

# Lezione 2

## Cenni storici

Sistemi Operativi (9 CFU), CdL Informatica, A. A. 2022/2023

Dipartimento di Scienze Fisiche, Informatiche e Matematiche

Università di Modena e Reggio Emilia

<https://weblab.ing.unimo.it/people/andreolini/didattica/sistemi-operativi>

# Quote of the day

(Meditate, gente, meditate...)

**“Quien no recuerda el pasado, es destinado a repetirlo.”**

*George Santayana (1863-1952)*

*Filosofo e scrittore,*

*Reason in Common Sense*



# Definizione di Sistema Operativo

(Di che cosa stiamo parlando)

**Sistema Operativo (SO)**: software che agisce da intermediario fra l'utente e l'hardware del calcolatore stesso.

## Obiettivi

Fornire un ambiente di lavoro comodo.

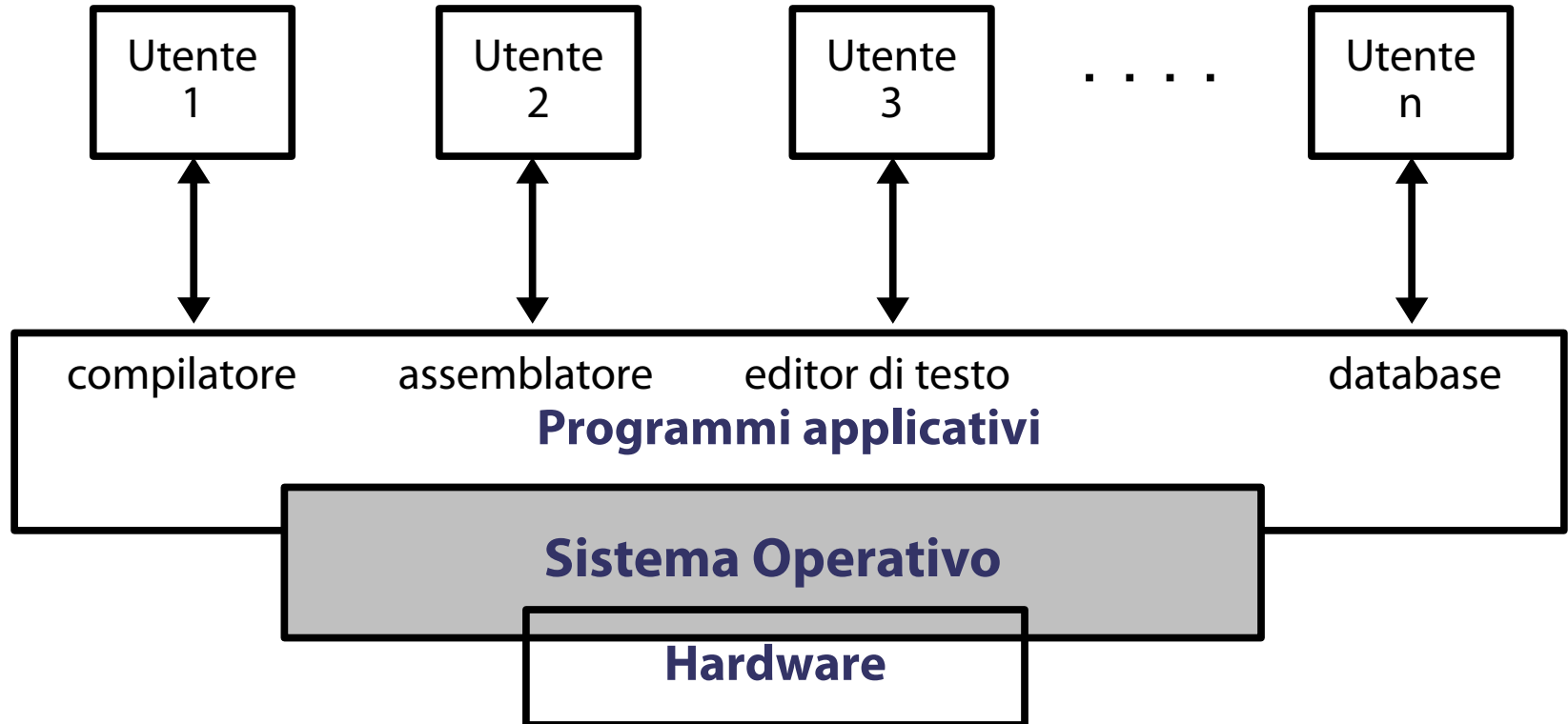
Fornire un ambiente di lavoro performante.

Fornire un ambiente di lavoro sicuro.

Mascherare i dettagli dell'hardware all'utente.

# SO in un sistema di calcolo

(Una immagine vale più di mille parole)



# Responsabilità di un SO 1/3

(Si farà mica un mazzo così?)

## Allocatore di risorse

Il SO alloca le risorse a sua disposizione (tempo di CPU, spazio su disco, porzioni di memoria) a programmi e utenti, in base alle necessità.

Il SO risolve i conflitti di assegnazione delle risorse in modo tale da operare efficientemente e correttamente.

# Responsabilità di un SO 2/3

(Eh, mi sa proprio di sì)

## Programma di controllo

Il SO è un gigantesco programma di controllo. Coordina l'esecuzione dei programmi, impedisce usi delle risorse non corretti, impedisce usi delle risorse maliziosi.

# Responsabilità di un SO 3/3

(Decisamente sì!)

## Creatore di astrazioni

Il SO si comporta come un abile prestigiatore: fornisce alle applicazioni l'illusione di essere padrone esclusive delle risorse della macchina (CPU, disco, memoria, rete, ...). In altre parole, alle applicazioni è fatta vedere una **macchina astratta** costituita dalle risorse ad esse assegnate. In realtà, il SO condivide le risorse realmente esistenti fra le diverse applicazioni.

# Un esempio

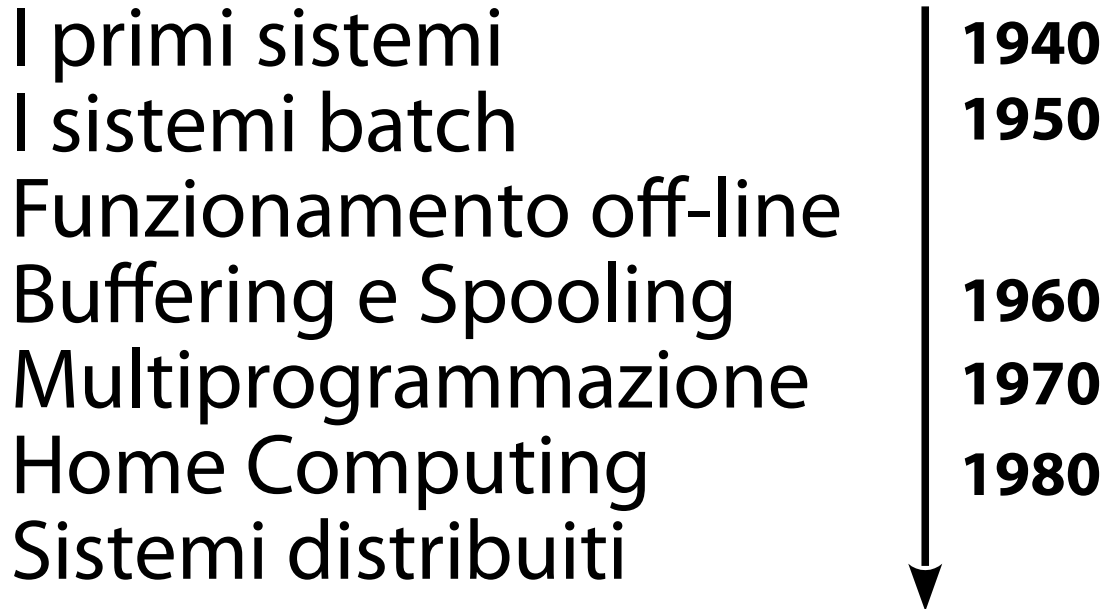
(Per provare a chiarire le idee)

Supponiate che il SO assegni all'incirca un terzo del tempo di CPU ad una applicazione. L'applicazione sembra vedere una CPU tutta per sé di potenza pari ad un terzo di quella reale. In realtà il SO sta eseguendo periodicamente porzioni di applicazioni in maniera concorrente; l'applicazione non vede nulla di tutto ciò.



# Evoluzione storica dei SO

(Back in the day...)



# **I PRIMI SISTEMI (1940-1960)**

# Caratteristiche

(Scaldapizzette giganteschi)

Macchine molto grandi, pilotate in maniera elettromeccanica (relais).

Input: switch elettrici, schede perforate

Output: Indicatori elettrici di stato, schede perforate

Ambiente operativo ad-hoc cablato nei circuiti.

Natura interattiva del sistema.

Programmatore  $\Leftrightarrow$  Operatore

Tempo di esecuzione della macchina gestito manualmente, tramite foglio prenotazioni.

# Macchina di Atanassov-Berry (1937)

(Il primo risolutore digitale di equazioni lineari)

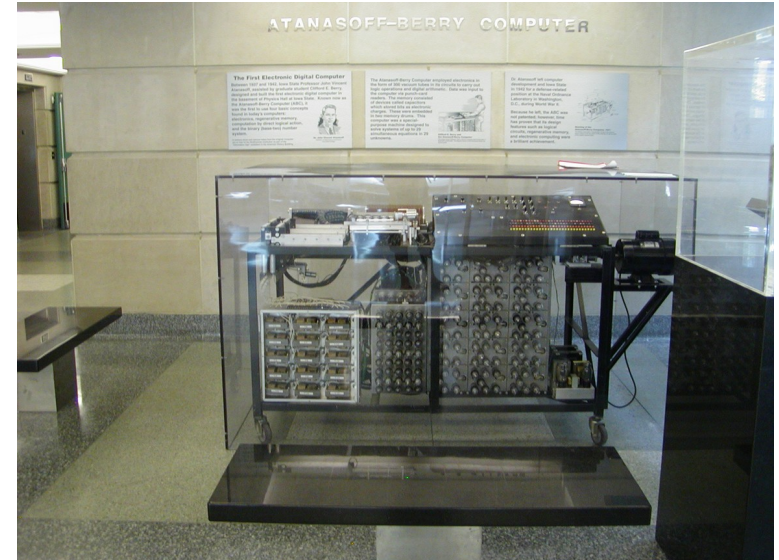
Uso di digit binari per la rappresentazione dei numeri.  
Calcolo tramite componenti elettronici.

Unità di calcolo separata da unità di memoria.

Uso di memoria "rigenerativa".

Assenza di stored program.

Assenza di SO.



# Colossus (1944)

(Colossus 1 – Germania Nazista 0)

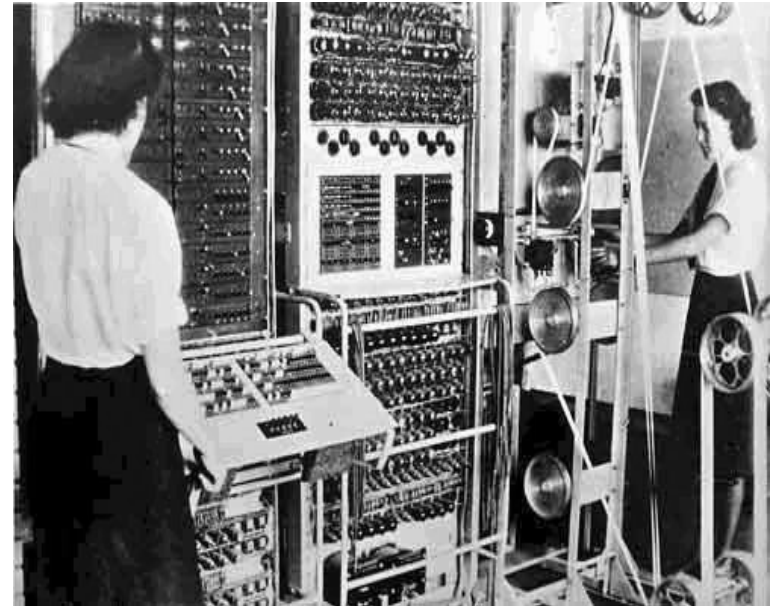
Primo calcolatore digitale britannico.

Sviluppato a Bletchley Park dal team di Alan Turing.

Fu in grado di rompere il codice cifrato ENIGMA, usato dai nazisti per le comunicazioni.

Programmabile mediante switch.

Assenza di SO.



# Alan Turing (1912-1954)

(Questo signore è stato probabilmente un pioniere...)

Ideatore del modello  
matematico alla base di un  
calcolatore “universale”:

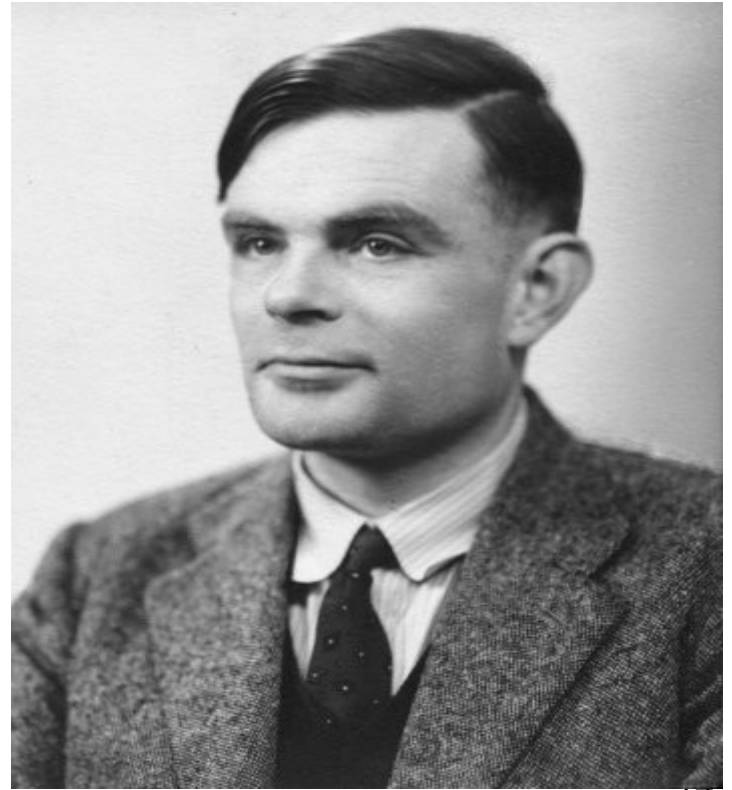
Macchina di Turing.

Pioniere della teoria della  
computazione.

Pioniere della crittoanalisi.

Pioniere della intelligenza  
artificiale (Test di Turing).

Pioniere della bioinformatica.



# John Von Neumann (1903-1957)

(Latinista, ellenista, matematico, ingegnere, pessimo autista, bevitore, viveur)

Inventore del modello  
architeturale di un moderno  
calcolatore.

Studioso dei sistemi di  
telecontrollo dei missili.

Fondatore della cosiddetta  
teoria dei giochi.



# Claude Elwood Shannon (1916-2001)

*(Suono di modem 56K che si connette ad Internet)*

Il padre della teoria dell'informazione.

Trasmissione digitale (1938).

Inventore del termine "bit" (1948).

Studio di ricostruzione e compressione dei segnali trasmessi.

Teorema del campionamento.

Fondatore della crittografia (1949).

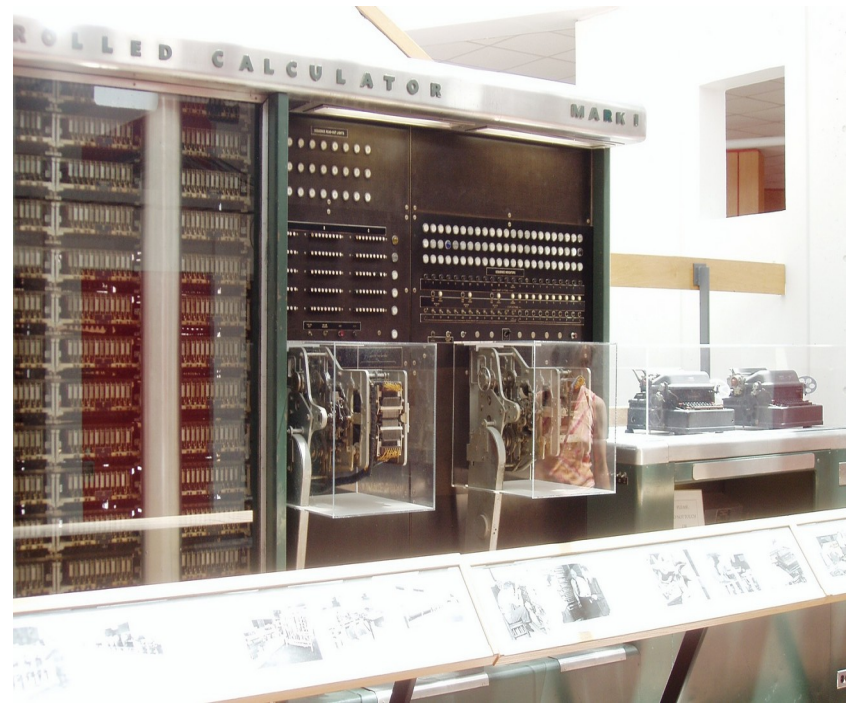




# Harvard Mark I (1944)

(Ruppe codici, calcolò le conseguenze della bomba atomica)

Progettato da Howard Aiken.  
Primo calcolatore in grado di eseguire calcoli complessi per un lungo periodo di tempo.  
Lungo 16m, alto 2.4m, profondo 0.5m, peso 4.5Ton, memoria 72 numeri da 23 cifre decimali.  
Assenza di SO.



# Grace Murray Hopper (1906-1992)

(L'ammiraglio che inventò i compilatori)

Primo programmatore di calcolatori digitali (Harvard Mark I).

Inventrice del COBOL.

Inventrice del termine "bug" (1946) riferito agli errori nel software.



# Computer bug (1946)

(Altro che C, C++, Python, Java, ...)

9/9


0800 Antman started  
1000 " stopped - antman ✓

1300 (032) MP-MC	1.2700	9.037 847 025
(033) PRO 2	<del>2.130476415</del>	9.037 846 895 correct
	2.130476415	4.615925059(-2)
	correct	2.130676415

Relays 6-2 in 033 failed special speed test  
in relay " " test.

Relays changed

1100 Started Cosine Tape (Sine check)  
1525 Started Multy Adder Test.

1545  Relay #70 Panel F  
(moth) in relay.

First actual case of bug being found.

1600 Antman started.  
1700 closed down.

Relay 2145  
Relay 3376

# Problemi dei primi sistemi

(Se deve intervenire l'uomo, il sistema non può performare bene)

Schema di prenotazione scomodo e inefficiente.  
Assenza di librerie generiche per la gestione delle risorse.

Immissione dei programmi scomoda.

Diagnosi e riparazione del sistema alquanto problematica.

→ Basso utilizzo delle risorse di calcolo

# Migliorie successive

(AKA "Più l'operatore sta lontano dal calcolatore, meglio è")

Creazione di un ambiente di sviluppo standardizzato.

"Device driver" per le periferiche.

Assemblatori, compilatori, linker per eseguibili.

Librerie di funzioni comuni.

Adozione di periferiche più avanzate

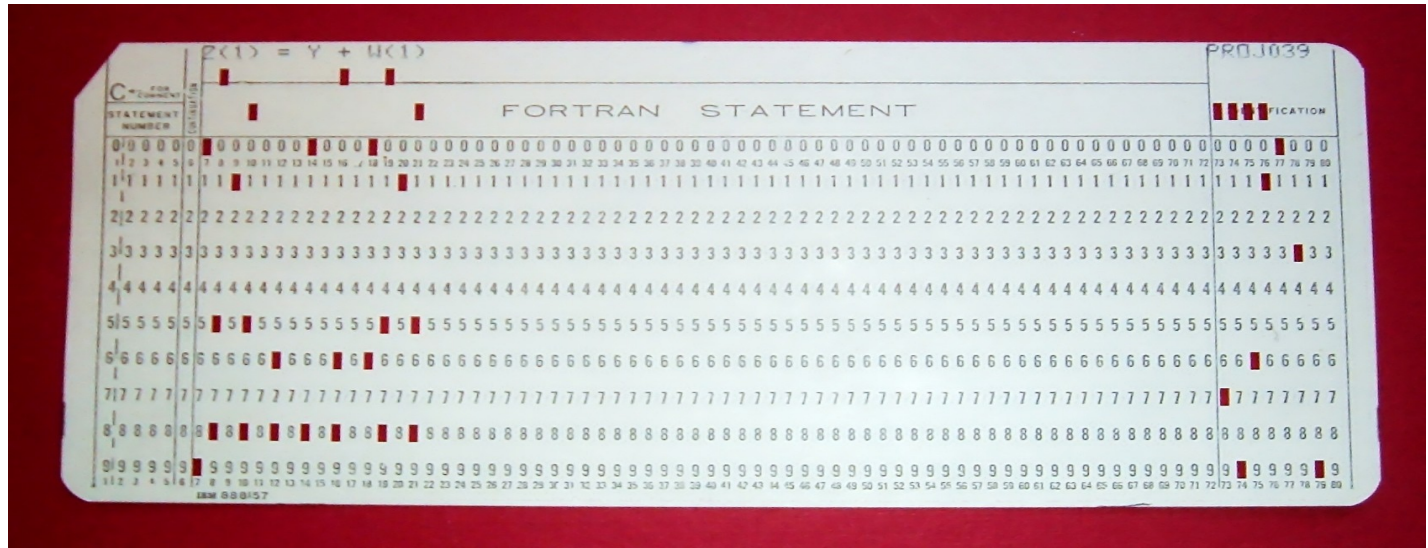
Lettori di schede, stampanti, nastri.

# Alcune periferiche innovative

(Innovative?!?)

## Schede perforate IBM (Fortran, 1964)

Codifica +  
Codifica -  
  
Codifica  
digit 0-9



Codifica di 80 cifre 0-9



# Alcune periferiche innovative

(Fanno molto James Bond)

## Unità nastro

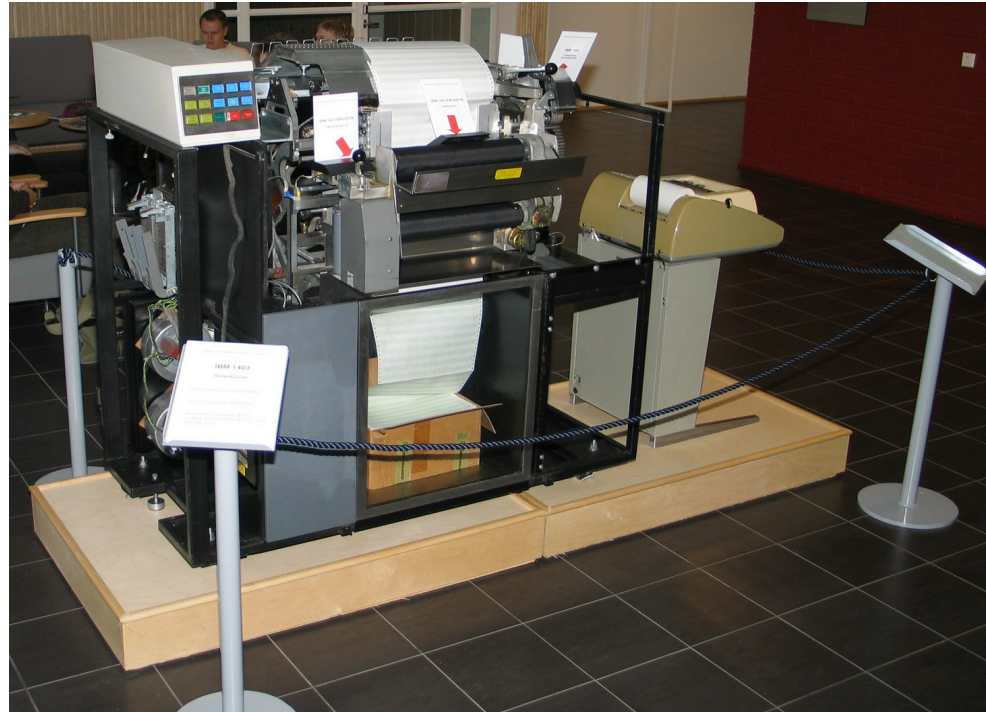
(Digital Equipment Corporation, 1964)



# Alcune periferiche innovative

(L'output del "Cervellone Elettronico")

Unità stampante  
(IBM 1403, 1959)





# I SISTEMI BATCH (1950-1960)

# Motivazioni legate al batch

(Il calcolatore diventa una macchina semi-autonoma)

**Idea di fondo:** risparmiare all'operatore le operazioni di gestione della macchina (complesse e soggette ad errore umano).

**Job:** un programma “impacchettato” in una sequenza di operazioni, memorizzate quasi tutte su schede perforate.

**Batch processing:** il calcolatore esegue una sequenza di job con ridotto (idealmente, nullo) ausilio dell'operatore.

# Operazioni di un job

(Un pacco di roba...)

- Inserimento e caricamento nastro compilatore.
- Esecuzione nastro compilatore.
- Rimozione nastro compilatore.
- Inserimento e caricamento nastro assemblatore.
- Esecuzione nastro assemblatore.
- Rimozione nastro assemblatore.
- Inserimento e caricamento nastro programma oggetto.
- Esecuzione nastro programma oggetto.
- Rimozione nastro programma oggetto.

# Problemi introdotti dai job

(Altro che ridurre il lavoro dell'operatore; pare di stare in palestra)

Il tempo necessario per la preparazione di un job eccessivamente elevato.

L'esecuzione alternata di programmi scritti in linguaggi diversi è piuttosto scomoda.

L'interruzione di un job è problematica.

# Soluzioni

(Al problema dei job, ovviamente)

Uso di un operatore professionista (diverso dal programmatore), in grado di operare in maniera efficiente con le macchine.

Uso di un sequenzializzatore automatico dei job (**monitor residente**).

# I monitor residenti

(Un grande passo in avanti)

## Monitor residente

Memoria principale

Caricatore

---

Sequenzializzatore  
di job

---

Interprete delle  
schede di controllo

---

Area programmi  
utente

# I componenti di un monitor residente

(Caricatore? Sequenzializzatore? Interprete?)

**Caricatore:** si occupa del caricamento del programma in memoria.

**Sequenzializzatore:** si occupa della scelta del prossimo job da eseguire.

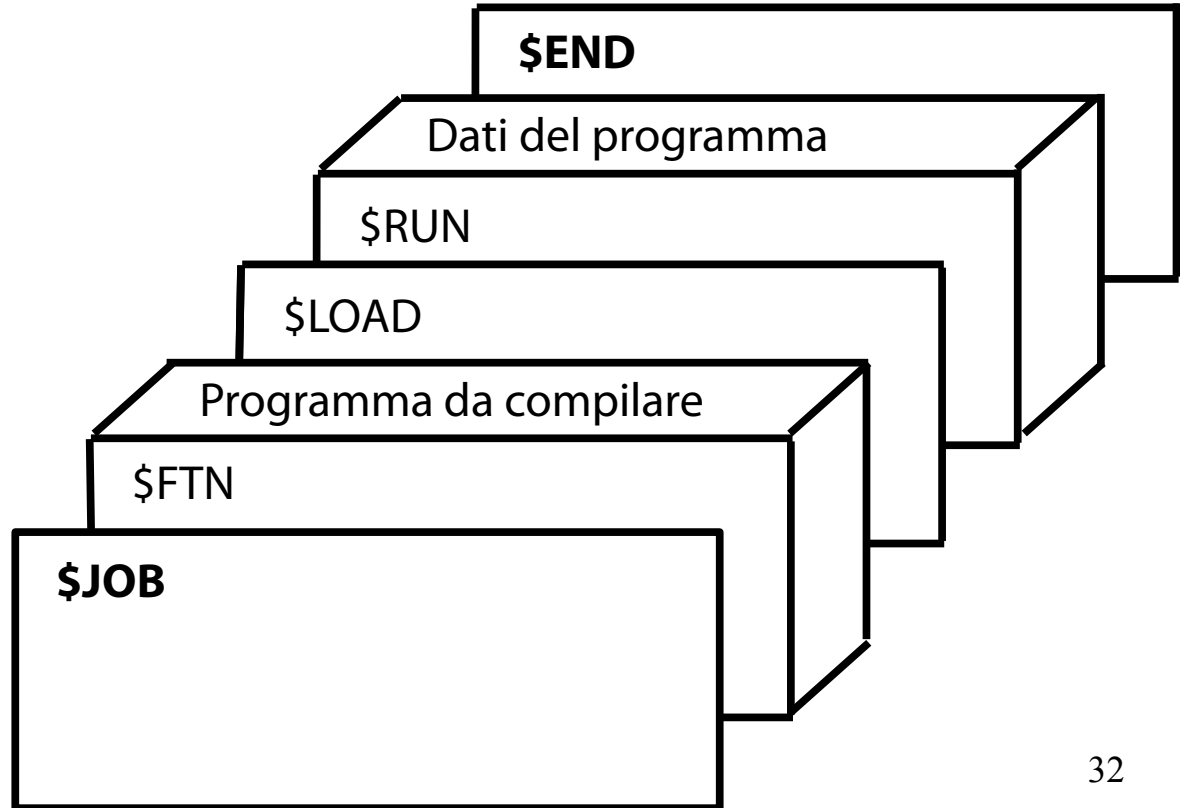
**Interprete:** gestisce le cosiddette “schede di controllo” che indicano al monitor quale programma eseguire.

# Le schede di controllo

(Caricatore? Sequenzializzatore? Interprete?)

Ciascun programma è descritto da una scheda di controllo che lo attiva.

Un job è delimitato da due schede speciali di controllo: **\$JOB**, **\$END**.





# Il pacco di schede

(Un'immagine continua a valere più di 1000 parole)



# Problemi legati ai sistemi batch

("Houston, we've had a problem")

Velocità I/O  $\ll$  velocità CPU.

Di conseguenza, la CPU è spesso inattiva, aspettando il completamento dell'I/O.

→ Tempi morti, consumo di potenza elettrica.

# **SOVRAPPOSIZIONE DELL'I/O (1950-1960)**

# L'idea di fondo

(Far calcolare alla CPU e far fare I/O alle periferiche simultaneamente)

Nelle tecniche per l'**I/O sovrapposto** si cerca di far lavorare simultaneamente il processore e la periferica di I/O.

→ Il disaccoppiamento fra I/O e calcolo dovrebbe permettere al processore di lavorare a pieno regime (posto che si abbia un numero sufficiente di job da sottometergli).

Esempio: CPU veloce il doppio rispetto al nastro

→ Due nastri usati in parallelo la tengono occupata tutto il tempo.

# Funzionamento offline

(La CPU non viene costantemente frenata da nastri e stampanti)

Nella modalità di funzionamento offline, l'elaboratore è separato fisicamente dalle periferiche.

Le schede perforate sono riversate su nastri veloci.

Il nastro veloce è montato sull'elaboratore.

La CPU legge i dati dal nastro veloce.

Il calcolatore riversa i dati su nastro veloce.

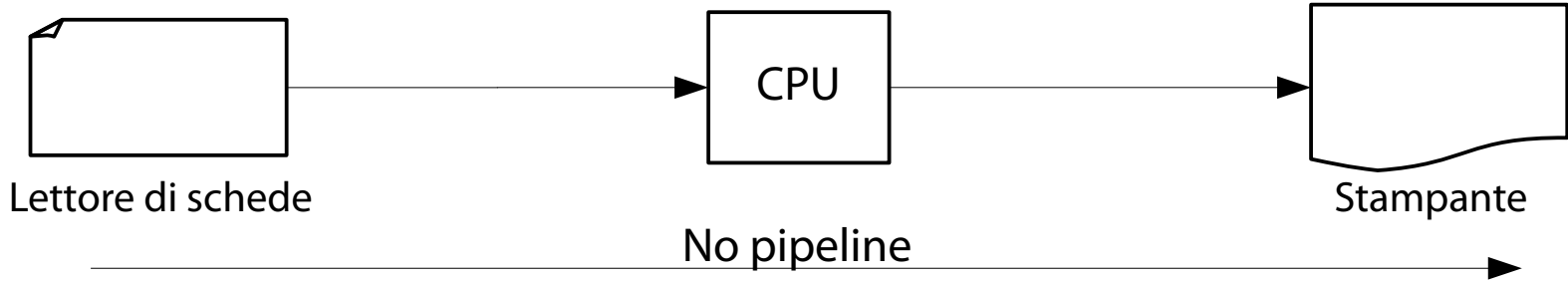
Il nastro veloce viene montato su una periferica di I/O.

Nel frattempo, il calcolatore riceve un nuovo nastro veloce → Funzionamento in pipeline.

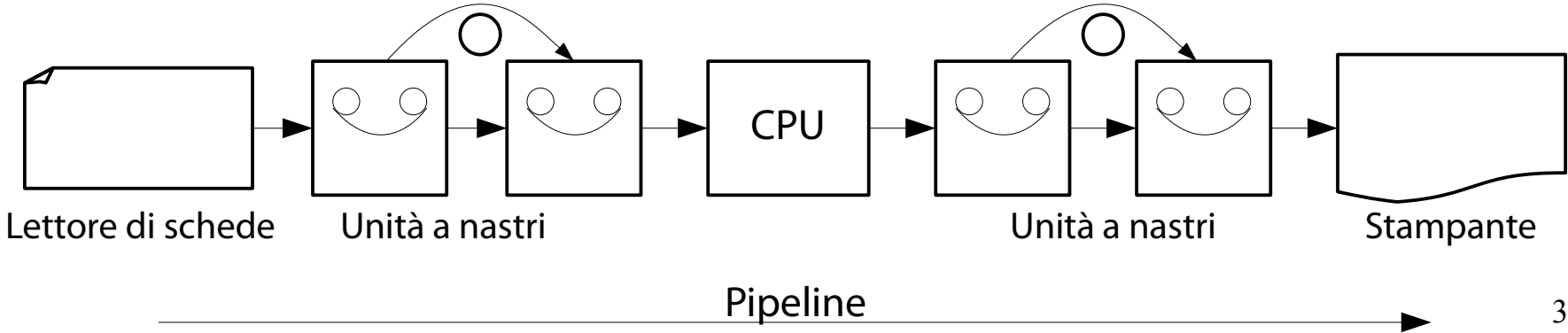
# Schema online vs. schema offline

(Un diagramma vale più di 1000 parole)

## Schema online



## Schema offline



# Un paragone con il mondo odierno

(Hmm... Unità lente, unità veloci, ...)

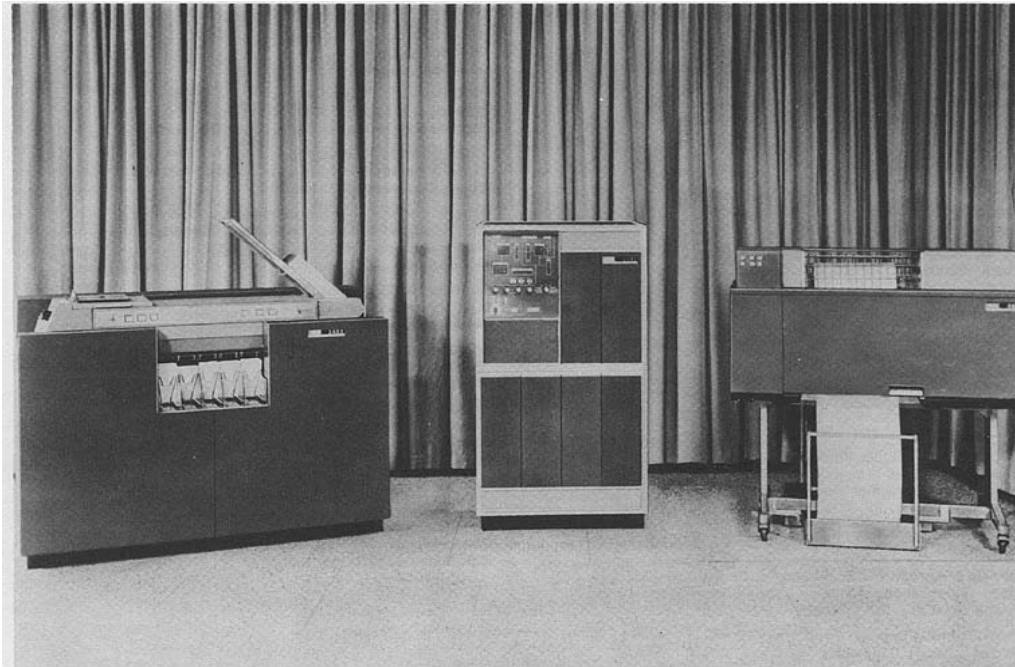
Unità lente → RAM DIMM.

Unità veloci → Memorie cache L1, L2.

# Separazione delle periferiche

(Le dimensioni non sono esattamente quelle di una cache L1...)

IBM  
(1959)



IBM 1402  
Card reader

IBM 1401  
Processor

IBM 1403  
Printer



# Problemi legati ai sistemi offline

(Si risolve un problema, se ne presenta un altro)

La modalità di funzionamento off-line richiede l'uso di più macchine separate.

→ Consumo elettrico ed ingombro maggiori.

Le procedure “fuori linea” di caricamento delle schede sui nastri e di riversamento dei nastri sulle stampanti devono essere svolte manualmente.

→ Operazioni lente e soggette all'errore umano.

Il nastro è acceduto sequenzialmente.

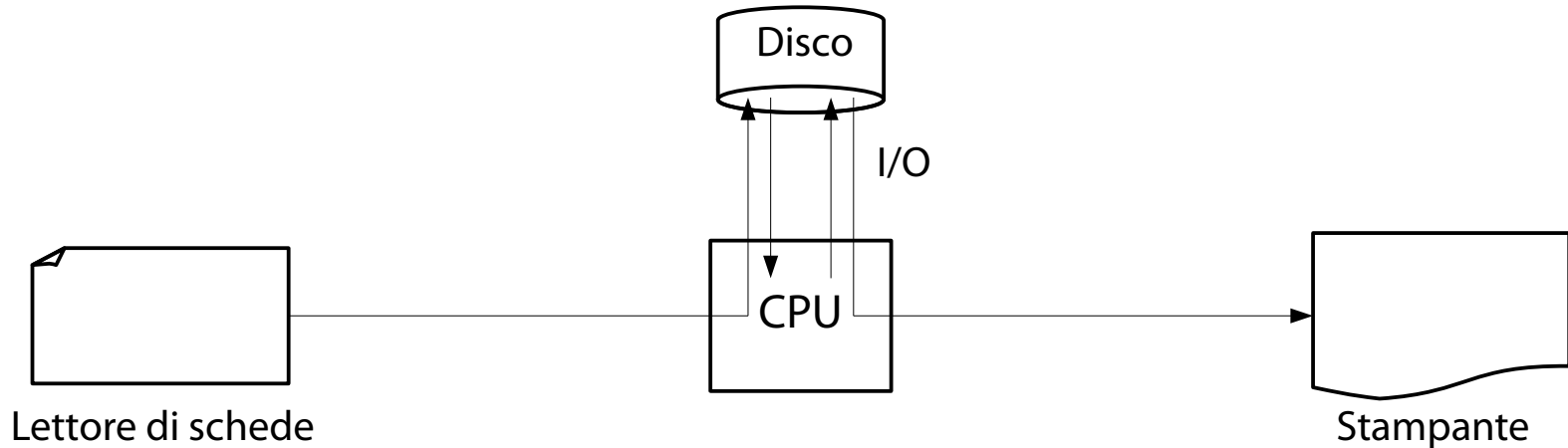
→ Se la CPU legge alla fine del nastro, le schede perforate non possono essere scritte al suo inizio (il nastro va riavvolto prima!).

# Soluzione: uso di dischi rigidi

(Si rimuove il vincolo di sequenzialità)

I nastri veloci sono sostituiti con **unità a disco rigido**, accessibili in maniera diretta.

- Cade il vincolo della sequenzialità.
- Aumentano le prestazioni di I/O.



# Hard disk IBM RAMAC 305 (1956)

(Capacità: 5M parole da sei bit, Tempo di accesso: 600ms)



# SPOOLING

(Arriva Superman)

Lo schema ora visto prende il nome di **SPOOL**.  
**SPOOL**: acronimo di **Simultaneous Peripheral Operations On Line**.

I dati sono riversati sotto forma di file temporanei sul disco.

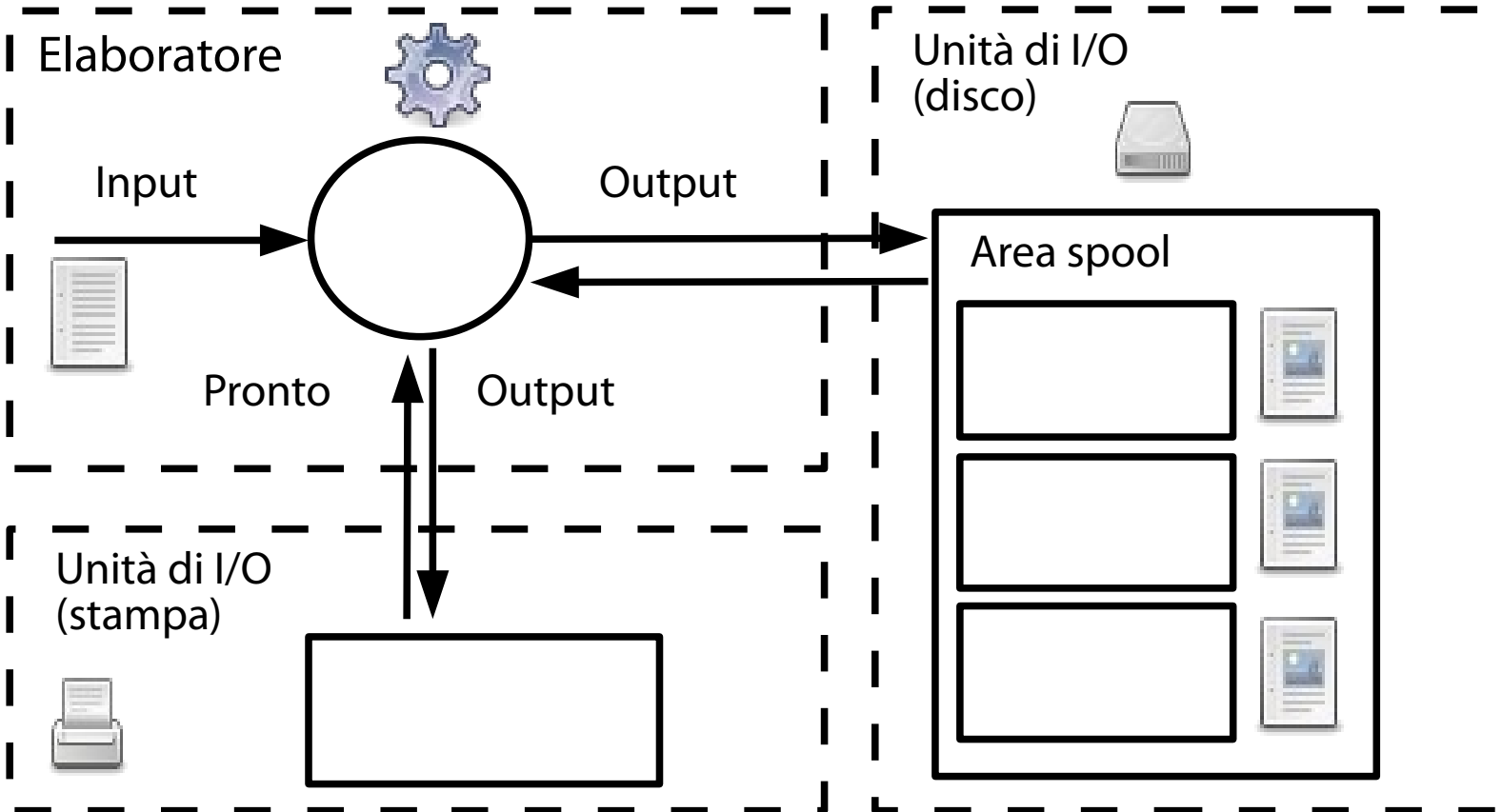
L'accesso è diretto.

Usato ancora oggi!

Posta elettronica, stampa.

# Schema di spooling

(Il disco è capiente ed accessibile casualmente)



# SPOOLING: vantaggi e svantaggi

(I/O sovrapposto, ma ancora )

## Vantaggi.

L'I/O è sovrapposto alla elaborazione → “multitasking”.  
L'utilizzazione della CPU aumenta considerevolmente  
(a patto di sottomettere un numero sufficiente di job).

## Svantaggi.

Il grado di sovrapposizione è ancora fortemente limitato dalla presenza delle periferiche lente (ad es., le schede perforate).

# **MULTIPROGRAMMAZIONE (1960)**

# Sistema multiprogrammato

(L'uomo conquista lo spazio)

**Idea di fondo:** i job sono eseguiti fino a quando non si verifica attesa sull'I/O (l'esecuzione non è più integrale).

Quando un job è in attesa di I/O, il sistema ripristina l'esecuzione di un altro job.

→ Finché c'è un job da eseguire, la CPU è attiva.

Estensione naturale dello SPOOL.



# Requisiti a livello di SO

(Che serve per andare sulla Luna?)

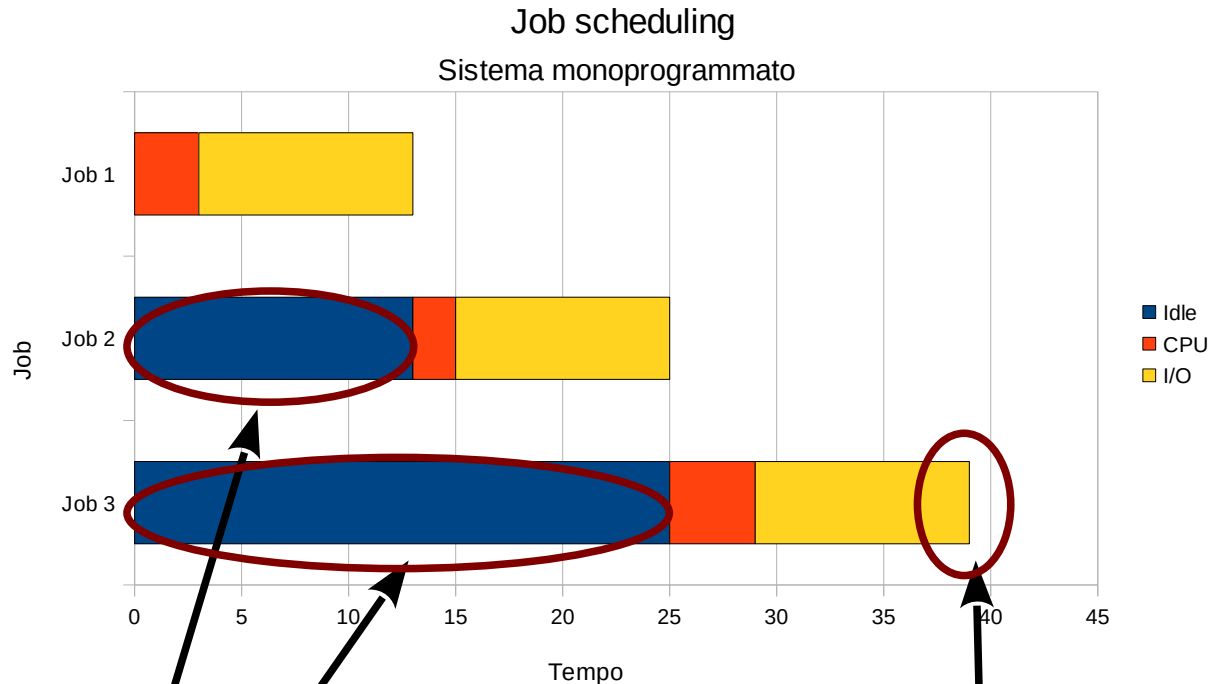
**Spooler:** uso di un disco rigido come area di memorizzazione degli input (provenienti da schede) e degli output (diretti alla stampante).

**Scheduler dei job:** decide quale job andrà in esecuzione in seguito all'interruzione di un job precedente.

**Gestore della memoria:** decide l'allocazione della memoria principale ai diversi job, evitando sovrapposizioni.

# Sistema monoprogrammato

(Prototipo di razzo di Werner Von Braun)



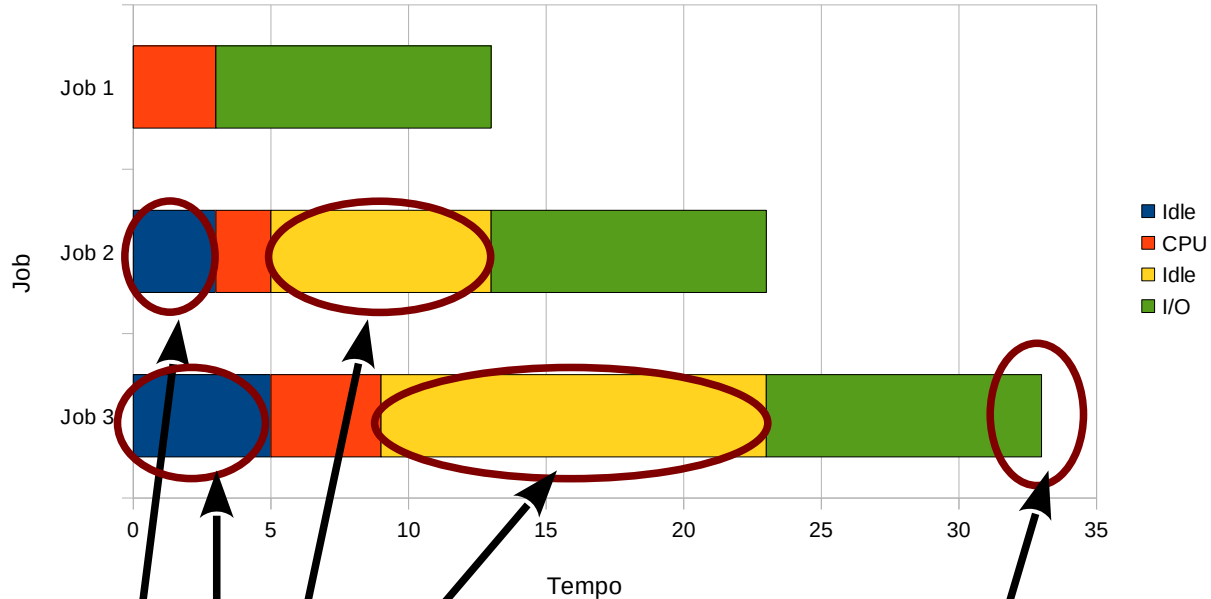
$13+25=38$   
istanti di idle

Finisce a 39!

# Sistema multiprogrammato

(Apollo 11)

Job scheduling  
Sistema multiprogrammato



$3+5+8+16=32$   
istanti di idle

Finisce a 33!

# Edsger W. Dijkstra (1930-2002)

(Uno dei più grandi)

Inventore del S O T H E.

Singolo utente.

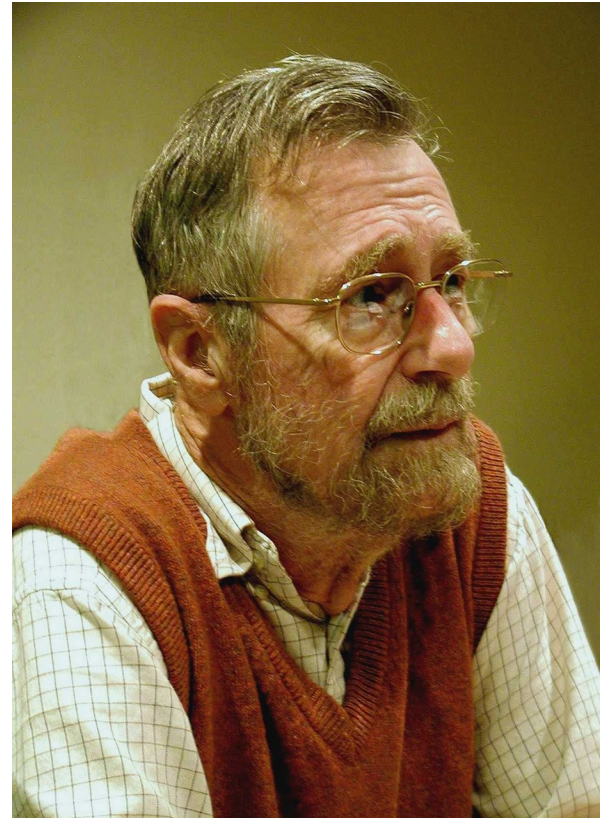
Multitasking.

Batch.

Ideatore dell'algoritmo per la scoperta dei cammini minimi su grafo.

Inventore del costrutto di sincronizzazione **semaforo**.

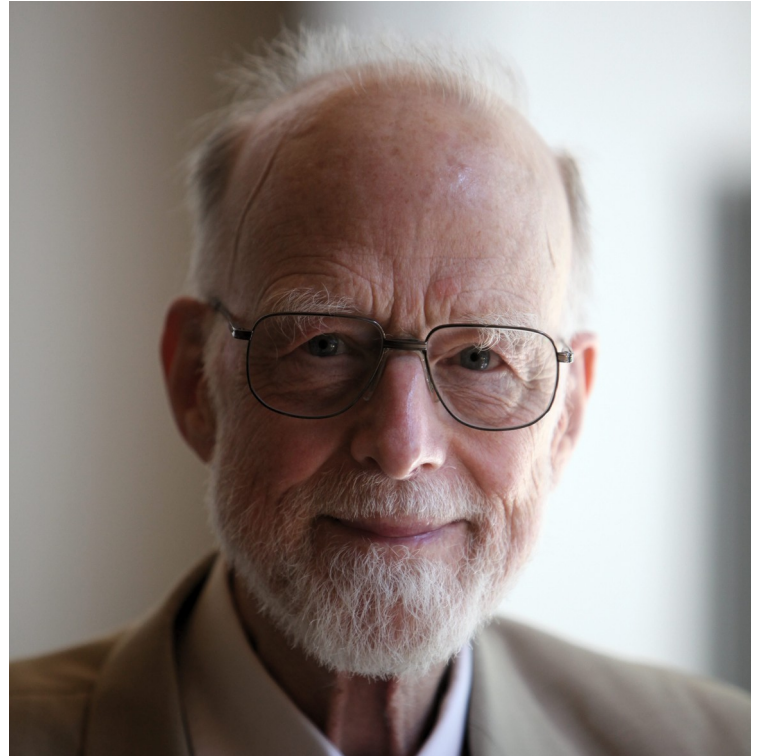
Inventore dell'algoritmo del banchiere per l'assegnazione delle risorse.



# Sir Charles Anthony Hoare (1934-)

(<https://www.youtube.com/watch?v=ywWBy6J5gz8>)

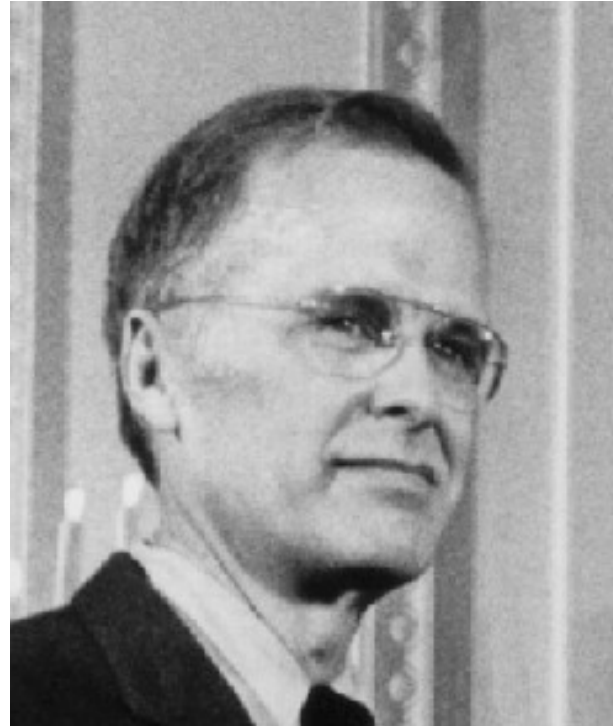
Ideatore del linguaggio di  
programmazione ALGOL 60.  
Ideatore dell'algoritmo di  
ordinamento Quick Sort.  
Inventore del costrutto di  
sincronizzazione **monitor**.



# John Backus (1924-2007)

("Can programming be liberated from the Von Neumann Style?")

Ideatore del linguaggio di programmazione FORTRAN.  
Ideatore del formalismo **Backus Naur Form** per la descrizione formale della sintassi di un linguaggio.  
Pioniere dello sviluppo dei compilatori.



# Problemi ancora irrisolti

(Sta diventando un incubo)

Mancanza di interattività fra macchina e utente durante l'esecuzione di un job.

Il programmatore può ricevere l'output di un job anche svariati giorni dopo la sua esecuzione.

# **TIME SHARING (1960)**



# Time sharing

(Jupiter and beyond the infinite)

La CPU esegue più job concorrentemente, in modalità multiprogrammata.

Ciascun job possiede, a turno, la CPU per un intervallo di tempo denominato **quanto**.

Se i quanti sono ragionevolmente piccoli

i job che fanno uso primario di CPU non stallano il sistema troppo a lungo.

il sistema dà una chance di esecuzione anche a quei job che fanno principalmente I/O.

# Vantaggi del Time Sharing

(Sistema enormemente più reattivo rispetto al batch)

Marcato aumento della **interattività** con l'utente.

Il terminale è reattivo ai comandi.

L'utente ha la sensazione di avere la macchina per sé.

Possibilità di implementare sistemi multi-utente con interfaccia basata su **terminale fisico**.

Il SO serve gli utenti in maniera "interlacciata".

# Terminale seriale

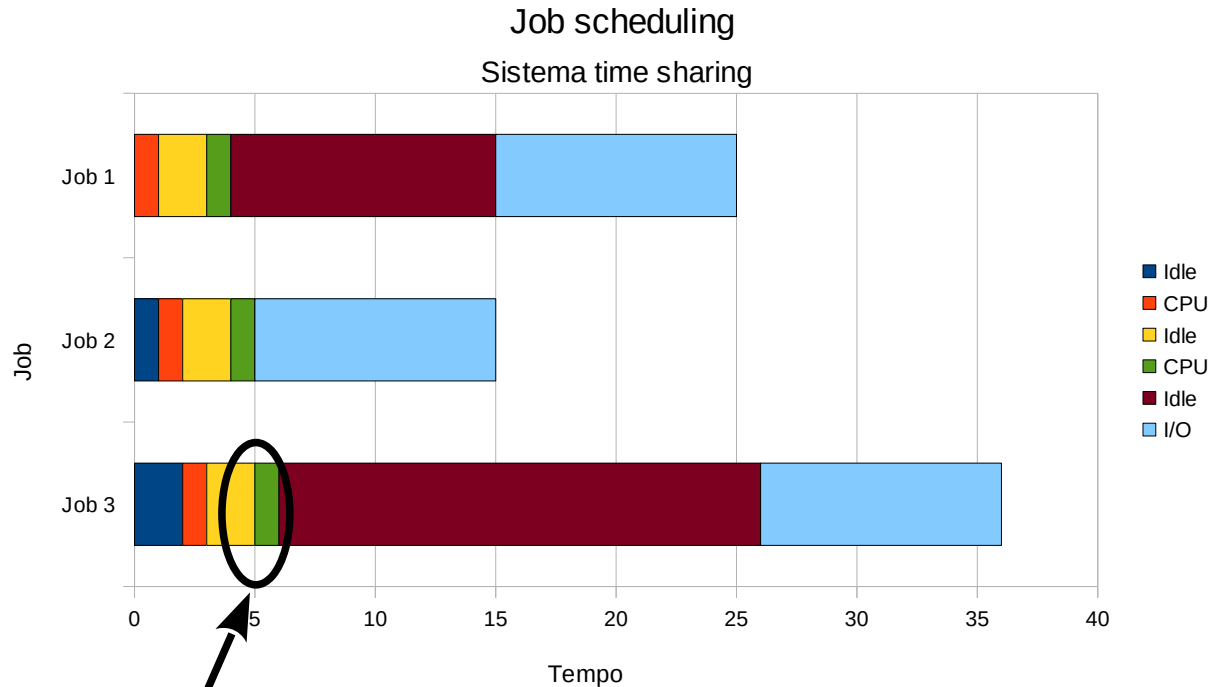
(Il mitico DEC VT-100 su cui hanno perso la vista generazioni di hacker)

DEC VT-100  
(Digital Equipment  
Corporation,  
novembre 1978)



# Sistema time sharing

(Enterprise NX-01)



L'uso della CPU termina a 6!

# MULTICS (1964-1969)

(Il cervellone elettronico della General Electric, nato al MIT)

Primo SO

time sharing.

multiprogrammato.

multiutente.

Antesignano di UNIX.

Non divenne mai  
veramente popolare.

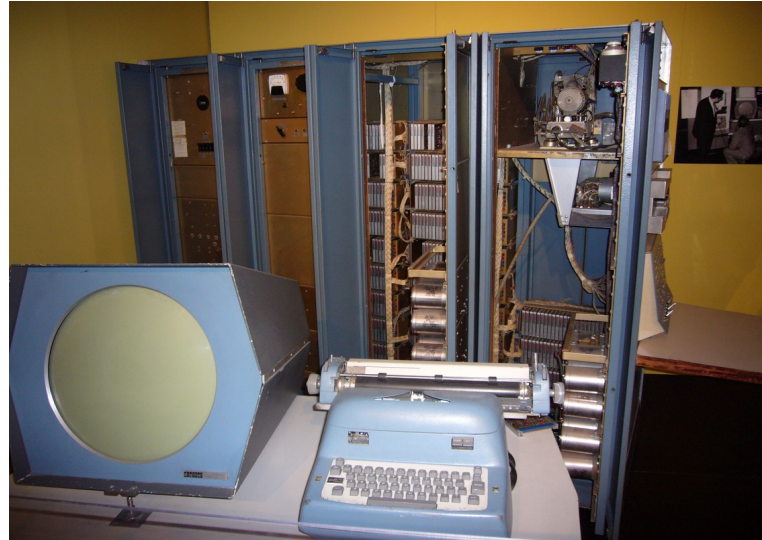
Progetto troppo  
complesso.



# DEC PDP-1 (1960)

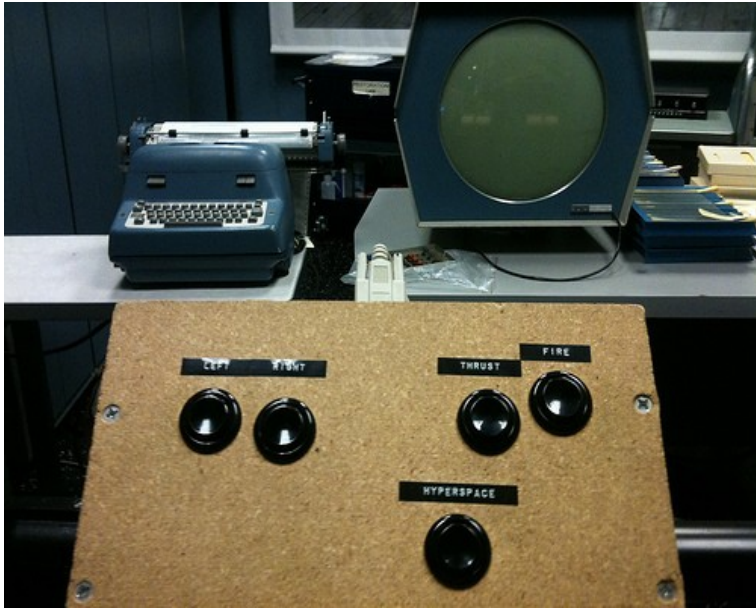
(Look ma! A video game!)

Programmed Data  
Processor I.  
Primo minicomputer a  
basso costo (120K\$).  
Primo SO in grado di  
eseguire un videogioco  
multi utenti real time.  
**Spacewar!** (1960).



# Spacewar! (1960)

(Look ma! A video game!)



# DEC PDP-10 (1964-1983)

(Look ma! A minicomputer for our campus!)

Il minicomputer che ha reso famoso il time sharing  
Usato in diverse università americane.

MIT AI Lab.

Stanford AI Lab.

Carnegie Mellon.

SO: TOPS-10.





# IBM System/370 (1970-1990)

(Una macchina molto vicina ai PC di oggi)

Memoria principale basata su circuiti integrati.  
Supporto per la memoria virtuale.  
Aritmetica floating point.  
Virtualizzazione.  
OS: VM/370.



# DEC VAX (1977-1989)

(Il concorrente principale dell'IBM S/370)

Estensione a 32 bit del PDP  
(**V**irtual **A**ddress e**X**tension).  
Paginazione su richiesta.  
OS: VMS.



# UNIX (1969-)

(Keep it simple and stupid, and you'll win)

Riscrittura semplificata di MULTICS.

Condotta su un vecchio PDP-7 (Bell Labs, AT&T).

Obiettivo del progetto: eseguire in maniera efficiente il videogioco "Space Travel".

Simulatore spaziale (movimento astri celesti, spostamento di una navicella).

Antenato del simulatore "Celestia".

# Ken Thompson (1943-)

(UNIX & C Wizard)

Coautore di UNIX.  
Ideatore del linguaggio B  
(precursore del C).  
Coautore del C.  
Autore dell'editor **ed** (antenato  
di **sed** e **vi**).  
Ideatore del sistema di codifica  
dei caratteri UTF-8.  
Co-inventore del linguaggio Go.



# Dennis Ritchie (1941-2011)

(C & UNIX Wizard)

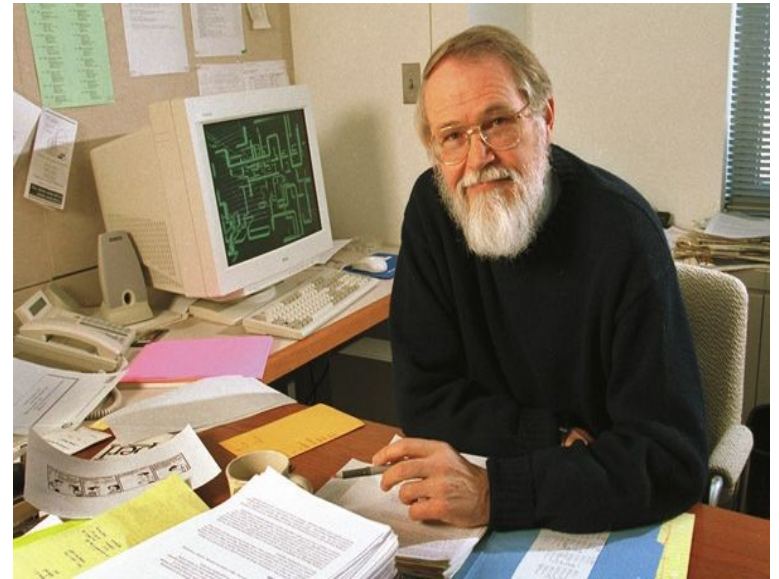
Coautore di UNIX.  
Coautore del C.  
Coautore del libro  
"The C programming language".  
Autore di Plan 9 (il SO  
"evoluzione" di UNIX).



# Brian Kernighan (1942-)

(Plumber guy)

Coautore del C.  
Coautore del libro "The C programming language".  
Coautore del linguaggio AWK (Aho, Weinberger, Kernighan).  
Autore di diverse utility UNIX.



# Bell Labs UNIX Time Sharing System

(The one and only UNICS TSS)

Prima distribuzione UNIX: UNIX Bell Labs (1974).

Eseguito su PDP-11.

Distribuzione completa di codice sorgente a costo zero per i ricercatori.

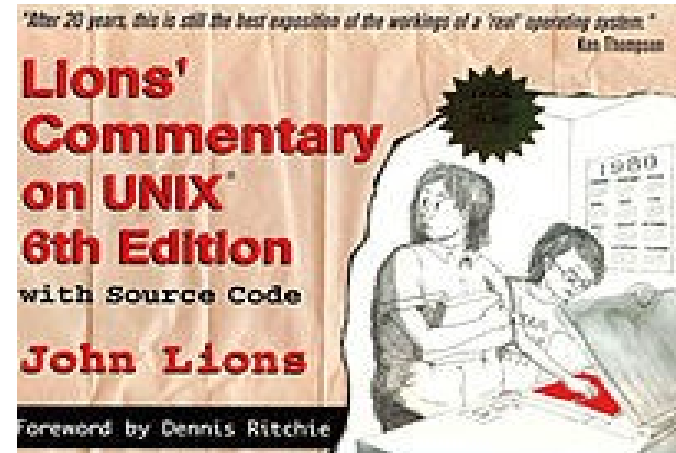
A partire dalla release v6, Bell decide di vietare la distribuzione del codice sorgente.

# John Lions (1937-1998)

(L'eroe rivoluzionario)

John Lions, ricercatore, decide di commentare l'intero codice di UNIX v6 e distribuisce in forma pirata il documento.

Tale documento, passato di generazione in generazione, è oggi un libro.





# La proliferazione dei sistemi UNIX

("La Diaspora")

Il Lions' Commentary ha permesso lo studio e l'implementazione di diverse versioni di UNIX in Università e centri di ricerca.

[http://en.wikipedia.org/wiki/File:Unix\\_history-simple.svg](http://en.wikipedia.org/wiki/File:Unix_history-simple.svg)

# Bill Joy (1954-)

(Hacker extraordinaire)

Creatore e gestore delle prime versioni di **BSD UNIX** (la release dell'Università di Berkeley).

Creatore dell'editor **vi**.

Fondatore di **Sun Microsystems** (1982).

Ha corretto, migliorato e portato su BSD 4.3 lo stack **TCP/IP**.

Primo UNIX a “navigare su Internet”.



# Richard Matthew Stallman (1953-)

(Saint Ignucius of the Church of Emacs)

Programmatore all'MIT di Boston (CSAIL).

Pioniere del Free Software Movement.

Creatore della Free Software Foundation.

Creatore del SO GNU (port di UNIX usufruibile gratuitamente).

Creatore di utility fondamentali quali GCC, GDB, Gmake, GNU Emacs.



# La grande pecca di GNU

(La grande fortuna di Linux)

GNU non era, al tempo, completato.

Mancava un pezzo fondamentale del SO: il **kernel**  
(il software che colloquia con le periferiche).

Era necessario installare GNU su un sistema UNIX  
esistente.

Fino a quando...

# Linus Benedict Torvalds (1969-)

(Ci mise una pezza, e che pezza...)

Nel 1991 comincia a scrivere un kernel sul suo 80386, "per hobby". Il kernel prenderà il nome di Linux e integrerà GNU. Si parlerà di **GNU/Linux**. È inoltre autore di:  
GIT.  
SubSurface.



# HOME COMPUTING (1970-)

# Il computer come strumento casalingo

(Video gaming at home! Hurray!)

Il calcolatore non viene più visto solo come uno strumento di calcolo. Si punta al mercato casalingo.

Videogiochi.

Small office.

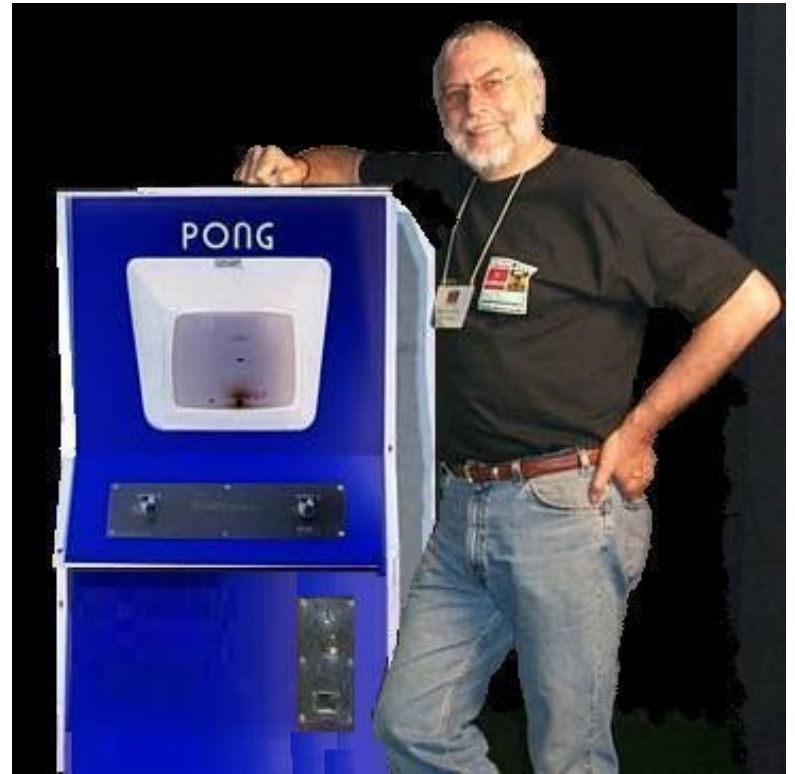
# Nolan Bushnell (1943-)

("ATARI, magari!" - spot del 1981)

Fondatore del primo grande colosso dei videogiochi.

Atari (1972).

Fu il primo ad intuire le potenzialità del mercato casalingo.





# Console videogiochi

(How it all started...)

Architetture hardware dedicate e poco costose.

Sistemi operativi dedicati, generalmente monoutente e monoprogrammati.

Sistemi programmabili in linguaggio macchina o in BASIC.

Periferiche alquanto primitive (nastri, floppy, ROM, joystick, video e audio).

# Console videogiochi

(Vecchie glorie)



Atari VCS2600



Commodore 64



Coleco Vision



Spectrum ZX81

# IBM Personal Computer (1981-1987)

(640KB ought to be enough for everybody)

Modello attuale di calcolatore pensato per uso domestico.

Contrapposizione con il mainframe.

Realizzato con componenti producibili facilmente su larga scala.

Enormemente più potente e versatile delle console precedenti.

Architettura aperta ed espandibile.

# PC compatibili (1985 ca -)

(La tomba del PC IBM)

I PC vengono “assemblati” a partire da componenti prodotti a basso costo.

Il mercato del PC originale crolla.

Alcune aziende si specializzano sulla produzione di determinati componenti.

Intel, AMD: processori.

Seagate, Western Digital, Samsung, Hitachi: hard disk.

Matrox, NVIDIA, SIS, ATI: schede video.

...

# Federico Faggin (1941-)

(Poche invenzioni, ma molto buone)

Inventore del primo processore Intel.

Intel 4004 (1971-1974).

Inventore del processore Z80.

Inventore del Touchpad.



# Vittime illustri dei PC compatibili

(Mors tua, vita mea)



Commodore Amiga 500



Atari 1040 ST

# William Henry Gates III (1955-)

(Bill Gates per gli amici)

Fondatore di Microsoft Corporation (1975).

Autore del SO MS-DOS (1982-2000).

Autore del SO Windows (1985-).



# I computer di oggi

(Mors tua, vita mea)



Desktop



Laptop



Tablet



Smartphone



# Problemi ancora irrisolti

(Help!)

Le prestazioni del singolo elaboratore peggiorano al crescere delle richieste.

Problemi di scalabilità delle prestazioni.

I dati utente non sono replicati.

Scarsa tolleranza ai guasti.

# **SISTEMI DISTRIBUITI (1969-)**

# Distribuzione del carico

(Disaccoppiare il sistema è la parola d'ordine!)

Distribuzione del calcolo su diversi nodi.

Sistemi multiprocessore (strettamente accoppiati).

Sistemi distribuiti in rete (debolmente accoppiati).

Vantaggi:

Condivisione delle risorse.

Accelerazione del calcolo (calcolo parallelo).

Aumento dell'affidabilità.

Comunicazione fra utenti.

# ARPANET (1969-)

(Internet)

Progetto finanziato dal DARPA.

**Obiettivo dichiarato:** creare una rete resistente alle rotture dei collegamenti.

Rete a commutazione di pacchetto.

I dati viaggiano fra tanti punti intermedi.

L'instradamento è deciso ad ogni punto.

Architettura modulare, eterogenea e scalabile.

Uso di protocolli di comunicazione per il colloquio fra dispositivi diversi.

# Leonard Kleinrock (1934-)

(Trasmise la stringa "login" da UCLA a Stanford)

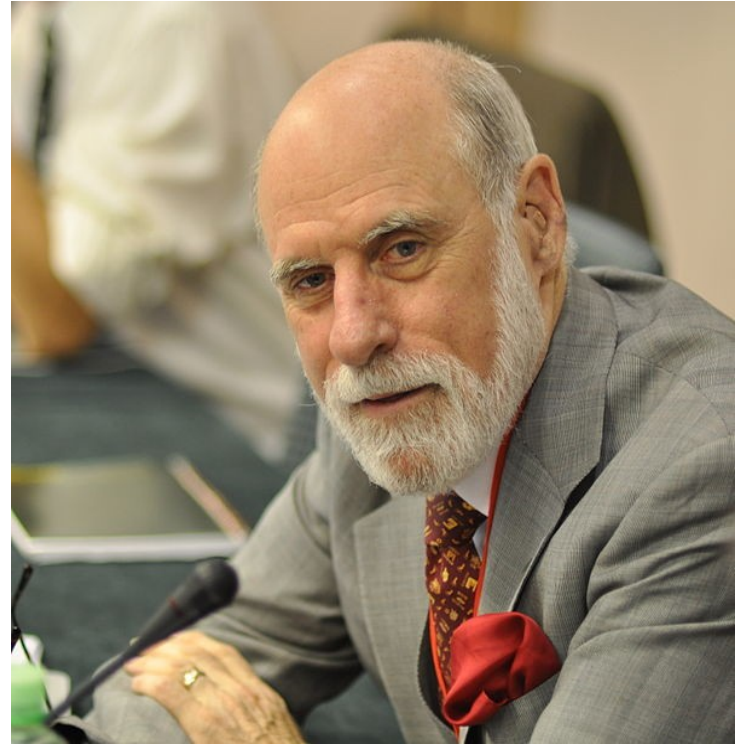
Fondatore della teoria delle code.  
Pioniere delle reti a commutazione di pacchetto.  
Fautore della prima comunicazione fra calcolatori (ottobre 1969).



# Vinton Cerf (1943-)

(Vicepresidente di Google)

Pioniere dei protocolli di  
trasmissione (TCP).  
Promotore della  
*Internet Society*.



# Robert Elliot Kahn (1938-)

(Grazie a lui SSH, FTP, HTTP, IRC, TELNET, ... sono una realtà)

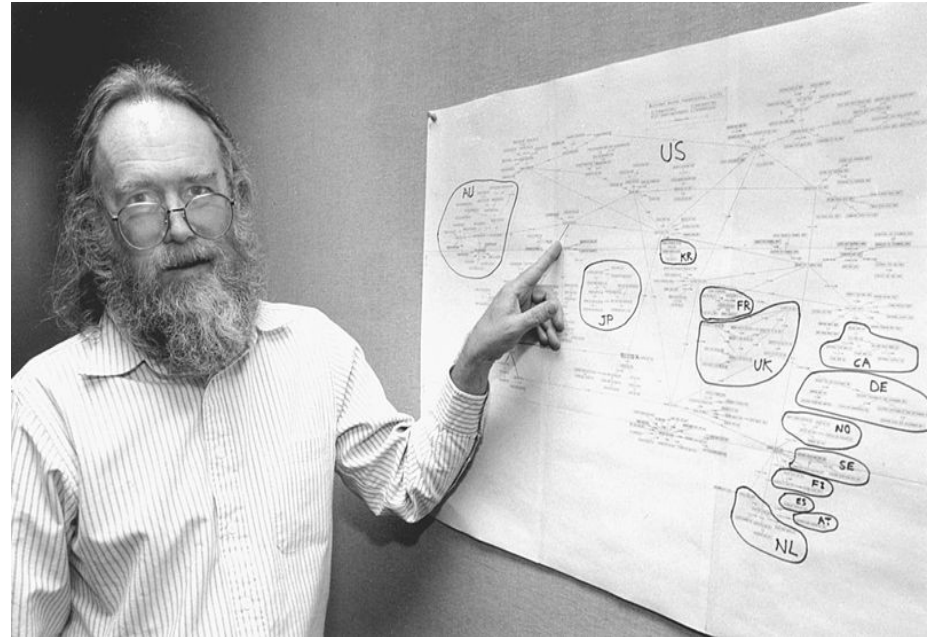
Pioniere dei protocolli di trasmissione (TCP).  
Fu il primo ad avere l'intuizione di un **protocollo di trasporto**.



# Jon Postel (1943-1998)

“A name indicates what we seek, an address indicates where it is, a route indicates how we get there”

Il padre dell'attuale Internet.  
Editor dei documenti RFC (**Request For Comment**).  
Membro del comitato di standardizzazione **Internet Engineering Task Force** (IETF).





# Tim Berners Lee (1955-)

(Un altro fisico che ha rivoluzionato il mondo)

Inventore del World Wide Web:  
la **killer application** di Internet.

**Obiettivo:** creazione di un  
ambiente di condivisione per gli  
articoli scientifici.

Collegamento ipertestuale fra  
documenti diversi (**hyperlink**).

