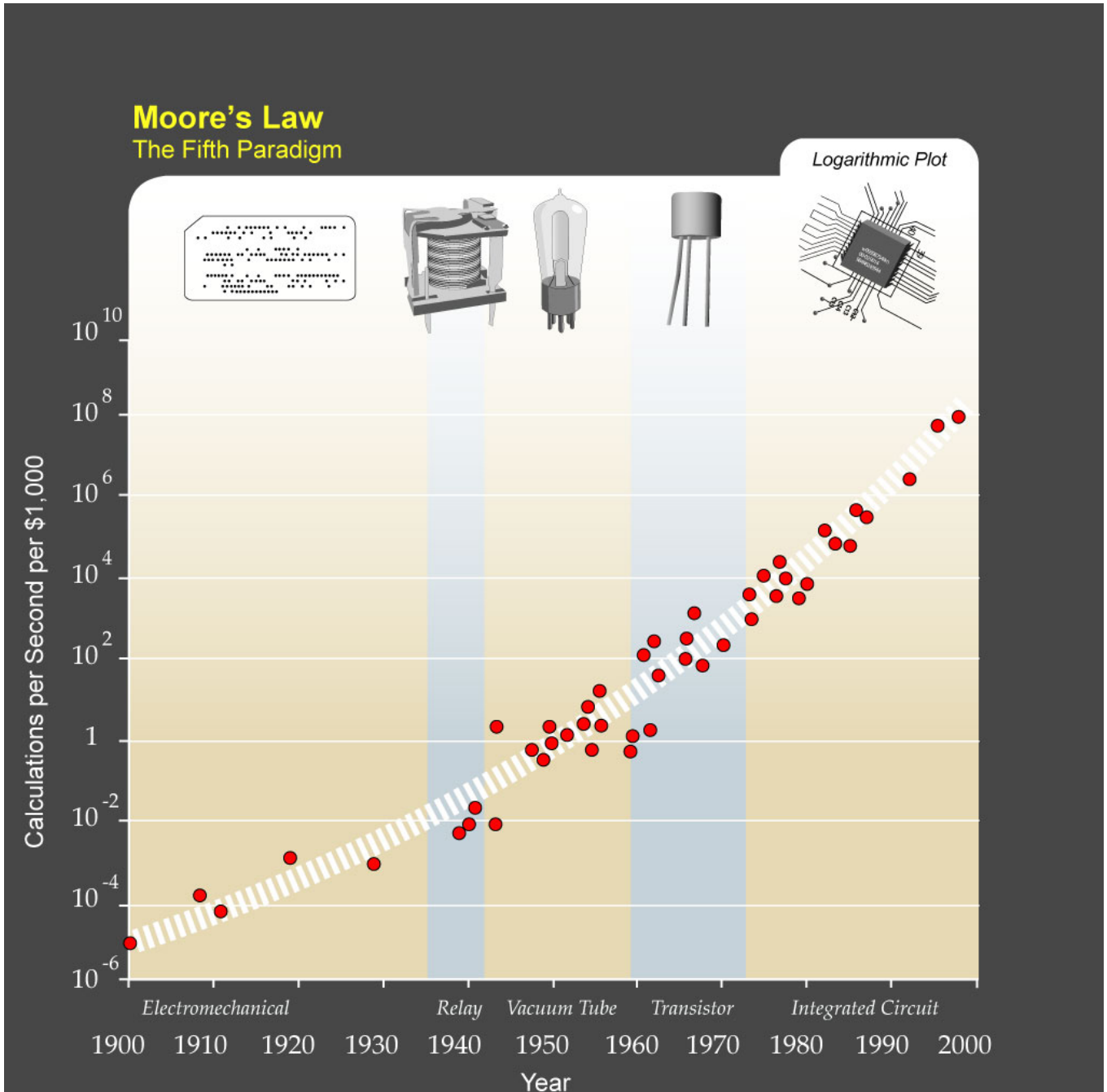


- **BUS** interni al sistema, di vario tipo: **Memory bus**, **PCI**, **ISA**
- Canali di comunicazione esterna, tramite interfacce:
  - **Porte seriali e parallele**
  - **USB Firewire** (IEEE 1394)
  - **Wi-FI** e **Bluetooth**

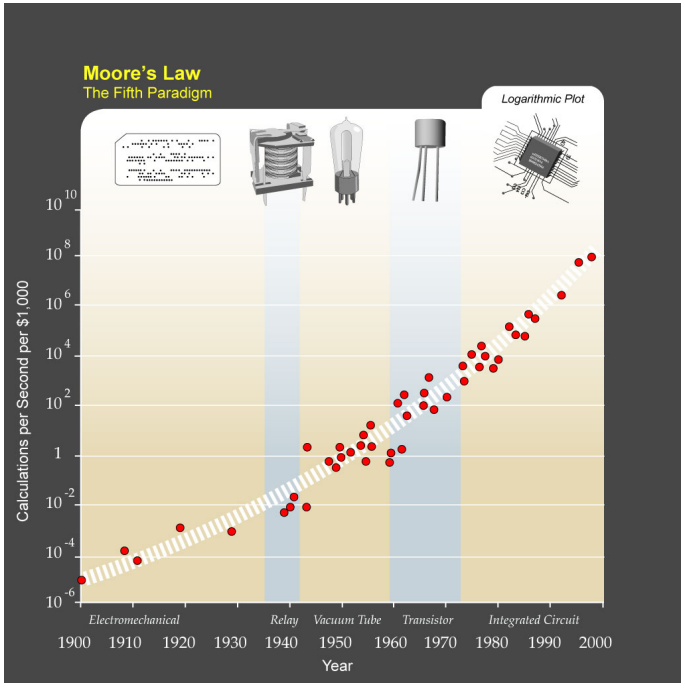
Parte III

Evoluzione delle architetture

La Legge di Moore (Gordon Moore, 1965, Intel)



La Legge di Moore (Gordon Moore, 1965, Intel)

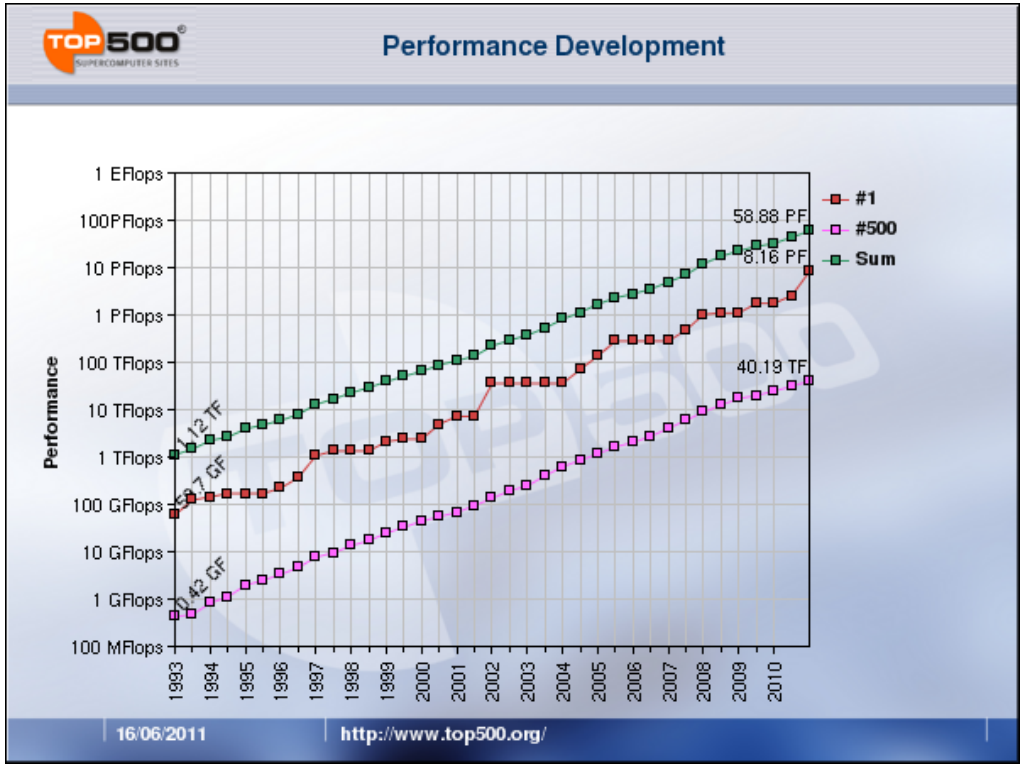


- Crescita esponenziale dovuta a miglioramenti tecnologici
- Limiti fisici
- Per continuare a crescere: necessarie modifiche all'architettura (sequenziale $\longrightarrow$ parallela)

Migliorie alla macchina di Von Neumann

- Co-processori
- **Pipeline** della CPU
- Gerarchie di memorie (**cache**)
- Processori di canali (Direct Memory Access, **DMA**)
- Architetture **multi processore**

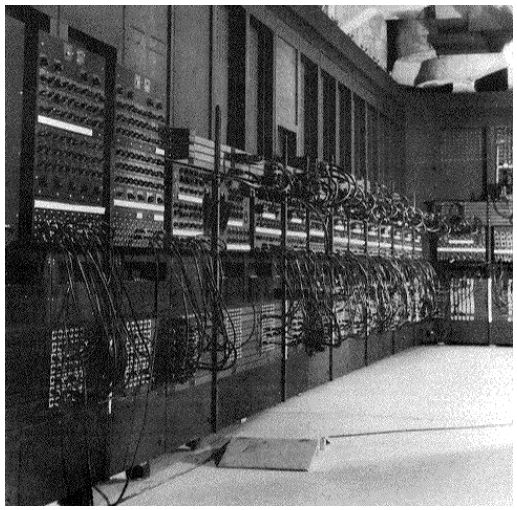
Supercomputer

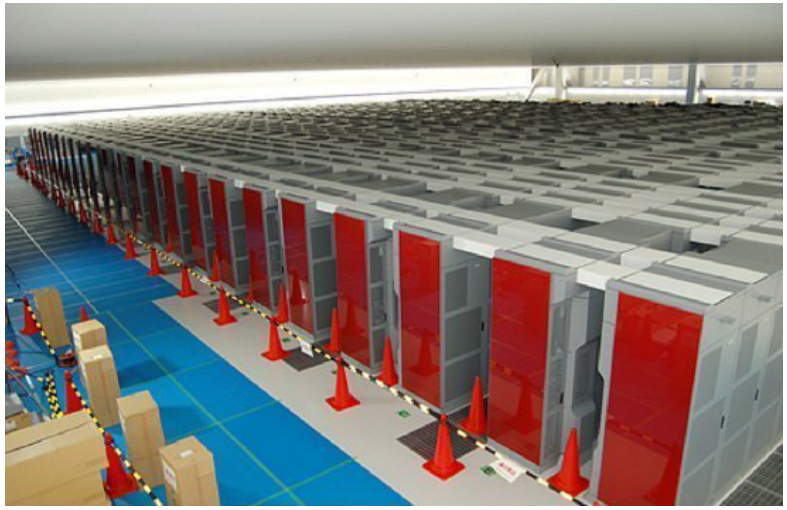


- Enti governativi, grossi progetti scientifici
- K computer (Fujitsu), **548.352 cores**, $>$ 8 Petaflop
  - Tianhe-1A (Cina), Jaguar (Cray, US), Nebulae (Cina), TSUBAME 2.0 (HP), . . . ,
  - Solo 10 computer $>$ 1 Petaflop (10^{15} Flops)

(fonte: `http://www.top500.org`)

Supercomputer



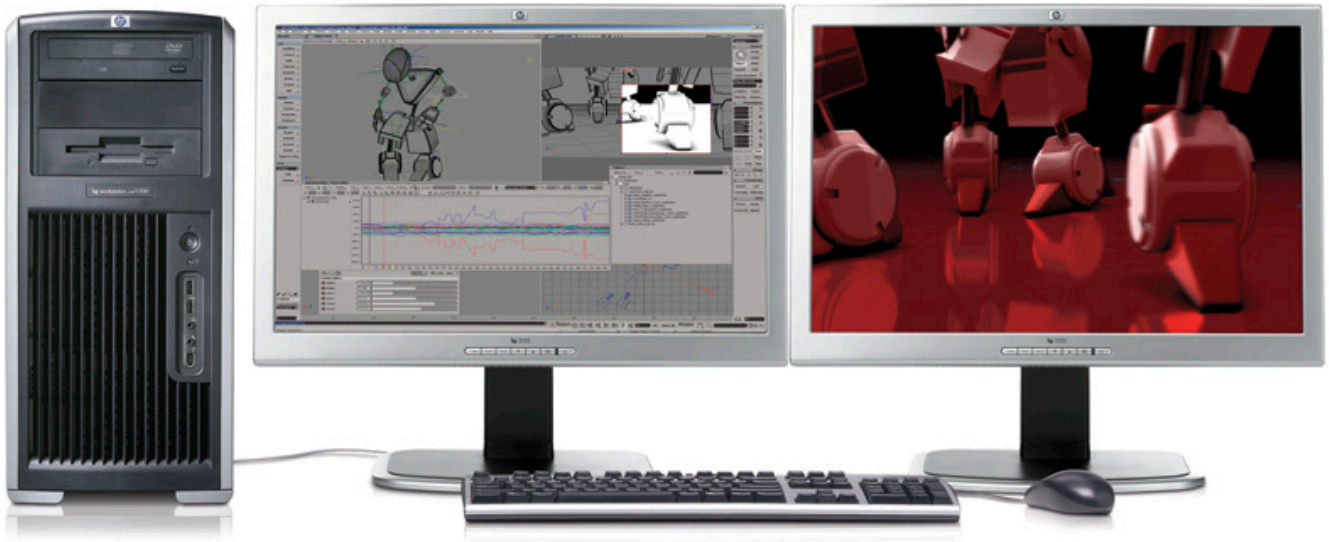


- ENIAC (1947)
  - 500 flops
  - 63 m^2 di spazio
  - 27 tonnellate

Mainframe

- Grandi imprese, milioni di euro
- IBM System z10
  `http://www.ibm.com/systems/z/`
  - B2B, cloud computing, data warehousing, virtualization, enterprise content management, transaction management, customer relationship management, ...
  - scalability, availability, security

Server e Workstation

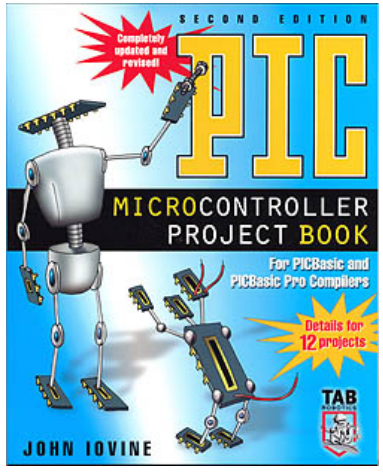


- PMI, migliaia/decine di migliaia di euro
- **Server**: per mettere risorse a disposizione di molti utenti contemporaneamente
  - servizi relativamente semplici
  - mainframe su piccola scala
- **Workstation**: orientata al singolo utente, per compiti che richiedono molte risorse
  - ambito grafico, progettazione CAD/CAM

Personal Computer

- Uso personale sia domestico che lavorativo
- Costo: centinaia/migliaia di euro
- Decine di miliardi di personal computer prodotti nel mondo

Microcontrollori

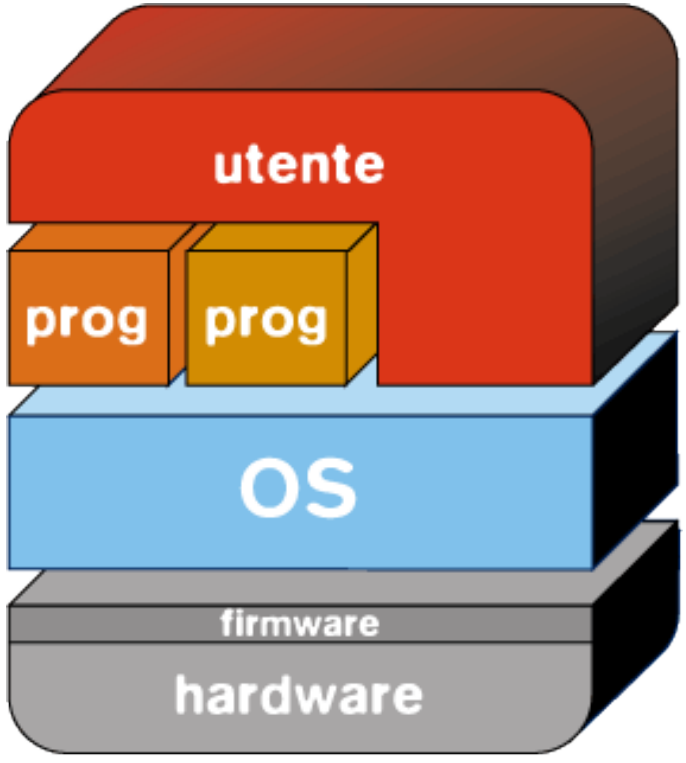


- Elettrodomestici, automobili, robotica, etc.
- Ovunque
- Euro/decine di euro

Parte IV

Il sistema operativo

Schema a livelli gerarchici



- Hardware + sistema operativo: **macchina astratta**

Cos'è un sistema operativo?

Definizione (Sistema Operativo)

Un sistema operativo è un programma che controlla l'esecuzione di programmi applicativi e agisce come interfaccia tra le applicazioni e l'hardware del calcolatore

- Obiettivi di un sistema operativo:
  1. **Semplicità**: semplificare l'utilizzazione della macchina da parte degli utenti
     - esecuzione di programmi
     - gestione dello storage (hard disk, CD, USB stick, ...)
     - interfaccia grafica con il sistema

Cos'è un sistema operativo?

Definizione (Sistema Operativo)

Un sistema operativo è un programma che controlla l'esecuzione di programmi applicativi e agisce come interfaccia tra le applicazioni e l'hardware del calcolatore

- Obiettivi di un sistema operativo:
  1. **Semplicità**: semplificare l'utilizzazione della macchina da parte degli utenti
  2. **Astrazione**: fornire una visione astratta delle risorse del calcolatore per i programmatori
     - Ponte tra visione astratta e visione fisica.
     - Esempio: *file system*.

Cos'è un sistema operativo?

Definizione (Sistema Operativo)

Un sistema operativo è un programma che controlla l'esecuzione di programmi applicativi e agisce come interfaccia tra le applicazioni e l'hardware del calcolatore

- Obiettivi di un sistema operativo:
  1. **Semplicità**: semplificare l'utilizzazione della macchina da parte degli utenti
  2. **Astrazione**: fornire una visione astratta delle risorse del calcolatore per i programmatori
  3. **Efficienza**: utilizzare in modo efficiente le risorse del calcolatore
     - arbitrare le diverse attività che vengono svolte in un calcolatore
     - promuovendo il parallelismo ed limitando i tempi morti

Cos'è un sistema operativo?

Definizione (Sistema Operativo)

Un sistema operativo è un programma che controlla l'esecuzione di programmi applicativi e agisce come interfaccia tra le applicazioni e l'hardware del calcolatore

- Obiettivi di un sistema operativo:
  1. **Semplicità**: semplificare l'utilizzazione della macchina da parte degli utenti
  2. **Astrazione**: fornire una visione astratta delle risorse del calcolatore per i programmatori
  3. **Efficienza**: utilizzare in modo efficiente le risorse del calcolatore
  4. **Protezione**: proteggere le risorse del computer e permettere un accesso controllato ad esse
     - supporto alla multi-utenza

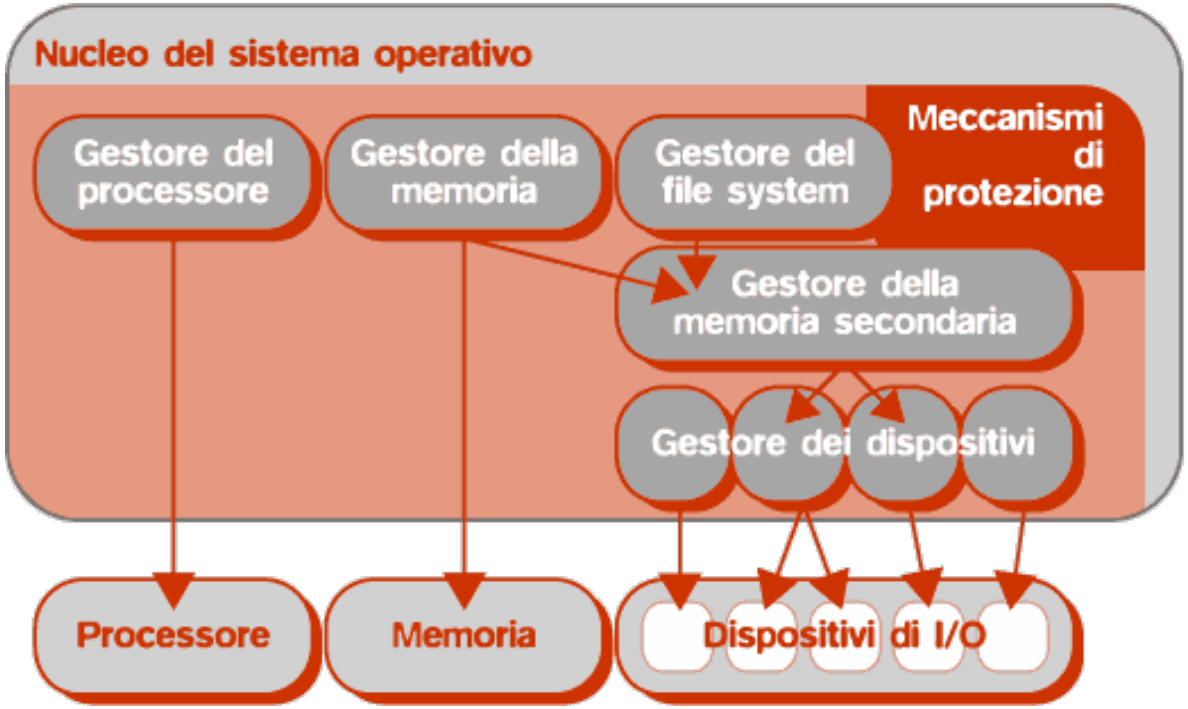
Servizi offerti dai sistemi operativi

1. Esecuzione di programmi
   - applicativi e di sistema
   - supporto alla **multi-programmazione**
2. Gestione dell'accesso alla **memoria** del calcolatore
   - principale e secondaria
3. Gestione dell'accesso ai **dispositivi di I/O**
4. Rilevazione e risposta agli **errori**
   - isolare gli effetti degli errori (ambiente protetto di esecuzione)
5. Amministrazione di **utenti** diversi (multi-utenza)
   - occorre separare le attività e i dati di ciascun utente
6. Controllo degli **accessi**
   - l'utente che lancia un programma ha il diritto di eseguirlo?
7. Accounting
   - misura e controllo della **quantità** di risorse usate (es: quota)

Elementi di un sistema di calcolo

- **Hardware**: il più difficile e complesso da programmare
- **Kernel**: gestore delle risorse del sistema
- **System Call**: interfaccia di programmazione su cui costruire **librerie** e **tool di sistema** (es: eseguire un programma, copiare un file)

Unità operative di un sistema operativo



- Supporto alla **multi-programmazione**
- Gestore della **memoria** principale, eventualmente usa anche il gestore della memoria secondaria (swap)
- Gestore di file system, memoria secondaria, e dispositivi di I/O: **astrazione di file e directory**
- Meccanismi di **protezione**: integrati nei vari moduli

Sistemi operativi speciali

- **Real-time**
  - hard real-time: controllo industriale
  - soft real-time: entertainment.
- **Alto parallelismo**
  - soluzioni specifiche per sfruttare al massimo le potenzialità dell'architettura
  - problemi: comunicazione/coordinamento dei processori, uso delle risorse condivise
- **Sistemi embedded**
  - aspetto cruciale: limitazione delle risorse
  - decoder per satellite, cellulari, sistemi di controllo per elettrodomestici, smart card

Parte V

Gestione dei processi

Processi

Definizione (Processo)

Un **processo** *è un'attività controllata da un* **programma** *che si svolge su un* **processore**.

- È il modo in cui un programma viene eseguito **nel tempo**.
  - Programma: entità statica.
  - Processo: entità dinamica.
- Ciascun processo ha uno **stato**, composto da un certo numero di informazioni
  - Immagine di memoria
  - Immagine nel processore
  - Stato di avanzamento

Stato di un processo

Definizione (Immagine di memoria)

*L'**immagine di memoria** di un processo è l'insieme di informazioni relative al processo mantenute nella memoria principale del calcolatore.*

- **Codice** del programma in esecuzione
- **Dati** su cui sta lavorando
- **Strutture dati del SO** per la gestione del processo

Stato di un processo

Definizione (Immagine nel processore)

*L'**immagine nel processore** di un processo è il contenuto dei registri del processore, che vengono utilizzati come memoria temporanea durante l'esecuzione delle istruzioni.*

- Esempi:
  - **Program counter**: la prossima istruzione da eseguire
  - **Flag**: esito di una operazione della ALU
  - **A**, **B**, ...: operandi

Stato di un processo

Definizione (Stato di avanzamento)

Lo **stato di avanzamento** *descrive l'attuale attività del processo.*

- Esempi di possibili stati di avanzamento
  - **Waiting**: in attesa di qualche evento
  - **Running**: in esecuzione
  - **Ready**: attende di essere eseguito

Multi-programmazione e time-sharing

Definizione (Multi-programmazione)

Tecnica di gestione della CPU secondo la quale durante i **periodi di I/O** *di un processo vengono eseguiti altri processi.*

Definizione (Time-sharing)

Tecnica di gestione della CPU secondo la quale l'esecuzione del processore viene suddivisa in un certo numero di **quanti temporali**; *allo scadere di un quanto, il processo corrente viene interrotto e l'esecuzione passa ad un altro processo.*

- Numerosi vantaggi:
  - Processore non inattivo durante operazioni di I/O (lunghe).
  - Memoria utilizzata al meglio, caricando il maggior numero di programmi possibile.
  - Impressione di esecuzione contemporanea di processi diversi.

Meccanismo degli interrupt

- Implementazione:
  - Tramite un componente del SO detto **scheduler**
  - Invocato ogni volta che una operazione di I/O viene:
    - **iniziata** (da un processo)
    - **terminata** (da un dispositivo)
  - Usa meccanismo degli **interrupt**

$\rightarrow$ **richiesta di I/O da parte di un processo**

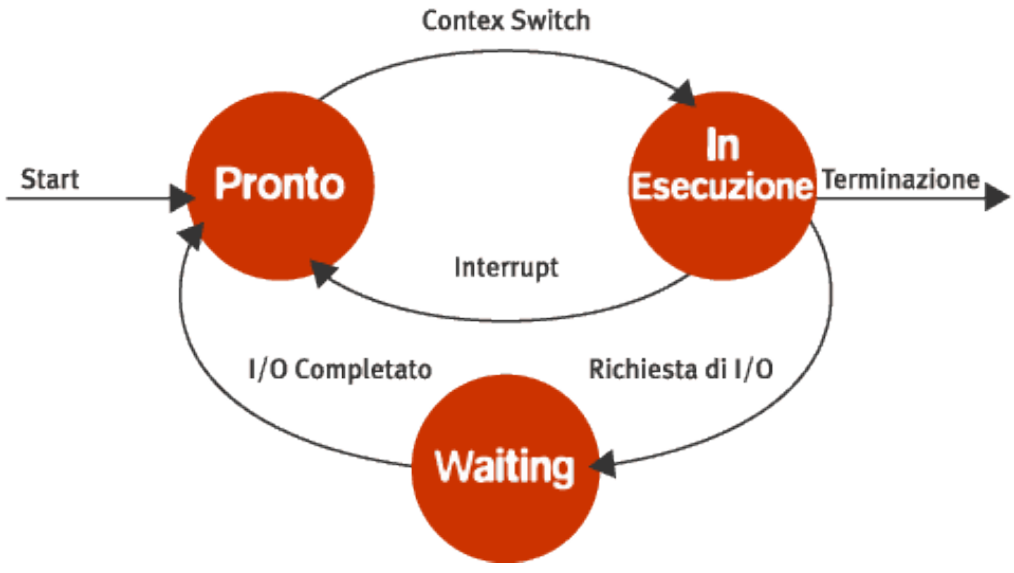
- interrupt software

$\rightarrow$ **completamento di tale operazione**

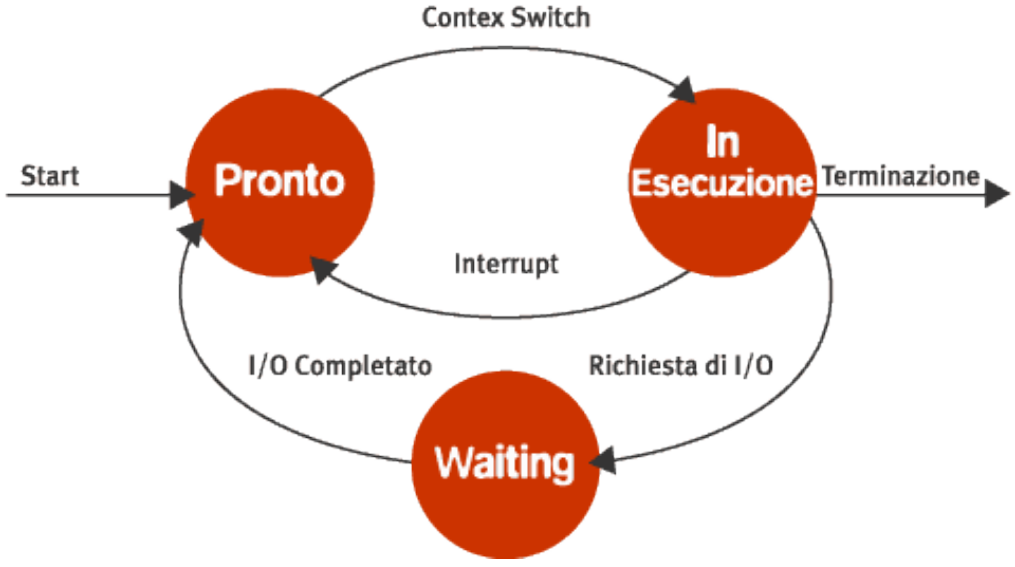
- interrupt hardware generato dal dispositivo associato

- suddivisione **in quanti temporali**
  - tramite un processo **timer**, che è in grado di generare un interrupt periodicamente
- Per passare da un processo a un altro: **context switch**.

Lo stato di avanzamento dei processi

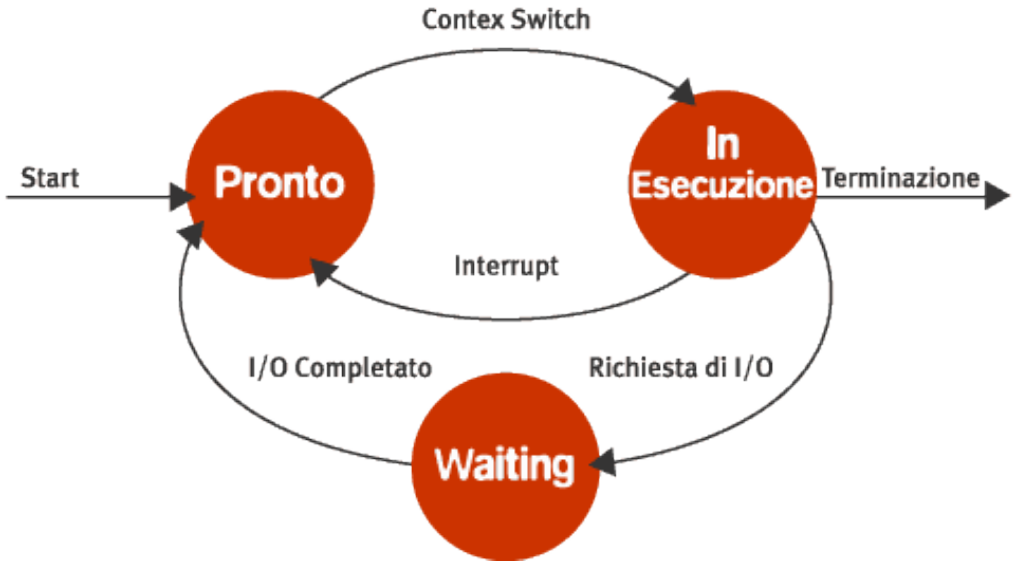


Lo stato di avanzamento dei processi



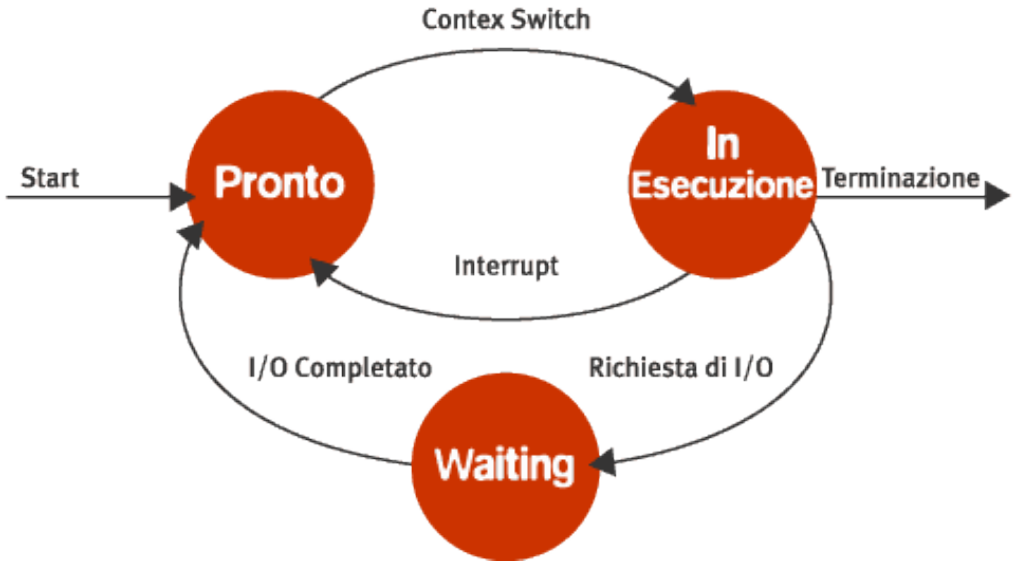
- **In esecuzione (running)**
  - il processo è in esecuzione
    - 1 CPU $\Rightarrow$ 1 solo processo in esecuzione per volta
    - multi-core: 1 processo per CPU
  - controllo passa al SO se:
    - interrupt hardware $\Rightarrow$ *ready*
    - interrupt software $\Rightarrow$ *waiting*

Lo stato di avanzamento dei processi



- **In esecuzione (running)**
- **In attesa (waiting)**
  - il processo è in attesa di evento esterno (tipicamente, I/O)
    - non può essere eseguito
  - quando l'evento si verifica $\Rightarrow$ *ready*

Lo stato di avanzamento dei processi

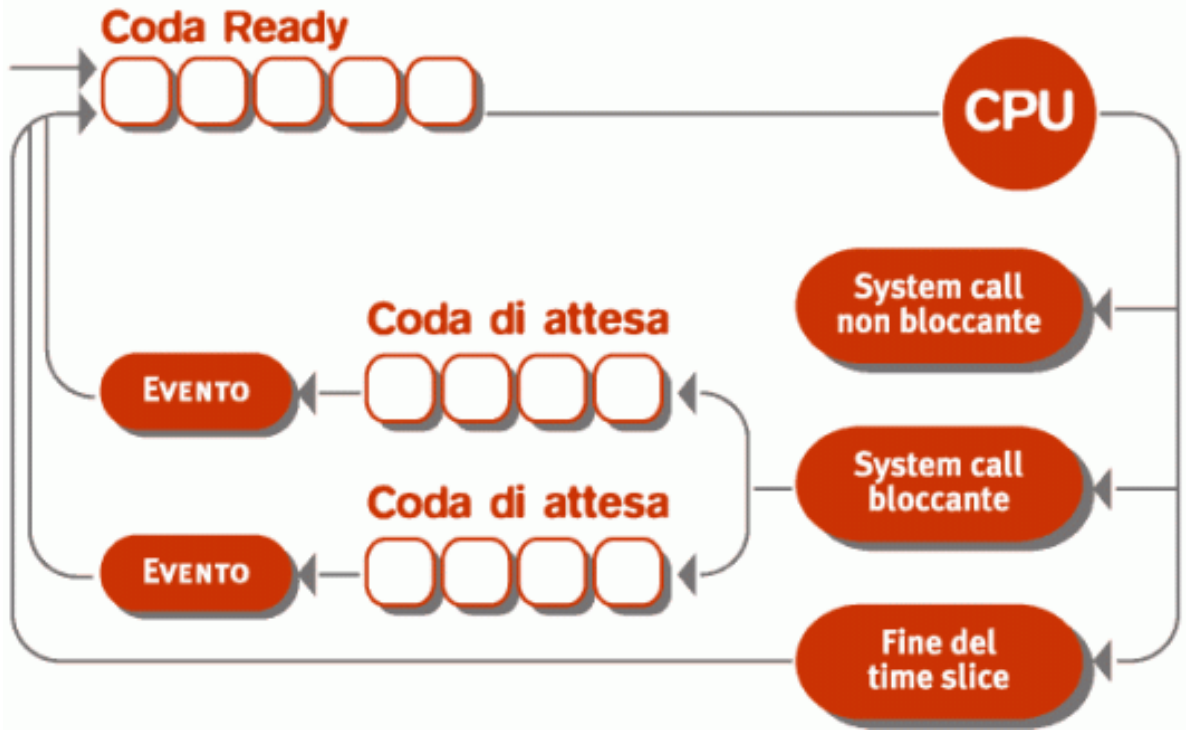


- **In esecuzione (running)**
- **In attesa (waiting)**
- **Pronto (ready)**
  - il processo può essere eseguito, ma
  - la CPU è impegnata da un altro processo
  - il processo attende che il SO lo faccia ripartire ($\Rightarrow$ *running*)

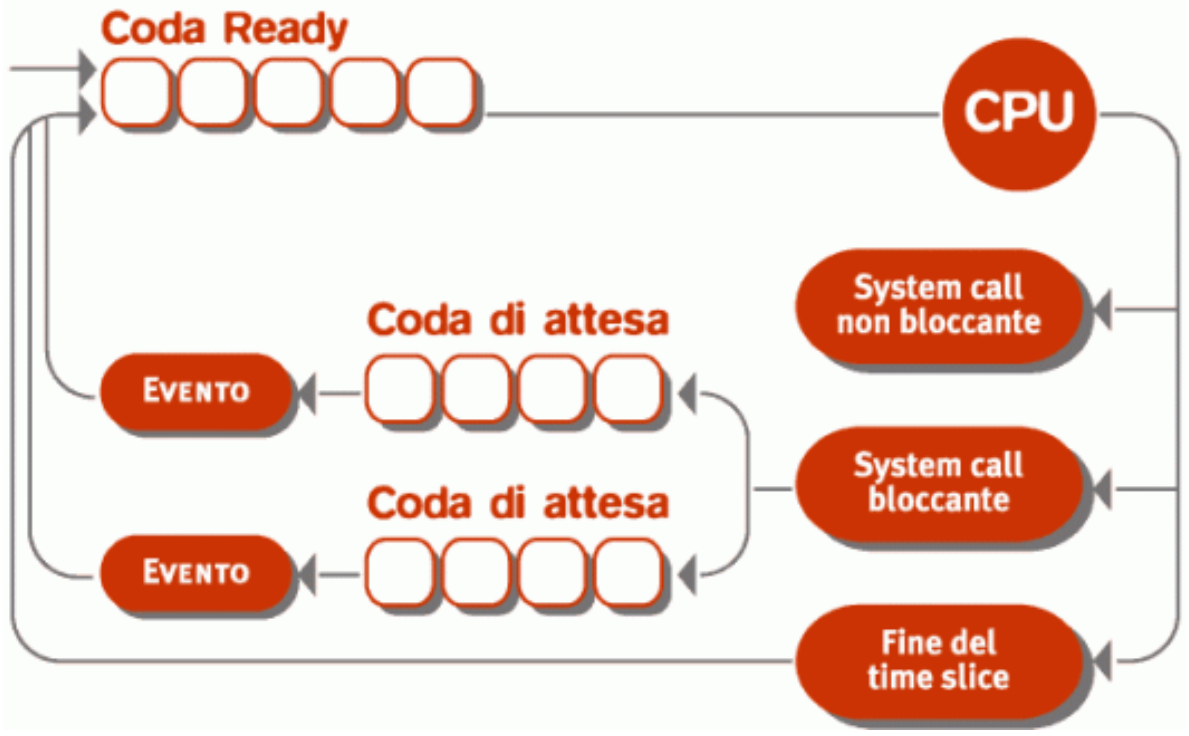
Lo scheduler

Definizione (Scheduler)

Lo **scheduler** *è il componente del sistema operativo che decide di volta in volta quale processo deve essere eseguito.*



Lo scheduler



- Usa **code di processi**, ordinati secondo **politiche**:
  - FCFS (First Come First Served)
  - Basate su priorità (importanza)
  - Per vincoli real-time (es. per riproduzione video)

Informazioni sui processi attivi (UNIX)

- Il SO tiene traccia di alcune informazioni sui processi, tra cui:
  - Direttorio corrente (**working directory**)
  - Info sui file usati dal processo (**file descriptor table**)
  - ID del processo (**Process ID, PID**)
  - ID del gruppo (**Group ID, GID**)
  - ID del processo "padre" (**Parent Process ID, PPID**)
  - Variabili di ambiente (es: `PATH`)

Come interagire con i processi

- Funzioni del SO per operare con i processi:
  - **Creazione di un processo**: `fork()`
  - **Esecuzione di un programma**: `exec()`
  - **Terminazione**: `exit()`
  - **Sincronizzazione tra processi**: `signal()` e `kill()`
  - **Comunicazione tra processi**: `send()` e `receive()`
- Operazioni comuni per un utente:
  - esecuzione di un programma (es. *double click*)
  - terminazione di un processo (es. *task manager*)

Parte VI

Gestione della memoria

Gestore della memoria

Definizione (Gestore della memoria)

Il **gestore della memoria** *è il componente del sistema operativo che si occupa di gestire la memoria per conto dei processi.*

- La memoria serve a ogni processo in esecuzione, per contenere **codice** e **dati**.
- La memoria (principale) è una **risorsa limitata** (e costosa):
  - Gerarchia delle memorie
    - Memoria principale (100 €): $\sim$ GB
    - Memoria secondaria (100 €): $\sim$ TB
- Uso di una porzione dell'Hard Disk (**partizione di swap**) per emulare la memoria principale (**memoria virtuale**)

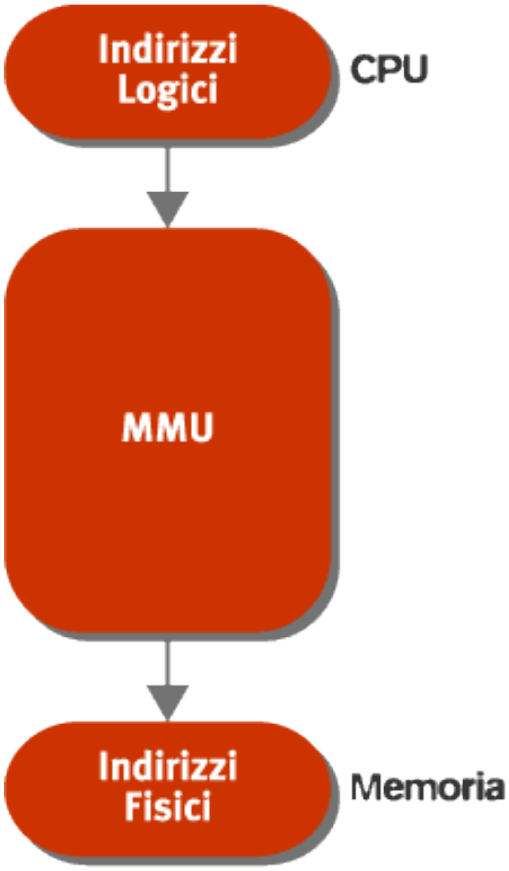
Gestore della memoria

Definizione (Gestore della memoria)

Il **gestore della memoria** *è il componente del sistema operativo che si occupa di gestire la memoria per conto dei processi.*

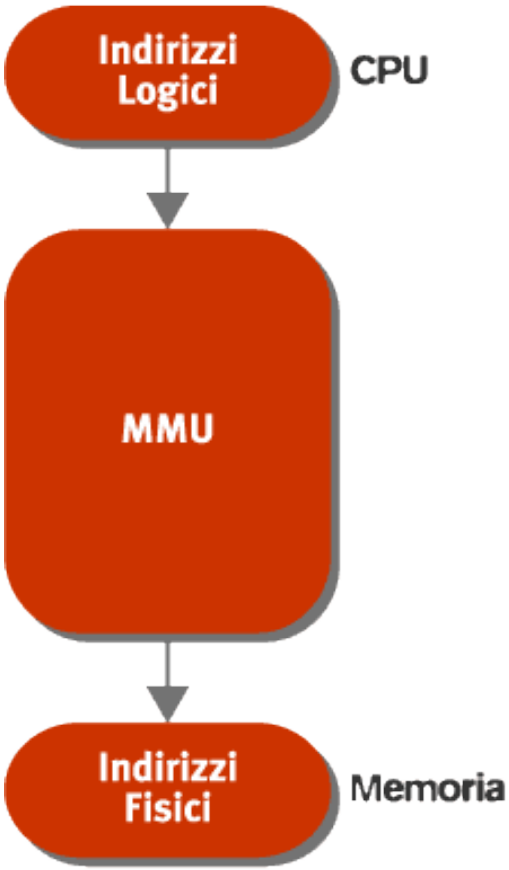
- Compiti del gestore della memoria:
  1. Tenere traccia di quali processi occupano quali porzioni di memoria
     - Uso di **tabelle di allocazione**
  2. Assegnare memoria ai processi che ne facciano richiesta
     - **Allocazione** di **pagine** di memoria
  3. Rilasciare la memoria quando non è più richiesta
     - **Deallocazione** di pagine di memoria

Memoria fisica e memoria logica



- **Spazio di indirizzamento logico**
  - Ogni processo è associato ad uno spazio di indirizzamento logico
  - Gli **indirizzi usati in un processo** sono indirizzi logici
- **Spazio di indirizzamento fisico**
  - Ad ogni indirizzo logico corrisponde un indirizzo fisico della memoria
  - Serve una **funzione di traduzione** da indirizzi logici a indirizzi fisici
  - La Memory Management Unit (MMU) è un dispositivo che esegue tale funzione

Memoria fisica e memoria logica



- Vantaggi di questa traduzione:
  - È meglio **nascondere** ai processi la reale organizzazione della memoria
    - Per **semplificare** il codice
    - → ad es: il codice usa sempre gli indirizzi da 0 a 1000
    - → indipendentemente dal gestore
    - → anche se in RAM non esistono tutti gli indirizzi da 0 a 1000
    - Per **proteggere** la memoria
    - → evitare che un processo interagisca su zone su cui non ha i diritti
  - La MMU può **ottimizzare** l'utilizzo della memoria
    - Il singolo processo non può sapere

La memoria virtuale

- Per poter eseguire un programma: dati e codice in memoria.
- Osservazione: **non tutti i dati** e **non tutto il codice** vengono utilizzati **in ogni istante**.

$\Rightarrow$ Non è necessario che l'intero spazio di indirizzamento logico di un processo sia in memoria

- Alcune parti sono utilizzate più raramente di altre.

Definizione (Memoria virtuale)

La **memoria virtuale** *è una tecnica che permette l'esecuzione di processi che sono parzialmente mantenuti in memoria principale e parzialmente mantenuti in memoria secondaria.*

Paginazione su richiesta

- È un meccanismo per implementare la memoria virtuale.
- Memoria logica: divisa in **pagine** di dimensione fissa.
- Memoria principale (fisica): divisa in **frame**, stessa grandezza.
- Ogni pagina logica può essere memorizzata:
  1. in un **frame** in **memoria principale**
  2. in un **blocco** della **memoria secondaria** (hard disk)
- Ogni volta che si accede a una certa pagina, la MMU sa se si trova nel caso (1.) o (2.).
  - Caso (1.) $\Rightarrow$ OK.
  - Caso (2.) $\Rightarrow$ bisogna copiare il blocco della memoria secondaria in un frame libero della memoria principale:

    2.1 C'è un frame libero $\Rightarrow$ OK.

    2.2 Non c'è nessun frame a disposizione $\Rightarrow$ occorre fare spazio!

    $\Rightarrow$ si sceglie il frame usato meno spesso o meno di recente, e si copia nella memoria secondaria (**swap**)

Osservazioni sulla memoria virtuale

- Consente di eseguire un insieme di **processi che richiedono più memoria di quella disponibile**.
- Ottimo! Però . . .
  - il ricorso a memoria secondaria rallenta di molto le prestazioni.
    - accesso a **1 dato** in RAM: decine di nanosecondi ($\sim 10^{-8}s$)
    - copia di **1 pagina** da HD: decine di microsecondi ($\sim 10^{-5}s$)
    - 3 ordini di grandezza
  - Per evitare un decadimento generale delle prestazioni occorre
    - avere moltissime più richieste a pagine presenti in RAM che non su HD nella partizione di swap
    - adottare un buon meccanismo di scelta delle pagine da trasferire su HD (caso 2.2)
- Meccanismo simile a quello delle **cache**

Parte VII

Gestione dei dispositivi

Gestione dei dispositivi di I/O

- Il buon funzionamento di un SO dipende anche da un **uso efficiente delle sue risorse** di I/O e di memorizzazione
- Due entità gestiscono il dispositivo:
  - **Driver** (SW)
  - **Controller** (HW)

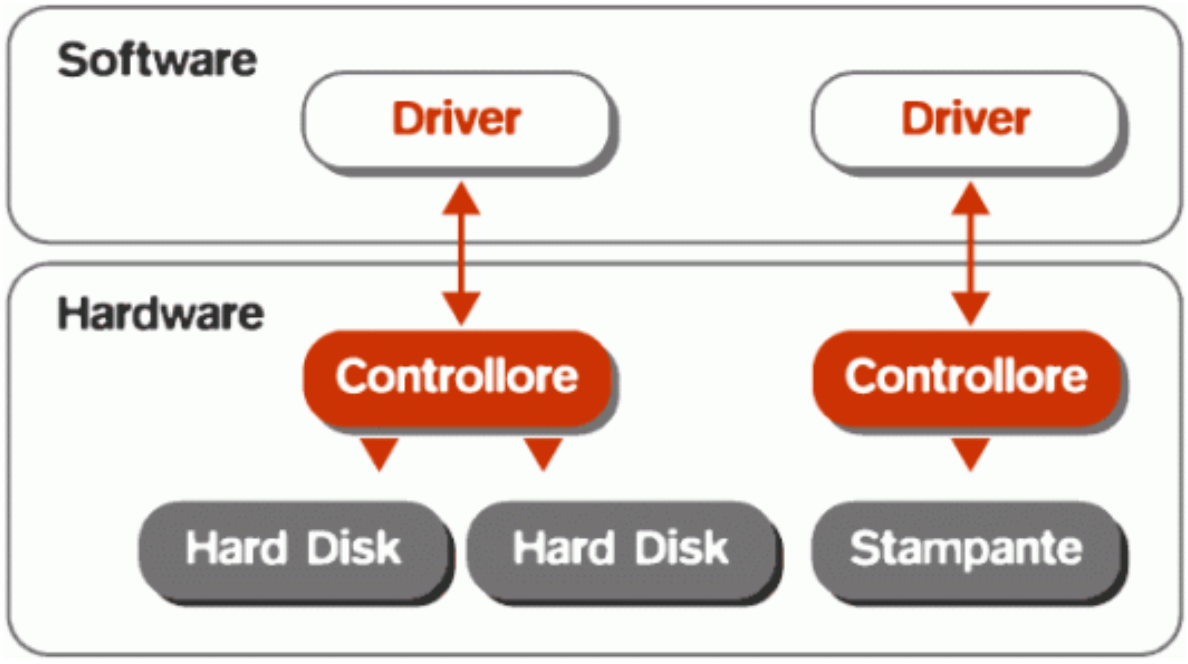
Definizione (Driver)

Il **driver** *è un componente del SO che si occupa di fornire ai processi un'interfaccia astratta del dispositivo.*

Definizione (Controller)

Il **controller** *è il dispositivo hardware che accetta richieste da parte del SO e le soddisfa, dialogando con il dispositivo stesso.*

Gestione dei dispositivi di I/O



Esempio: lettura di un file

- **Driver**
  1. riceve in input una richiesta di lettura di file
  2. traduce la richiesta in istruzioni per lo spostamento della testina e la lettura di un particolare blocco
- **Controller**
  3. riceve dal driver le istruzioni per lo spostamento della testina e lettura di un blocco,
  4. le esegue nel momento migliore, secondo politiche interne
  5. restituisce il risultato direttamente alla RAM tramite DMA.

Driver e controller

- Il **driver** traduce in un insieme di passi a basso livello richieste ad alto livello.
  - Spesso supporta politiche di accesso/allocazione delle risorse.
  - Esempio: gestione di una coda di stampa.
- Il **controller** di solito implementa meccanismi semplici.
  - Può essere in grado di accettare richieste complesse e soddisfarle in più passi, eseguendo un programma.
  - Esempio: gestione di più dischi in RAID.

Meccanismi di comunicazione tra driver e controller

- **Programmed I/O**
  - Il processore verifica periodicamente se il risultato è pronto
  - Il processore copia il risultato in memoria
- **Interrupt-driven I/O**
  - Un interrupt avvisa il processore che il risultato è pronto
  - Il processore copia il risultato in memoria
- **Direct Memory Access (DMA)**
  - Un interrupt avvisa il processore che il risultato è pronto in memoria
- Alcune osservazioni:
  - Programmed I/O: utile solo se la CPU non ha altro da fare!
  - DMA: il meccanismo più efficiente per grandi quantità di dati.
  - Dispositivi che trasmettono pochi byte (mouse): anche interrupt-driven può andare.

Gestione della memoria secondaria

- Alla base di ogni operazione del sistema operativo
  (contiene codice e dati di ogni applicativo)
- Obiettivo principale è l'**ottimizzazione**:
  - **Gestione della coda di richieste**
    - Esempio: meccanismo dell'ascensore piuttosto che FCFS
    - Utilizzo: movimento delle testine degli HD
  - **Meccanismi di caching**
    - Mantenere in RAM i dati cui si è acceduto più spesso
    - Necessario garantire consistenza delle informazioni

Parte VIII

File system

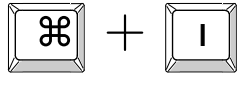
Il file system: un'astrazione

- I computer possono utilizzare **diversi media** per registrare in modo permanente le informazioni
  - esempi: dischi rigidi, floppy, nastri, dischi ottici
  - ognuno di questi media ha **caratteristiche fisiche diverse**
- Un **file system** nasconde la complessità dei diversi media proponendo una astrazione:
  - indipendente dal supporto di memorizzazione
  - efficiente
  - conveniente da usare
- **Per l'utente**, un file system è composto da **due elementi**:
  - **file**: l'**unità logica di memorizzazione**;
  - **directory** (**folder**): un insieme di informazioni per **organizzare i file** che compongono un file system

File

- Ogni file è caratterizzato da un insieme di **attributi**:
  - **nome** del file
  - informazioni su **locazione** e **dimensione**
    - difficile memorizzare su blocchi contigui $\Rightarrow$ frammentazione
    - utilizzo di **indici** per tener traccia dei blocchi occupati
  - informazioni **temporali**: creazione, ultima modifica, ultimo accesso
  - **proprietà**: per associare un file a un proprietario (**owner**)
    - quali sono i **privilegi** dell'owner sul file
    - chi ha il permesso di **leggere**, **modificare**, **eseguire** un file
  - **tipo**: file, immagini, ... Per indicare il tipo:
    - uso di **estensioni** del nome, es. `Lucidi_07.pdf`
    - informazioni **nel contenuto**, es. primi caratteri: `%PDF...`

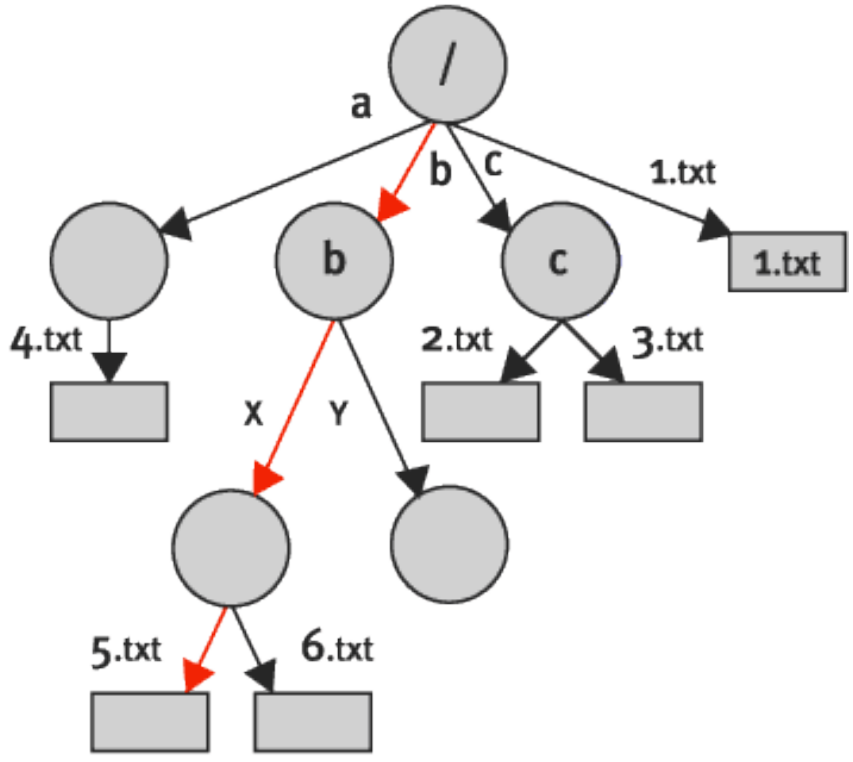
Come operare sui file

- Funzioni del SO per operare con i file:
  - **Creazione di un file**: `create()`
  - **Apertura/chiusura di un file**: `open()`/`close()`
  - **Lettura/modifica**: `read()`/`write()`
  - **Posizionamento**: `lseek()`
  - **Cancellazione**: `unlink()`
  - **Modifica degli attributi**: `chmod()`, `chown()`
- L'**apertura** serve per caricare tutte le informazioni sulla gestione di un file in alcune strutture dati del SO
- La **lettura** e **scrittura** avvengono tramite l'astrazione di una **testina di lettura**
- Operazioni comuni per un utente:
  - esecuzione di un programma (es. *double click*)
  - spostamento di un file (es. *drag & drop*)
  - visualizzazione di informazioni sul file (es. 

)

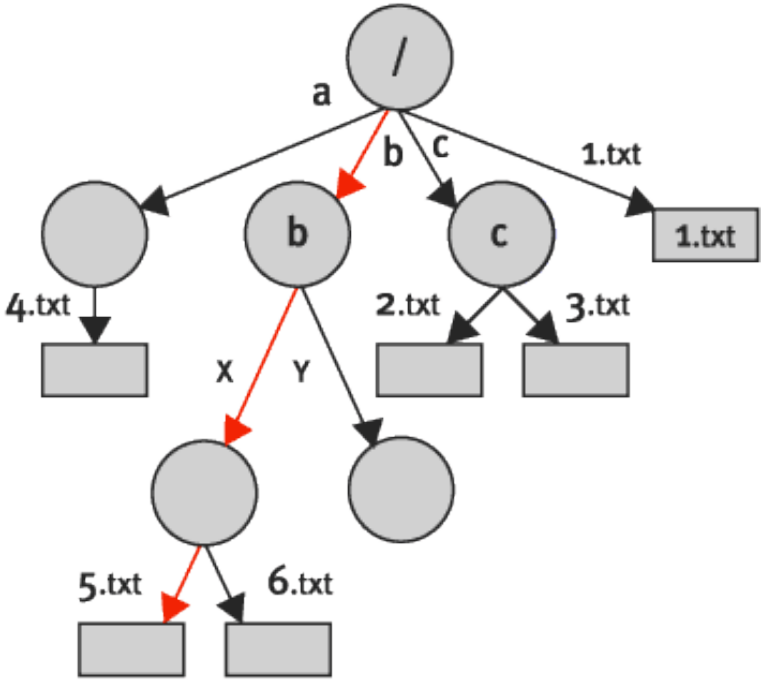
Directory

- Concetto fondamentale per **organizzazione gerarchica**
  - Struttura **ad albero**, parte da un direttorio **radice**
  - Directory $\Rightarrow$ **nodi**, file $\Rightarrow$ **foglie**
  - Percorso (**path**) per raggiungere un file:
    - Relativo: a partire dal direttorio corrente (`.`)
    - Assoluto: a partire dal direttorio radice (`/`)

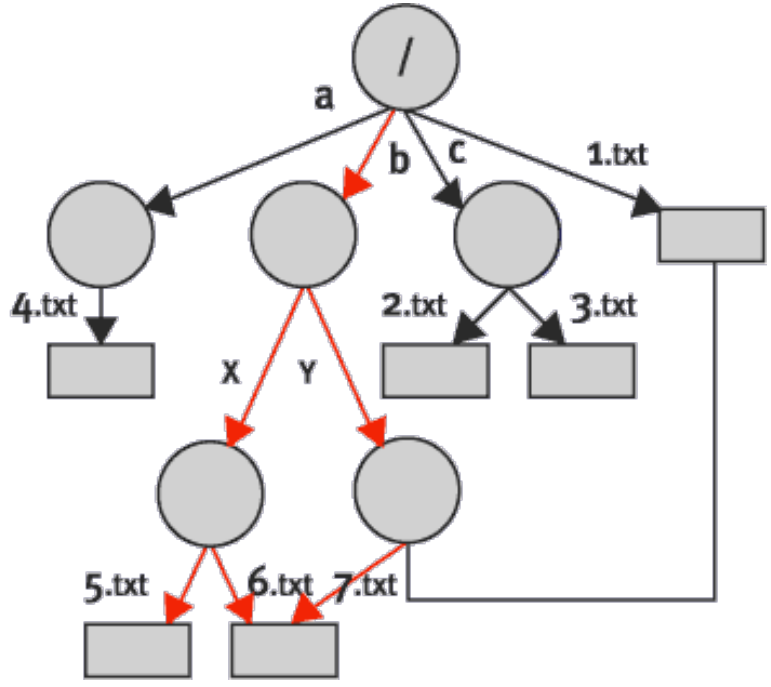


Alberi e grafi

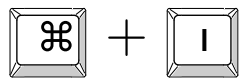
- Un solo percorso assoluto per raggiungere un file
- NTFS, FAT (*Windows*)



- Più percorsi assoluti per raggiungere un file
- MAC OS, UNIX: **link**



Come operare sulle directory

- Funzioni del SO per operare con i file:
  - **Creazione** di una directory: `mkdir`
  - **Modifica del direttorio corrente**: `cd`
  - **Visualizzazione del direttorio corrente**: `pwd`
  - **Modifica** di una directory $\Leftrightarrow$ **creazione/cancellazione di un file**: `create()`, `unlink()`
  - **Modifica degli attributi di una directory** `chmod()`, `chown()`
  - **"Esecuzione" di una directory $\Leftrightarrow$ transito** `cd`
  - **Linking di un file**: `ln`
- Operazioni comuni per un utente:
  - modifica del direttorio corrente (es. *double click*)
  - rinominazione di un direttorio (es. *click*)
  - visualizzazione di informazioni sulla directory (es. 

)

Parte IX

La sicurezza nei sistemi operativi

Sicurezza

- Problemi di sicurezza visti per Internet:
  - **Disponibilità.** Ciò che inviamo viene ricevuto dal destinatario?
  - **Confidenzialità.** Ciò che inviamo viene letto solo dal destinatario?
  - **Autenticità.** Sappiamo con certezza chi è il mittente?
  - **Integrità.** Sappiamo che il documento non è stato modificato nel tragitto?
- In un Sistema Operativo, il problema della sicurezza:
  - è l'insieme di meccanismi che vengono utilizzati per il **controllo di accesso alle risorse**
  - in generale, coinvolge **non solo il sistema di calcolo**, ma anche aspetti amministrativi e legali

Problemi fondamentali relativi alla sicurezza

- **Autenticazione**: associare ad un utente l'identità
- **Autorizzazione**: verificare se un utente ha il diritto di compiere un'operazione
- **Protezione**: evitare che un'operazione venga compiuta da chi non ne ha i diritti

Rendere sicuro un sistema

- Quali tipi di attacchi attesi?
  - **Attacchi passivi**: accedere a contenuto senza modificarlo
  - **Attacchi attivi**: forzare l'integrità o disponibilità del sistema.
- **Politica di sicurezza** deve essere adeguata al **valore** del sistema da proteggere.
- Tipi di attacchi attesi variano da sistema a sistema:
  - Banche: attacchi mirati a sottrarre denaro
  - Siti Web: denial of service o modifica del contenuto
- Valutare anche il **costo** di una particolare politica di sicurezza

Autenticazione

- Meccanismi che devono essere basati sull'utente
- Possono concentrarsi su tre tipi di elementi:
  - Qualcosa che l'utente **sa**: un **dato segreto concordato** in precedenza (password o PIN)
  - Qualcosa che l'utente **possiede**: un **oggetto riconoscibile** da parte della macchina (scheda magnetica o smart card)
  - Qualcosa che l'utente **è**: conformità di una **caratteristica fisiologica o comportamentale** con un dato "biometrico" di riferimento.
- Tecniche con **sicurezza** e **costo** crescenti.
  - Password: tecnica più largamente utilizzata
  - Bancomat: tessera magnetica + PIN
  - Ingresso in banca: a volte, riconoscimento della retina

Debolezze dei sistemi basati su password

- Molti problemi causati da una scarsa cultura della sicurezza.
  - Dove conservare la password?
  - Password facili da ricordare e/o difficili da indovinare?
  - Attacchi brute-force e basati su dizionario
  - Password "sbirciata" di nascosto
  - Login spoofing
  - Sniffing (per password trasmesse in chiaro)
  - Keystroke logging, metodi acustici, elettromagnetici, telecamere
    - Anche per le normali chiavi! (key bitting), `http://vision.ucsd.edu/~blaxton/sneakey.html`
  - Parecchie idee su: `www.schneier.com`

Autorizzazione

- Due tipi di entità nei Sistemi Operativi:
  - **entità attive**: processi
  - **entità passive**: risorse, quali file, aree di memoria, ...
- Entità attive compiono operazioni sulle entità passive **per conto di utenti**.
- Compiti del meccanismo di autorizzazione:
  1. permettere agli utenti di **specificare le azioni** che un'entità attiva può compiere o meno su un'entità passiva
  2. a ogni **richiesta** di svolgere un'azione svolte da partedi un'entità attiva, **verificare** se è ammissibile
- Tale meccanismo **sovrintende ogni chiamata di sistema**
- Principio del **privilegio minimo**: consentire a ciascun processo il minimo indispensabile dei privilegi per svolgere un compito
  - Esempio: `passwd`

Protezione

- Per rappresentare l'**insieme di regole di autorizzazione**, si utilizza il concetto di **dominio di protezione**
  - Descrizione di cosa si può fare e cosa no su una certa entità
  - Insieme di coppie <*Entità passiva*, *operazione consentita*>

Definizione (dominio di protezione)

Un dominio di protezione è un insieme di associazioni (coppie) che descrivono un insieme di entità passive e i tipi di operazioni che possono essere effettuati su ognuna di esse.

- Ogni processo opera all'interno di un **dominio di protezione**
- In ciascun dominio di protezione possono trovarsi 0, 1 o più utenti e/o processi.

Realizzazione dei domini di protezione

- Varia a seconda del SO
- Windows: **Access Control List**
  - Ogni entità passiva è associata ad una lista, che contiene delle coppie <*dominio di protezione*, *operazione consentita*>
- UNIX/Linux: UGO +/- RWX
  - Ogni entità passiva è associata a due identificatori: **utente proprietario** (UID) e **gruppo di utenti** (GID)
  - Si divide il mondo dei processi in tre insiemi:
    - User (U)
    - Group (G)
    - Others (O)
  - A ciascuno di questi insiemi vengono garantiti/negati diritti di
    - Lettura (R)
    - Scrittura (W)
    - Esecuzione (X)

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**
- **Backdoor**
- **Worm**
- **Virus**
- **Rootkit**
- **Grayware (spyware e adware)**
- **Dialer**

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
  - Scopo: garantire accesso da remoto
  - Atttività più frequenti: spedire SPAM, rubare, modificare e/o distruggere dati (tra cui: password), caricare e scaricare file, spiare l'attività dell'utente
  - Portata del potenziale danno dipende dai privilegi della vittima
- **Backdoor**
- **Worm**
- **Virus**
- **Rootkit**
- **Grayware (spyware e adware)**
- **Dialer**

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
  - Tipicamente una debolezza già presente nel SO (senza installare programmi aggiuntivi)
- **Worm**
- **Virus**
- **Rootkit**
- **Grayware (spyware e adware)**
- **Dialer**

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
- **Worm**: un programma autoreplicante, che si diffonde autonomamente senza intervento degli utenti del sistema
  - usa la rete per inviare copie di se stesso ad altri computer
    - es: usando rubrica e-mail per selezionare future vittime
  - non è attaccato a un programma esistente
  - tipico effetto: malfunzionamenti dovuti a consumo di risorse
- **Virus**
- **Rootkit**
- **Grayware (spyware e adware)**
- **Dialer**

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
- **Worm**: un programma autoreplicante, che si diffonde autonomamente senza intervento degli utenti del sistema
- **Virus**: un programma autoreplicante che si installa all'interno di un programma legittino
  - non viene eseguito autonomamente
  - worm e virus possono rimanere in letargo per periodi di tempo
- **Rootkit**
- **Grayware (spyware e adware)**
- **Dialer**

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
- **Worm**: un programma autoreplicante, che si diffonde autonomamente senza intervento degli utenti del sistema
- **Virus**: un programma autoreplicante che si installa all'interno di un programma legittino
- **Rootkit**: un sistema software di uno o più programmi con lo scopo di mascherare una avvenuta presa di controllo
  - attaccano il SO, e si installano come driver o moduli di kernel
  - spesso capaci di eludere anti-virus e anti-spyware
  - uso di backdoor
- **Grayware (spyware e adware)**
- **Dialer**

Malware

- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
- **Worm**: un programma autoreplicante, che si diffonde autonomamente senza intervento degli utenti del sistema
- **Virus**: un programma autoreplicante che si installa all'interno di un programma legittino
- **Rootkit**: un sistema software di uno o più programmi con lo scopo di mascherare una avvenuta presa di controllo
- **Grayware (spyware e adware)**: alcune applicazioni fastidiose o indesiderate ma non particolarmente nocive
  - **Spyware**: registra consuetudini di navigazione in rete (**profiling**)
  - rischio di registrare informazioni private
  - **Adware**: software che mostra banner nei browser Web
- **Dialer**

Malware

- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
- **Worm**: un programma autoreplicante, che si diffonde autonomamente senza intervento degli utenti del sistema
- **Virus**: un programma autoreplicante che si installa all'interno di un programma legittino
- **Rootkit**: un sistema software di uno o più programmi con lo scopo di mascherare una avvenuta presa di controllo
- **Grayware (spyware e adware)**: alcune applicazioni fastidiose o indesiderate ma non particolarmente nocive
- **Dialer**: programmi che stabiliscono una connessione a pagamento senza che la vittima se ne accorga
  - tipico scopo: ricavo da una connessione a pagamento provocata verso un numero costoso

Come ripararsi?

- http://www.cs.unibo.it/~babaoglu/courses/security06-07/lucidi/schneier.pdf

Parte X

Strumenti di amministrazione

Strumenti di amministrazione

- Interfacce **grafiche** vs. **testuali**
  - linguaggi di **scripting** per procedure più complesse
- Procedure di amministrazione di un sistema:
  - gestione degli utenti
  - configurazione, manutenzione e aggiornamento del SO
  - manutenzione dei dispositivi (driver)
  - installazione del software applicativo
  - specifica e implementazione delle politiche di sicurezza
  - backup periodico
- Sempre di più: procedure automatizzate
  - auto-update, time machine, ...

Modulo C, Moduli D.5-D.8