Sistemi Operativi: Introduzione

Amos Brocco, Ricercatore, DTI / ISIN

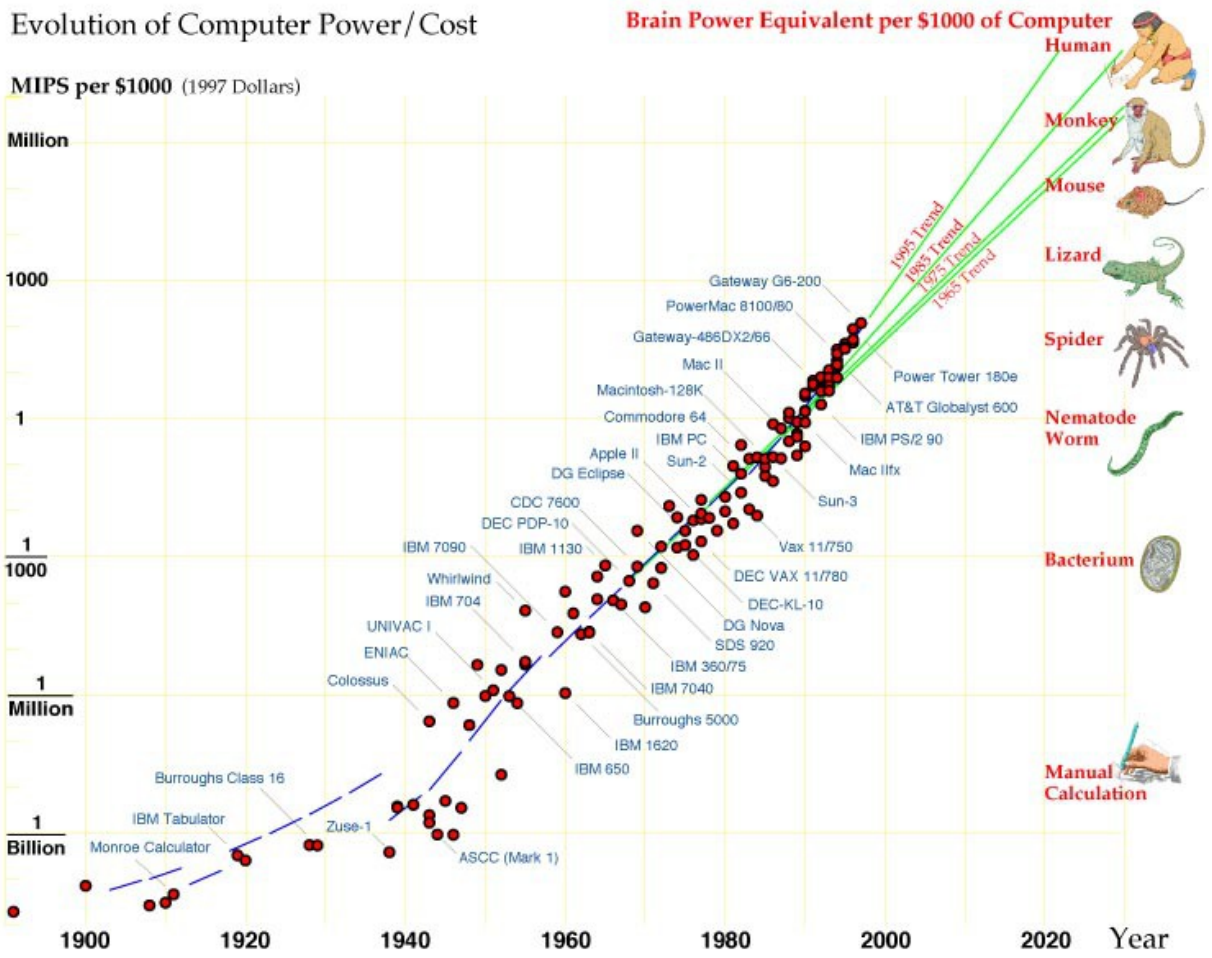
Basato su:

[STA08] “Operating Systems: Internals and Design Principles”, 6/E, William Stallings, Prentice Hall, 2008
[STA12] “Operating Systems: Internals and Design Principles”, 7/E, William Stallings, Prentice Hall, 2012
[TAN01] “Modern Operating Systems”, 2/E, Andrew S. Tanenbaum, Prentice Hall, 2001
[TAN09] “Modern Operating Systems”, 3/E, Andrew S. Tanenbaum, Prentice Hall, 2009
[SIL07] Operating Systems Concepts, Silberschatz, Galvin, Gagne, 7/E, Pearson, 2005

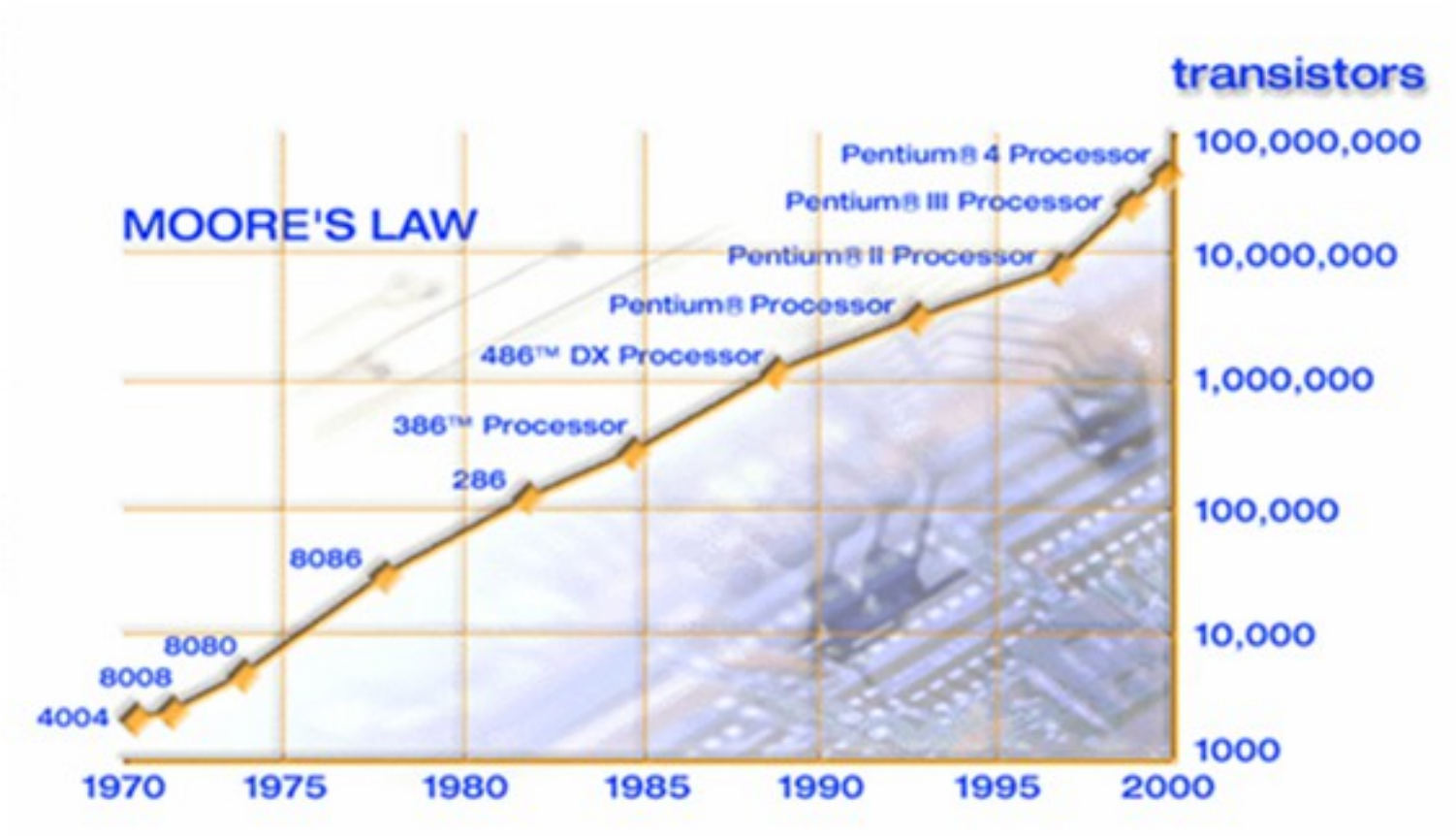
Obiettivi

- Sapere perché è necessario avere un sistema operativo
- Conoscere l'architettura di base di un computer e le problematiche ad essa legate
- Saper definire cos'è un sistema operativo
- Distinguere tra le diverse tipologie di sistema operativo

Evoluzione: performance



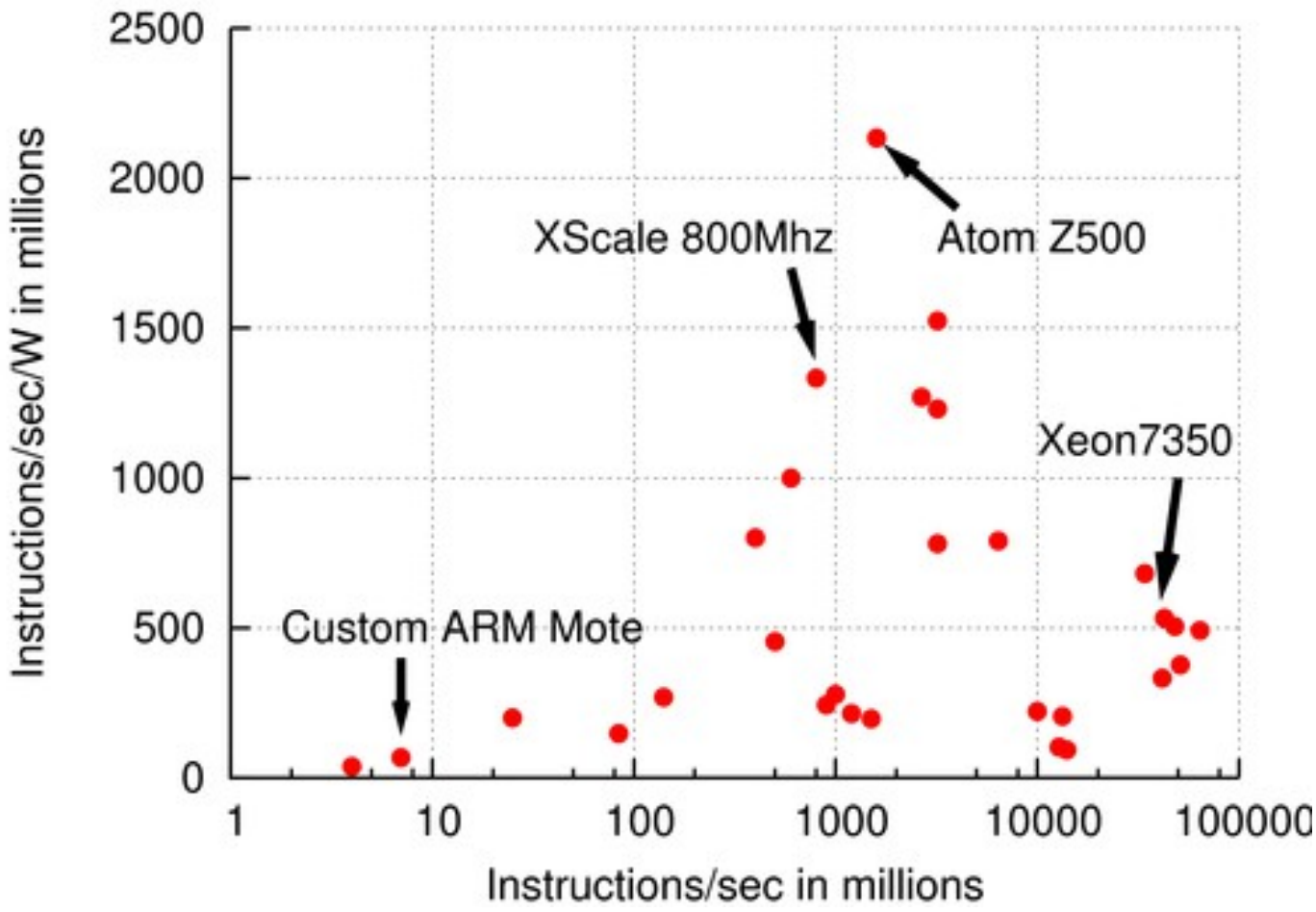
Evoluzione: livello di integrazione



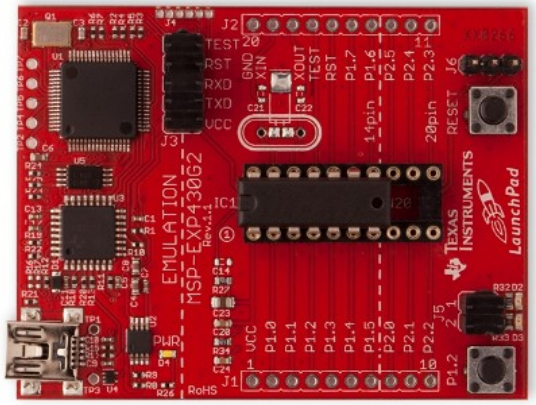
"Le prestazioni dei processori, e il numero di transistor ad esso relativo, raddoppiano ogni 18 mesi."

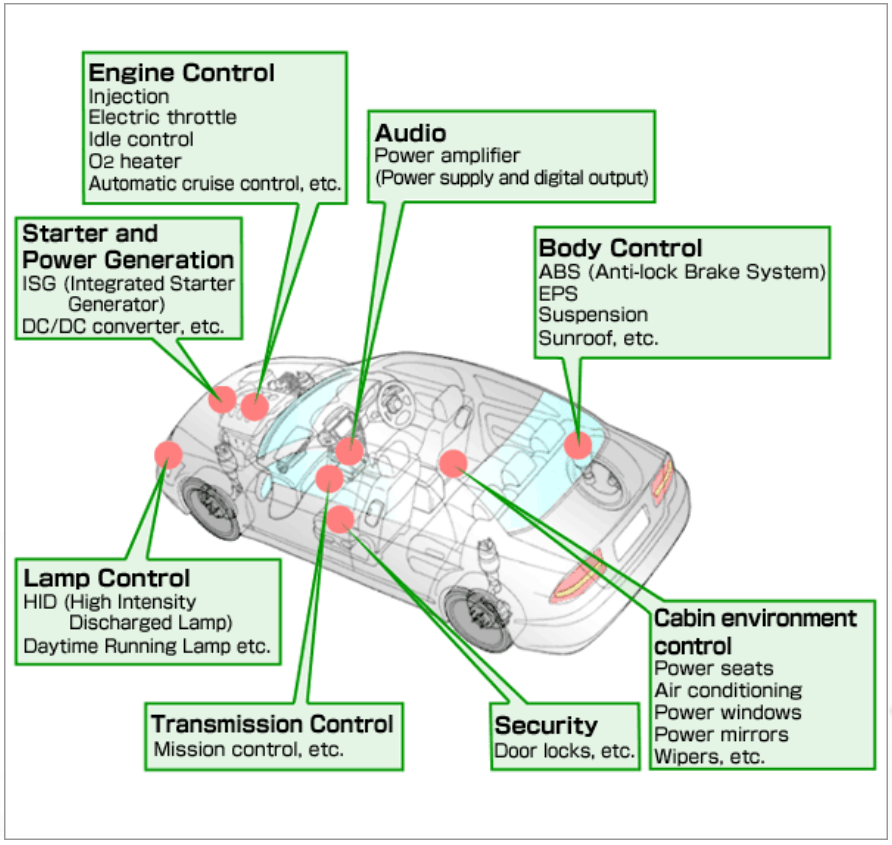
Limite: numero di transistor => consumo => calore

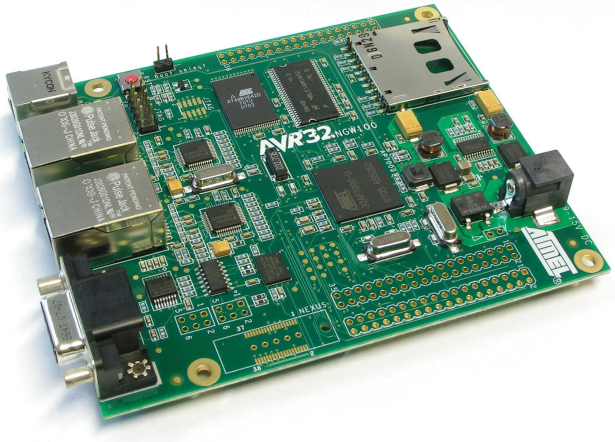
Evoluzione: efficienza



Evoluzione: diffusione

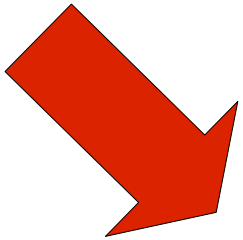






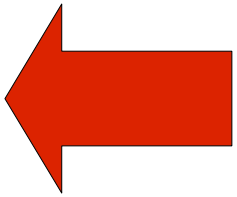
Evoluzione: il presente e il futuro

- Sistemi multi-core
- Virtualizzazione
- Calcolo parallelo su GPU
- Cloud computing
- ...



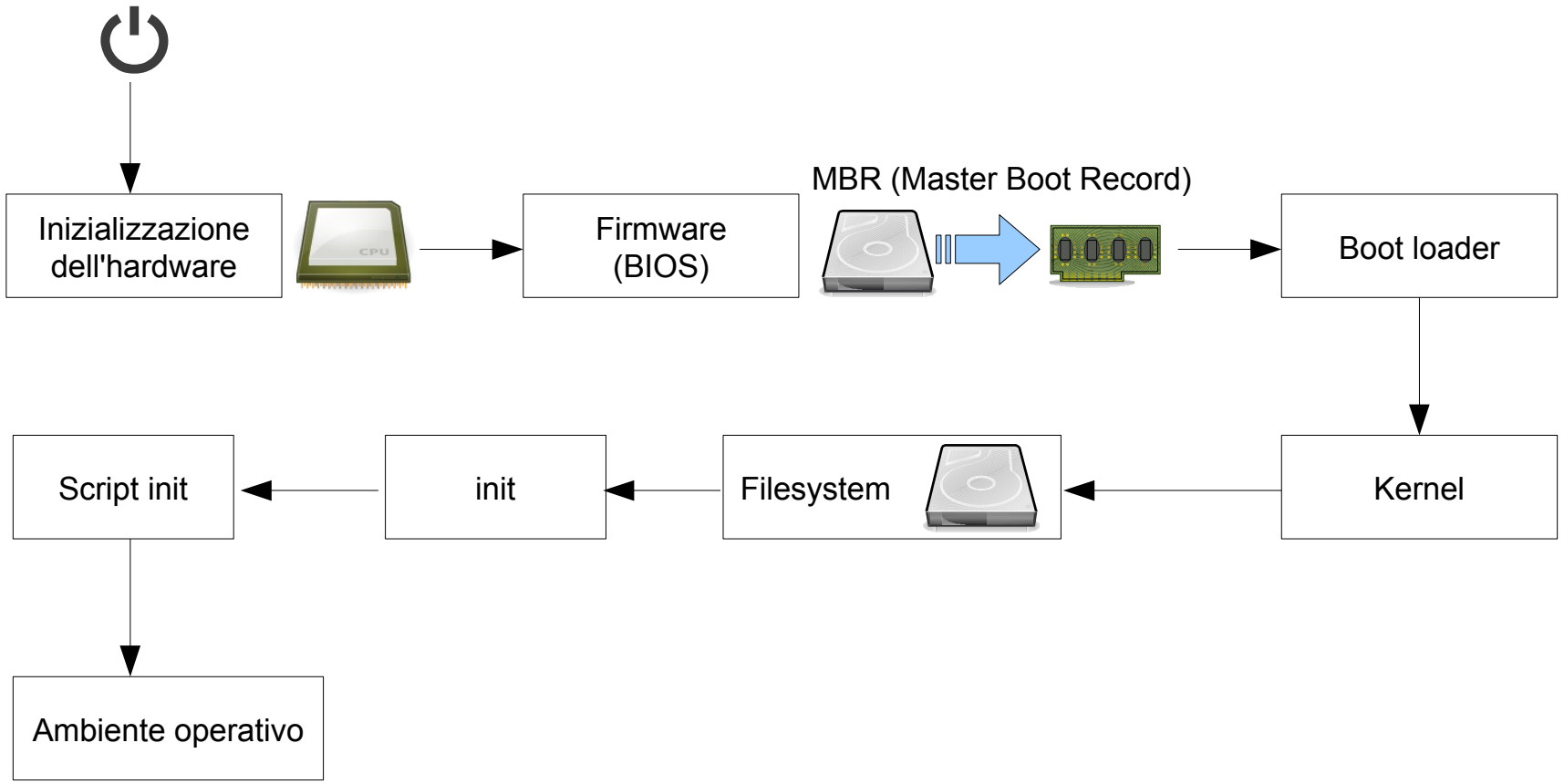
+

Complessità

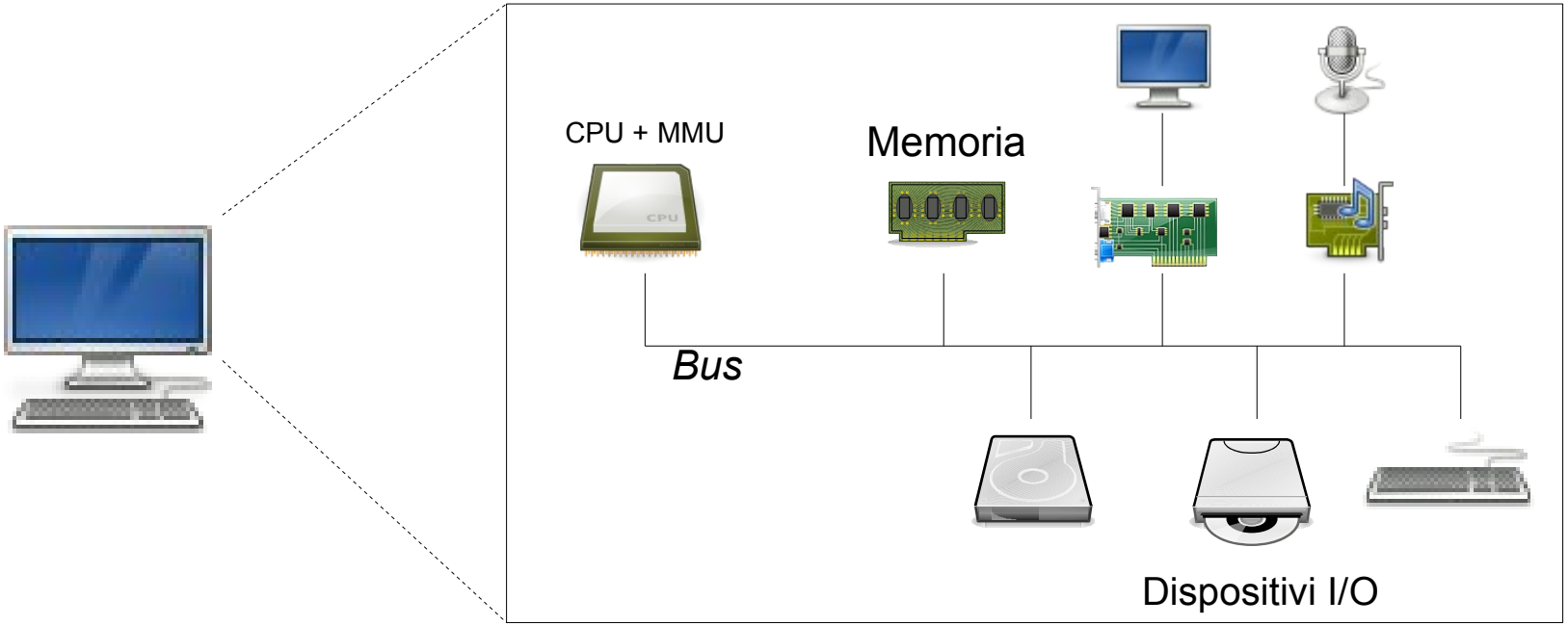


Diverse architetture
Diversi tipi di memoria, disco, ecc.
Diversi tipi di periferica
Diversi tipi di connettività
Diversi ambienti di utilizzo
Requisiti diversi

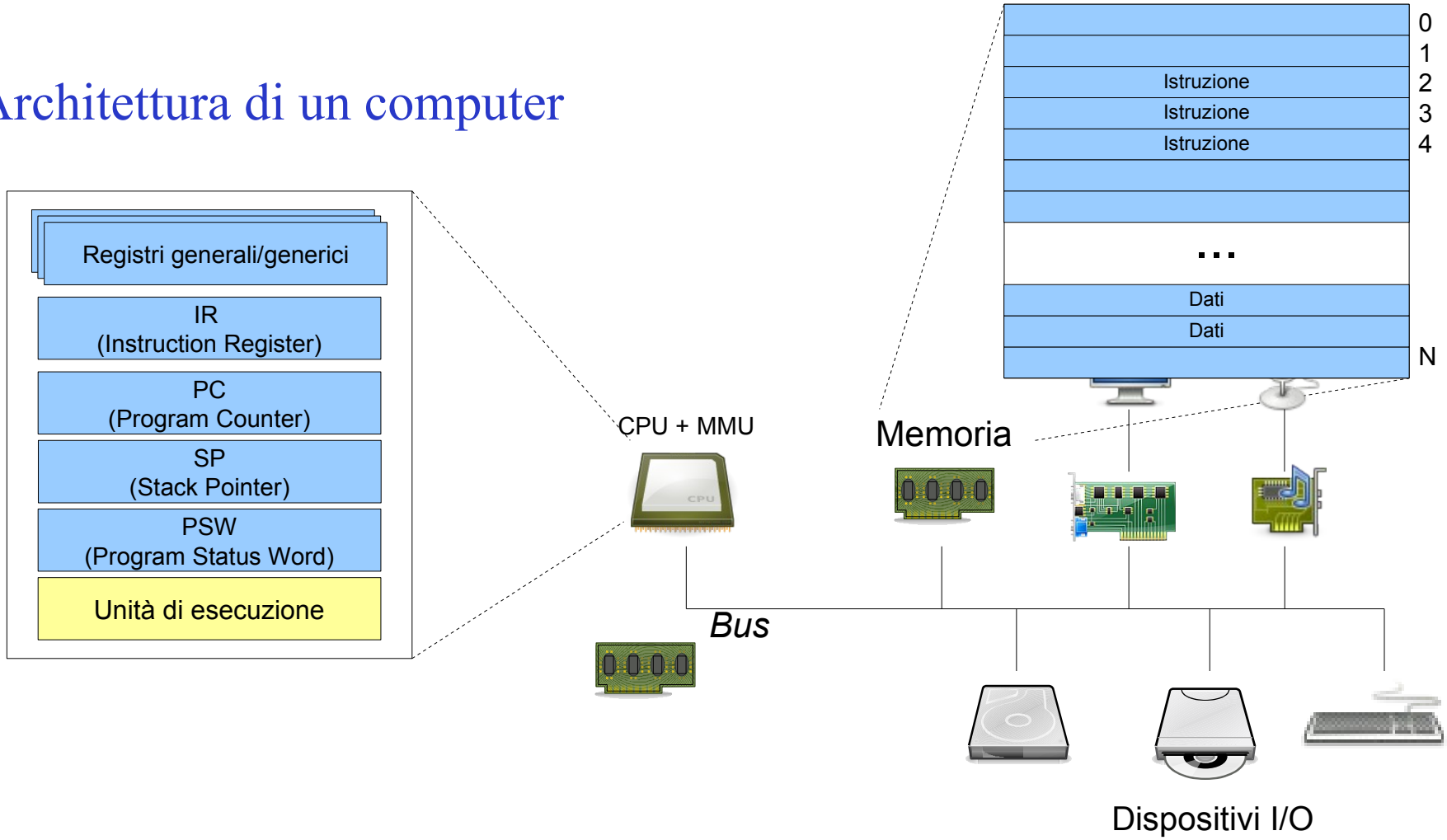
Semplifichiamo un po'.... il boot del nostro computer...



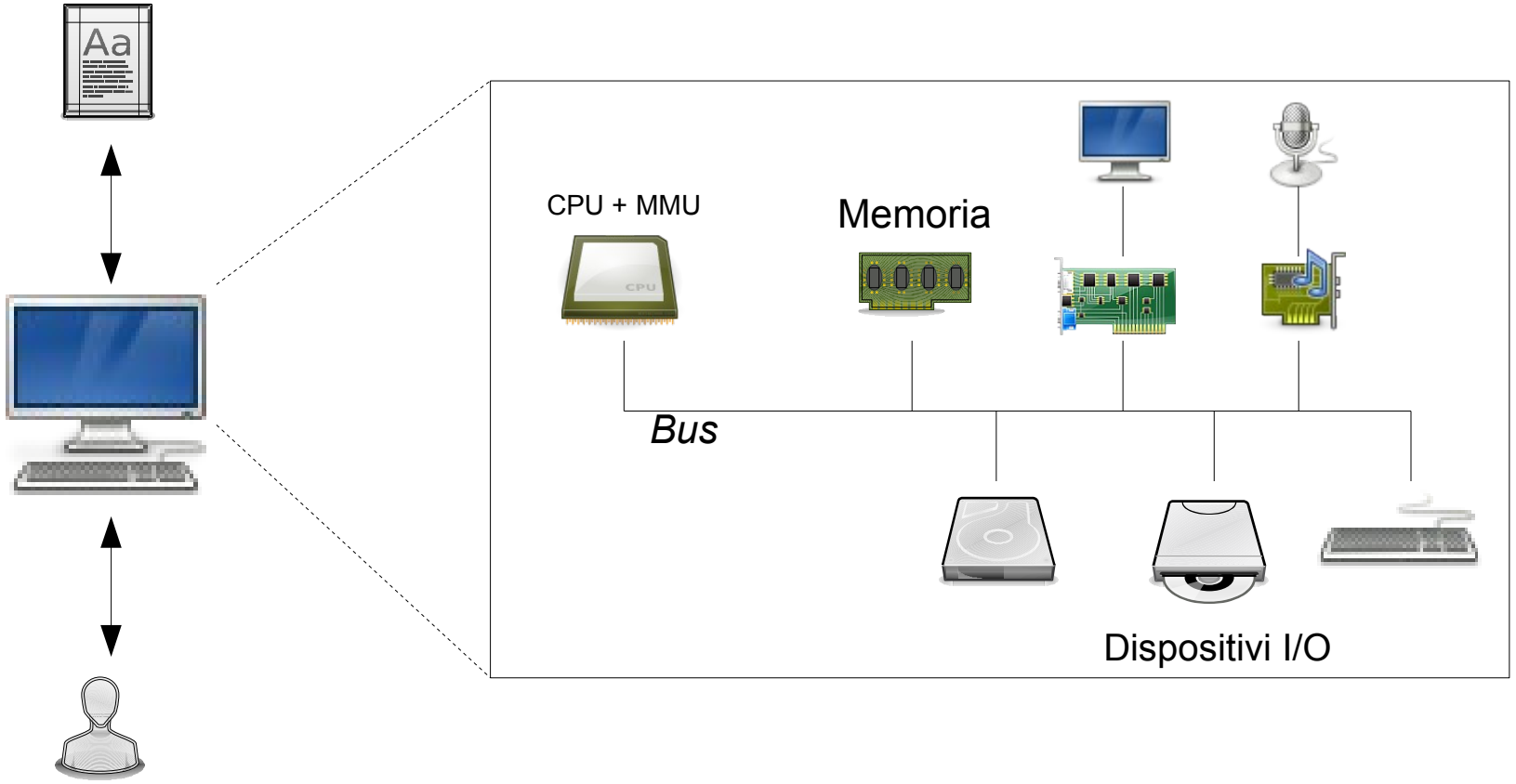
Semplifichiamo ancora un po'.... architettura di un computer



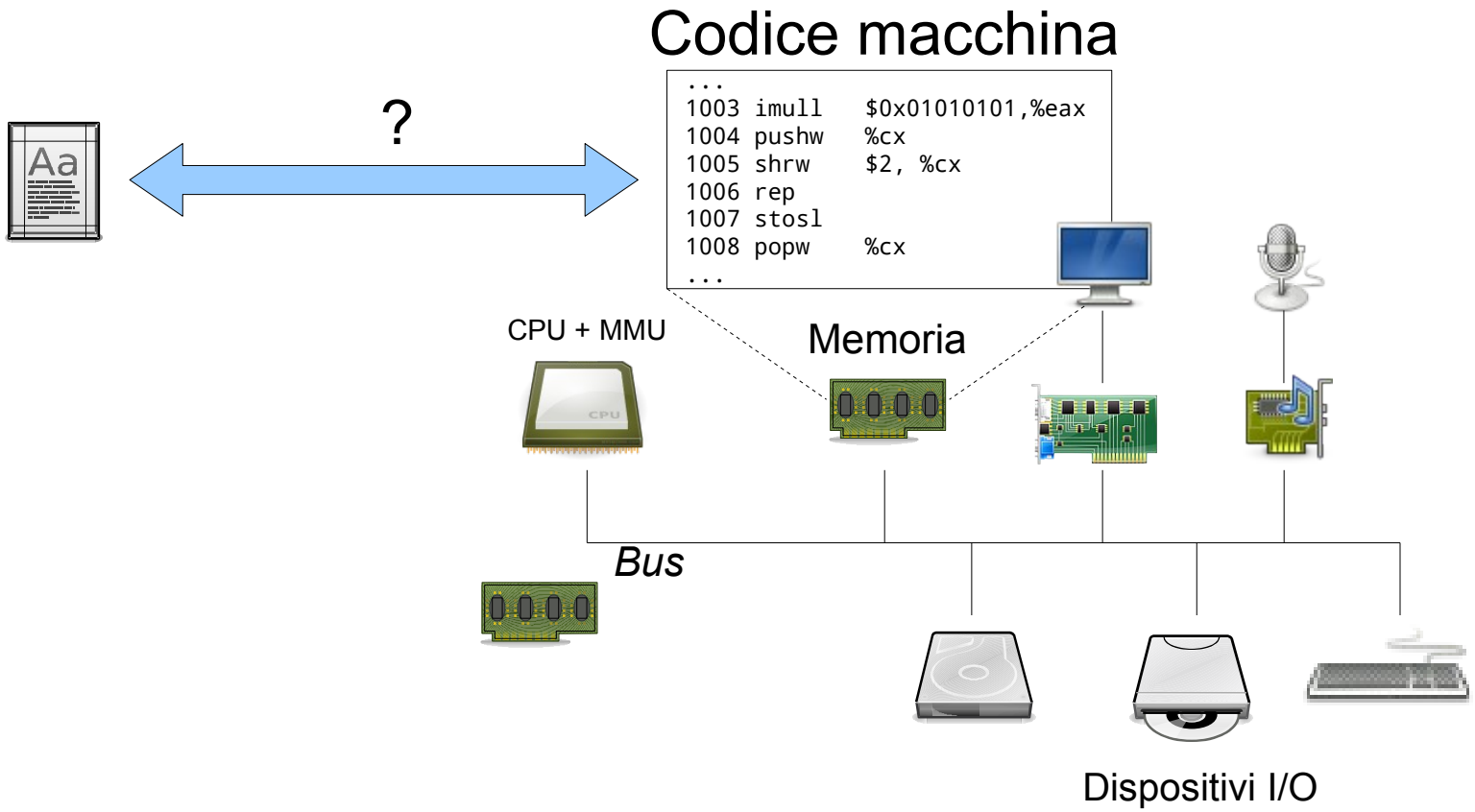
Architettura di un computer



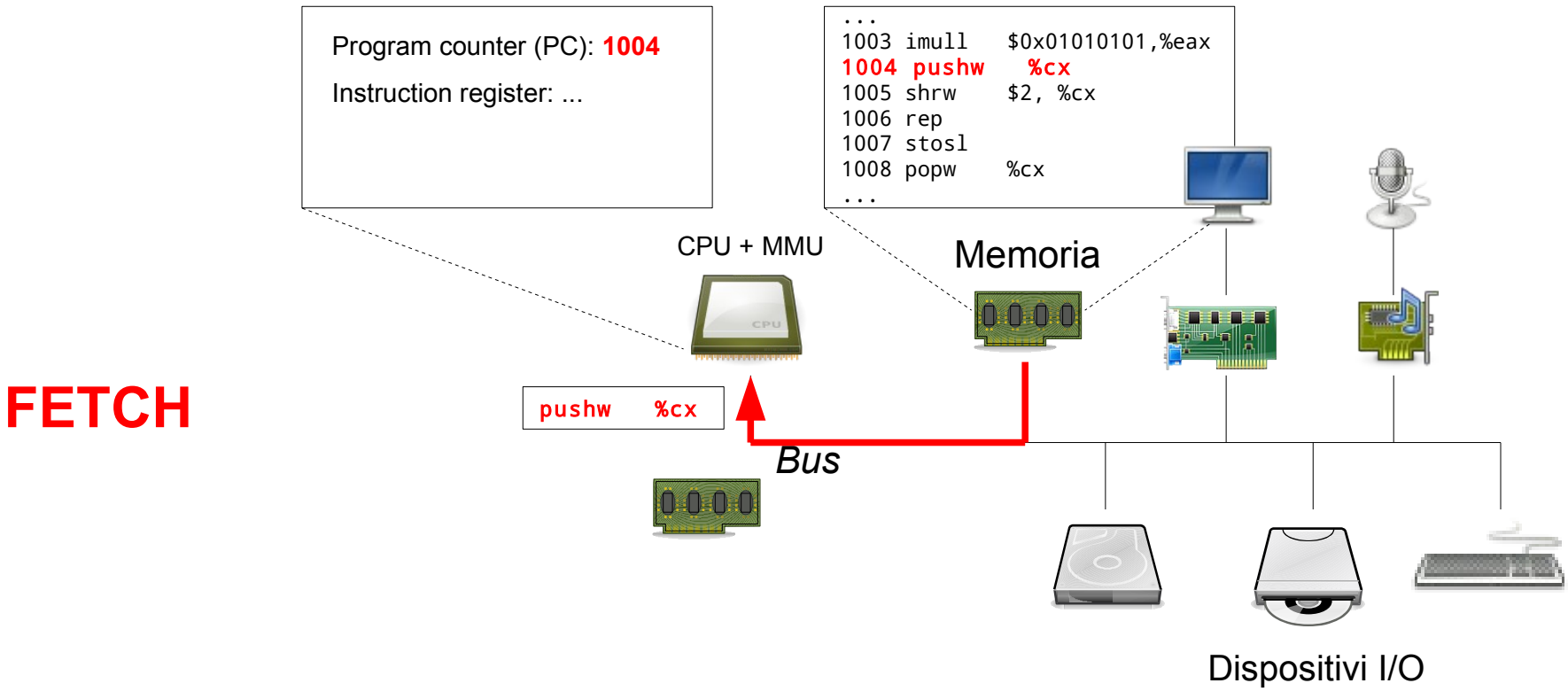
Esecuzione del software



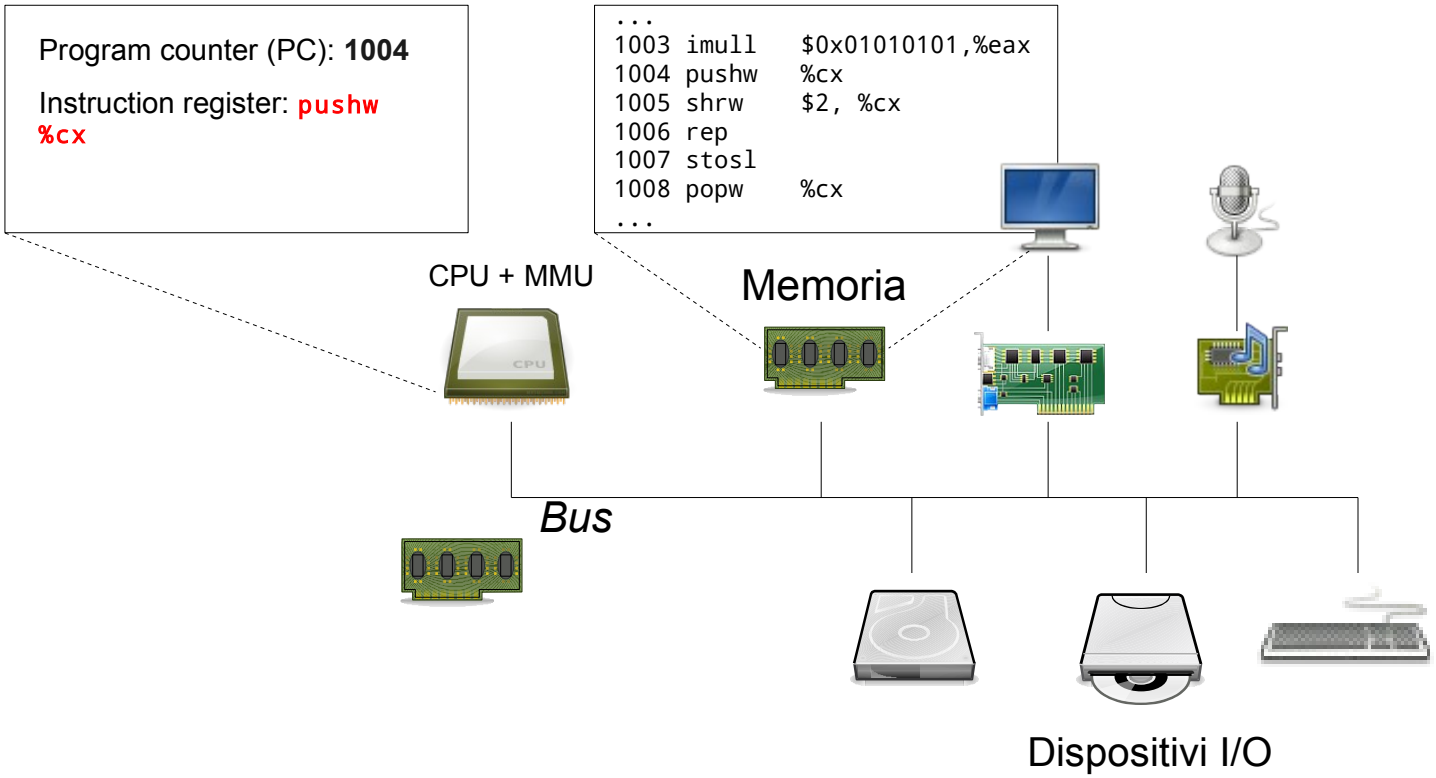
Esecuzione del software



Esecuzione del software



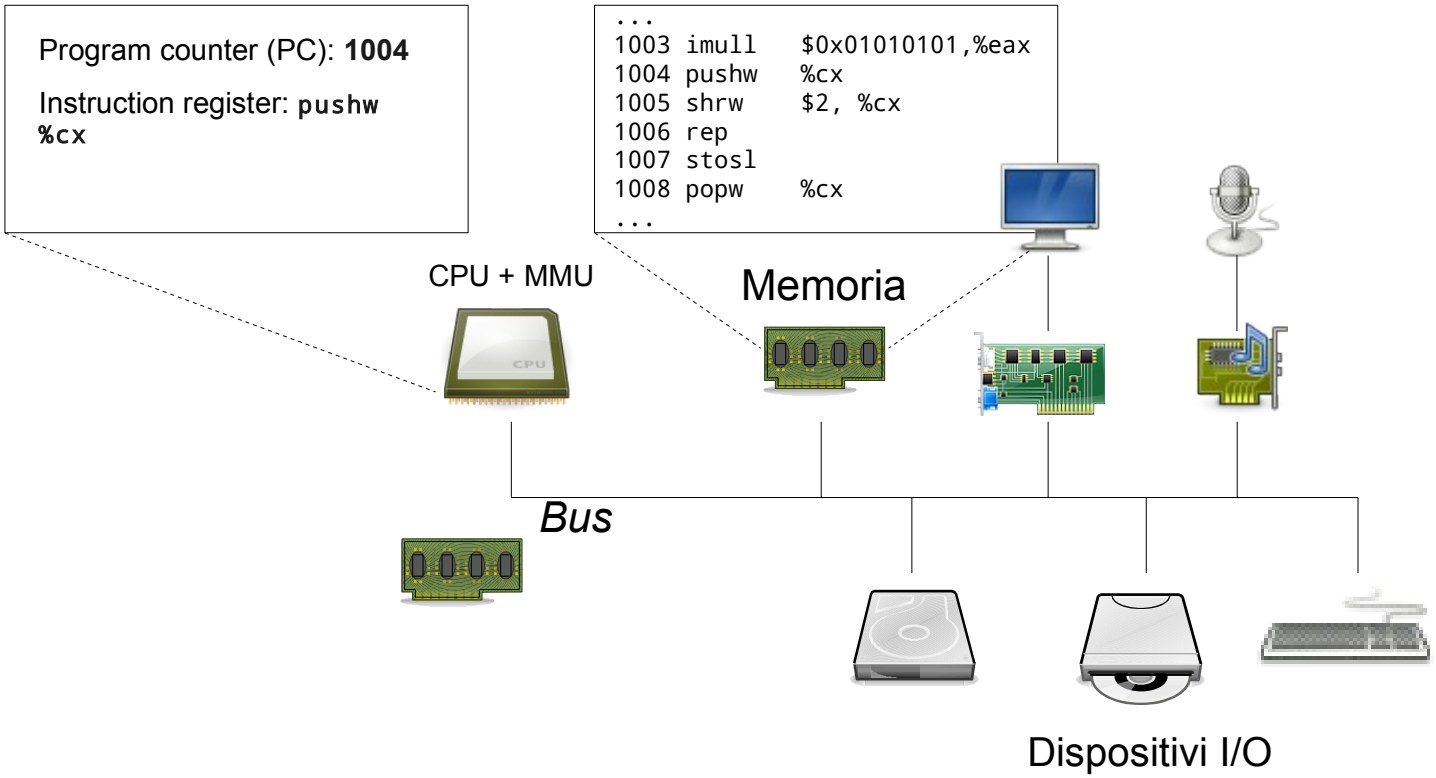
Esecuzione del software



FETCH

DECODE

Esecuzione del software

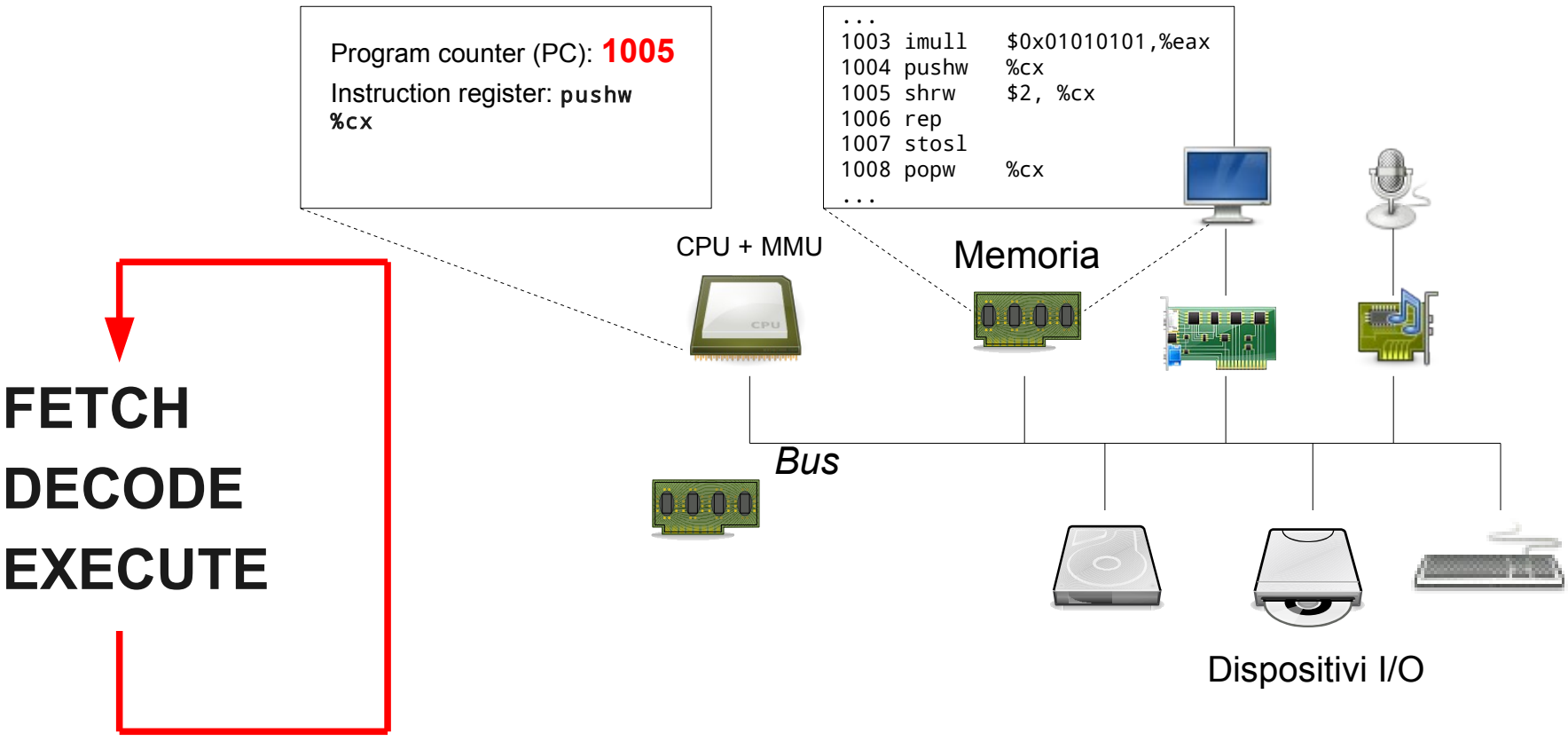


FETCH

DECODE

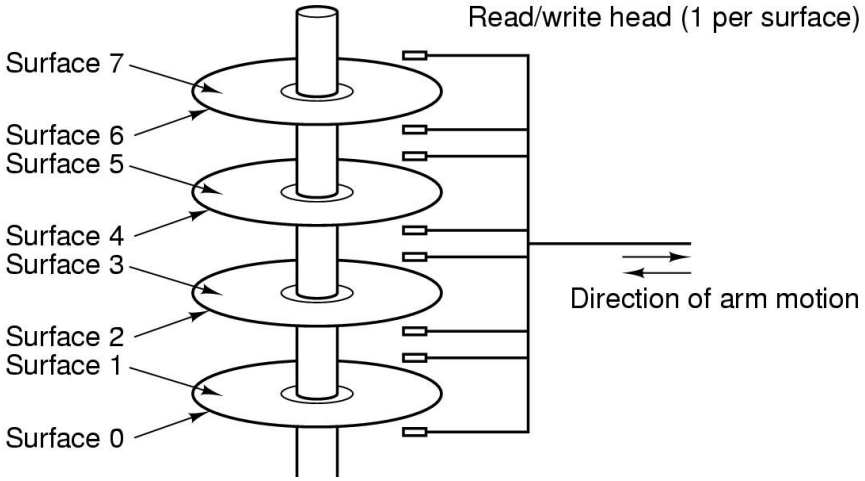
EXECUTE

Esecuzione del software



Pipeline di esecuzione

Memorie di massa



Tratto da da [TAN01]

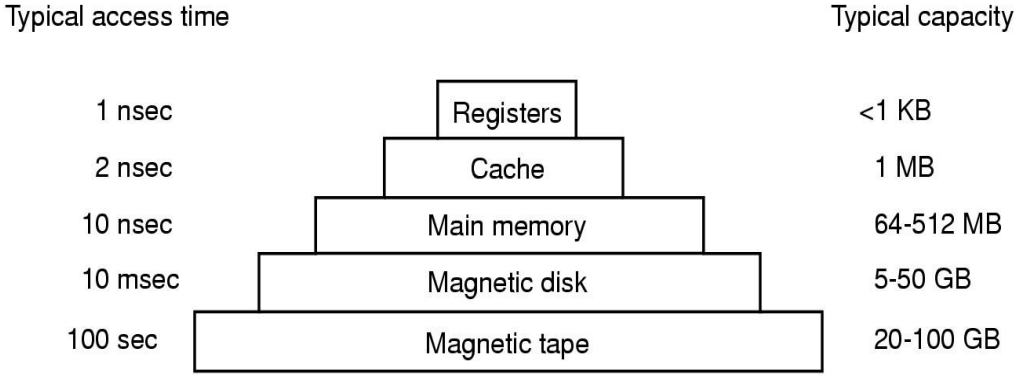
- Memorizzare nella RAM non è sufficiente:
  - Perché lo spazio a disposizione non è sufficiente
  - Per mantenere informazione a lungo termine
    - ...dopo aver terminato un'applicazione
    - ...dopo aver spento il computer
  - Per permettere la condivisione
    - ...tra processi
    - ...tra sistemi

Differenze tra i diversi tipi di memoria

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Tratto da da [STA07]

Differenze tra i diversi tipi di memoria



Tratto da da [TAN01]

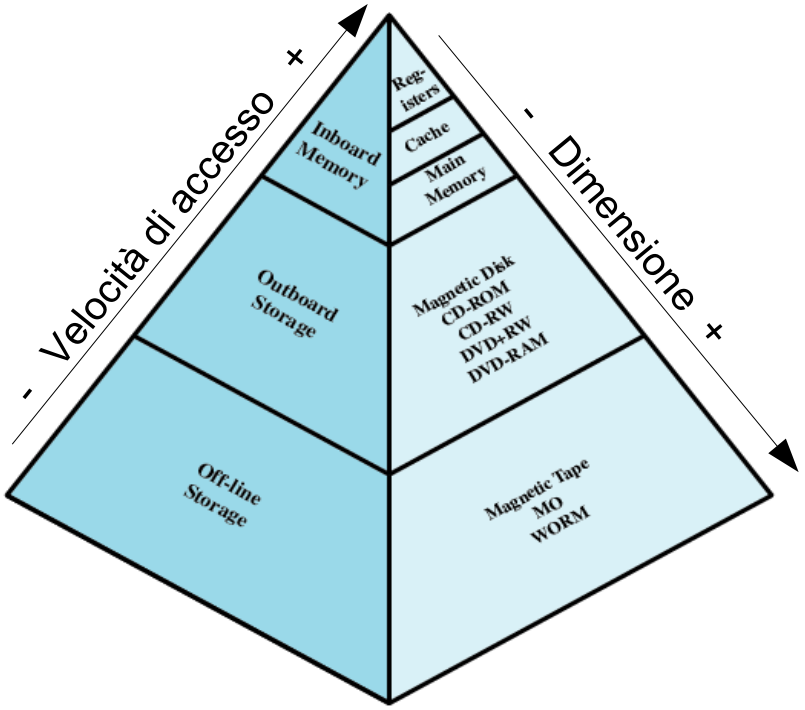


Figure 1.14 The Memory Hierarchy

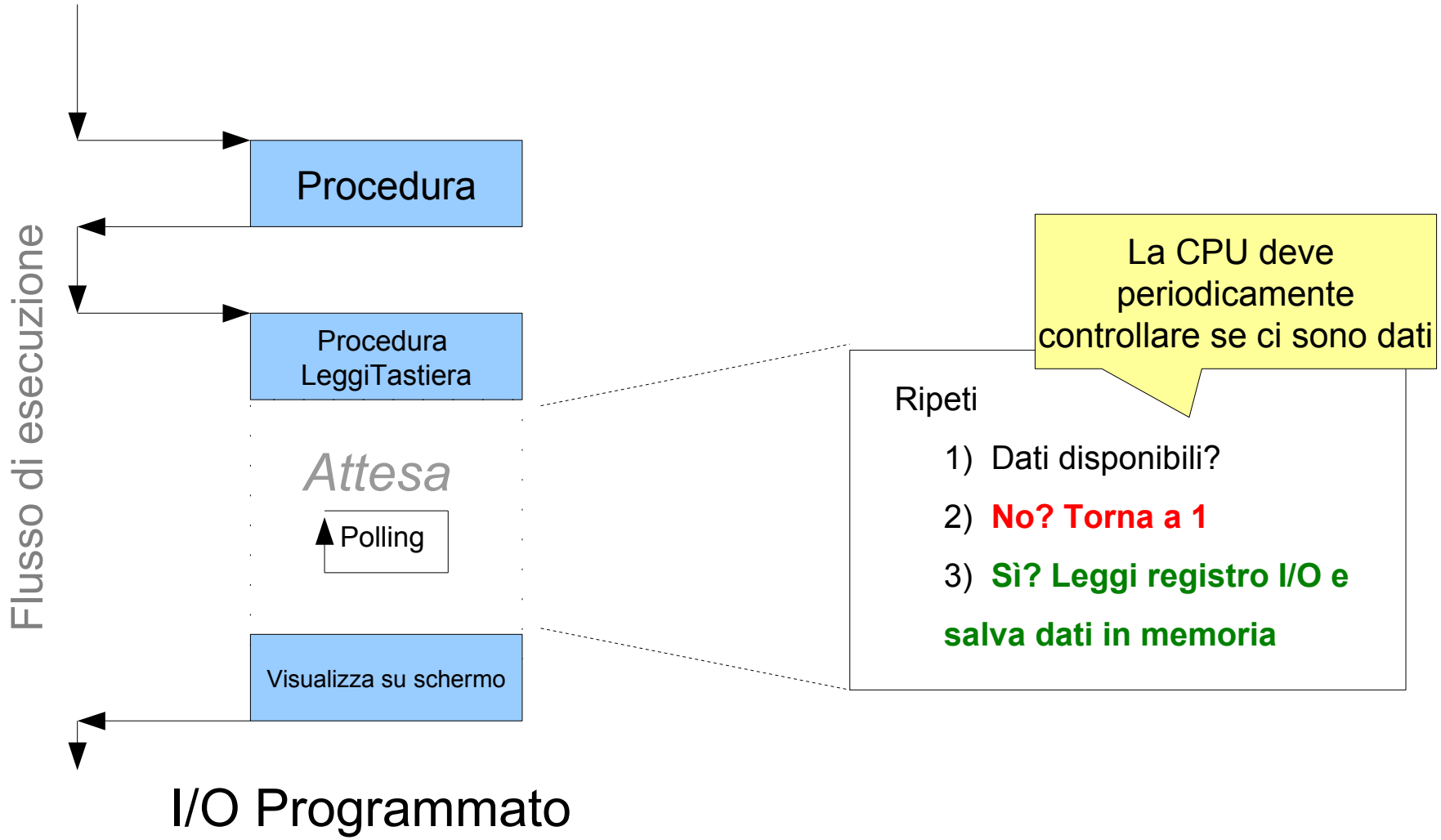
Tratto da da [STA09]

Problematiche quando si interagisce con le periferiche

- Quando la CPU deve leggere o scrivere dei dati su periferiche esterne (es. dischi, tastiera) dobbiamo aspettarci tempi di risposta più alti
  - **Maggiore latenza**
  - **Velocità di trasferimento minore**

- **Se non teniamo in considerazione questo problema l'efficienza del sistema è compromessa!**

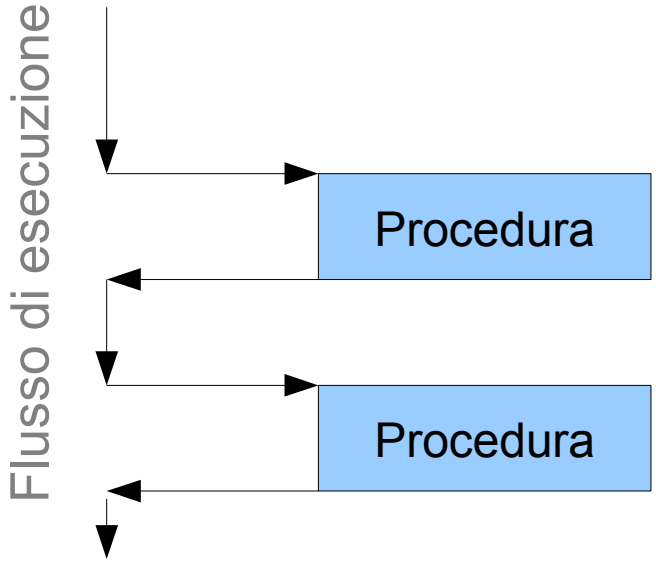
I/O Sincrono: Programmato/Polling



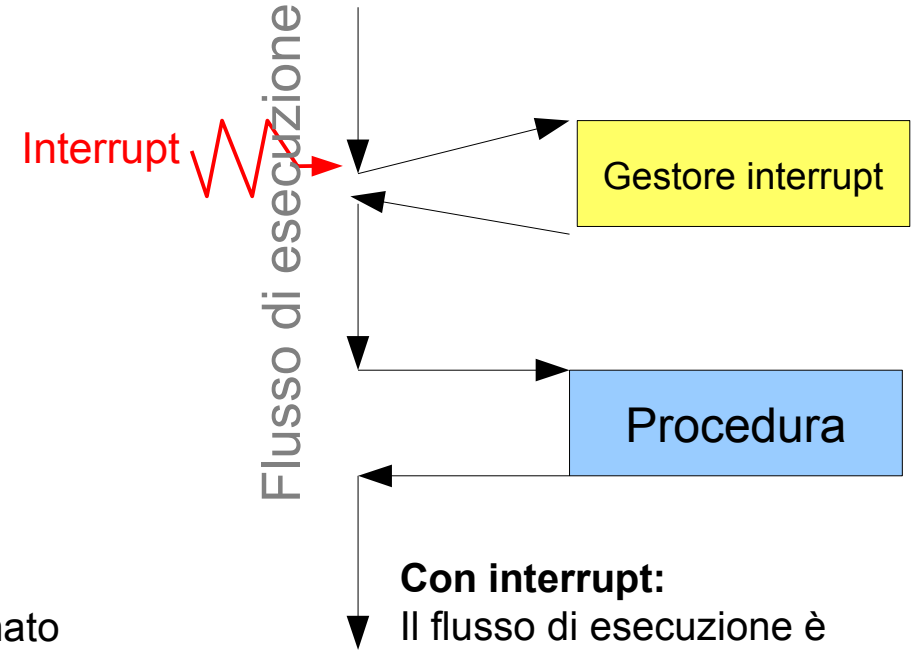
I/O Programmato

Interrupt

- Meccanismo che permette di **interrompere** il ciclo di esecuzione normale della CPU in modo **asincrono** per segnalare un evento
  - tipicamente utilizzato dalle periferiche hardware, può essere anche generato in modo **sincrono** dai programmi (**trap/eccezione**)



Senza interrupt:
Il flusso di esecuzione è determinato dalla sequenza di istruzioni

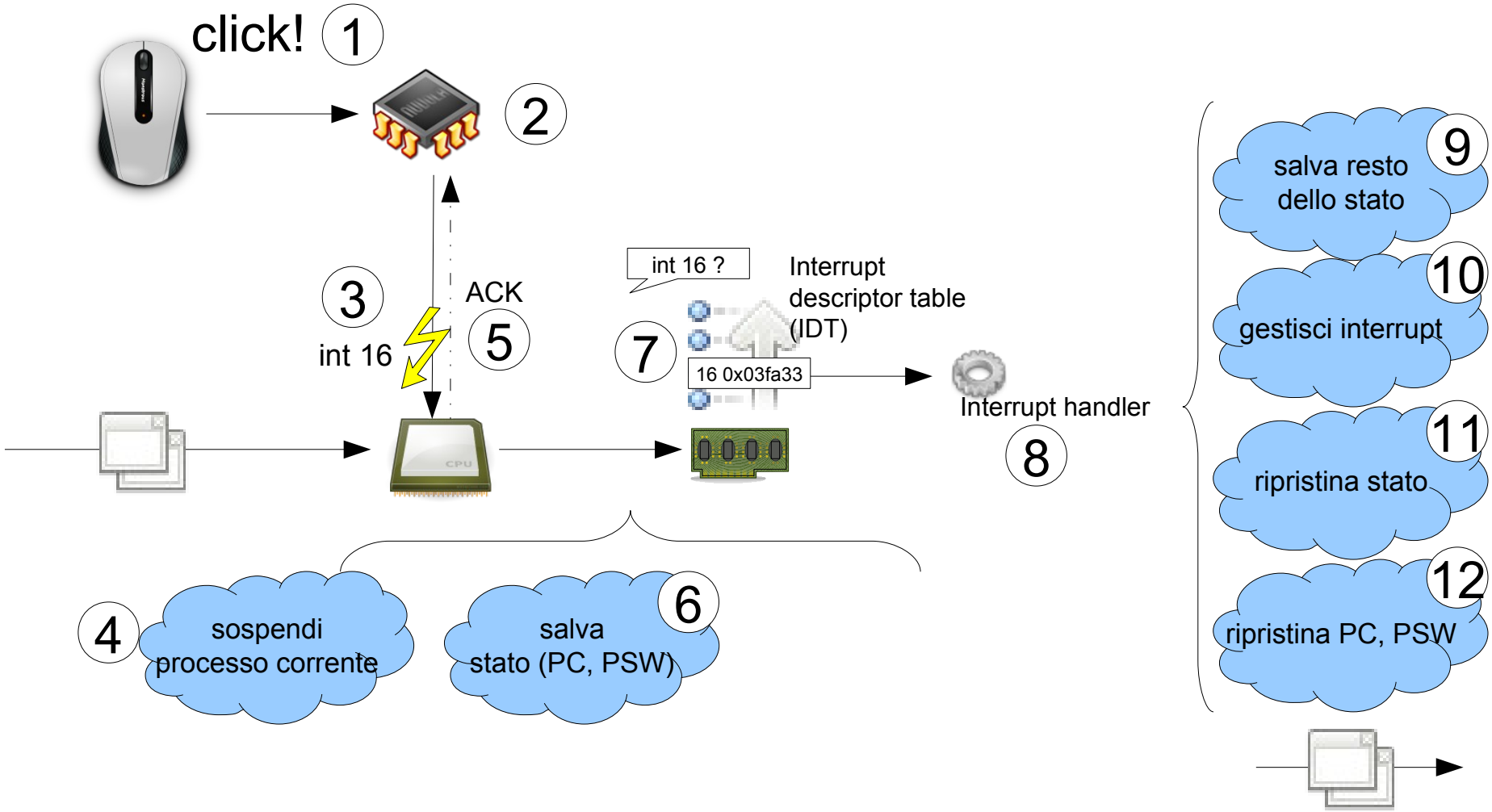


Con interrupt:
Il flusso di esecuzione è modificato da un evento esterno (interruzione)

Esempi di interrupt

- Interrupt software
- Timer
- Interrupt di periferiche I/O
- Problemi hardware

Gestione degli interrupt hardware

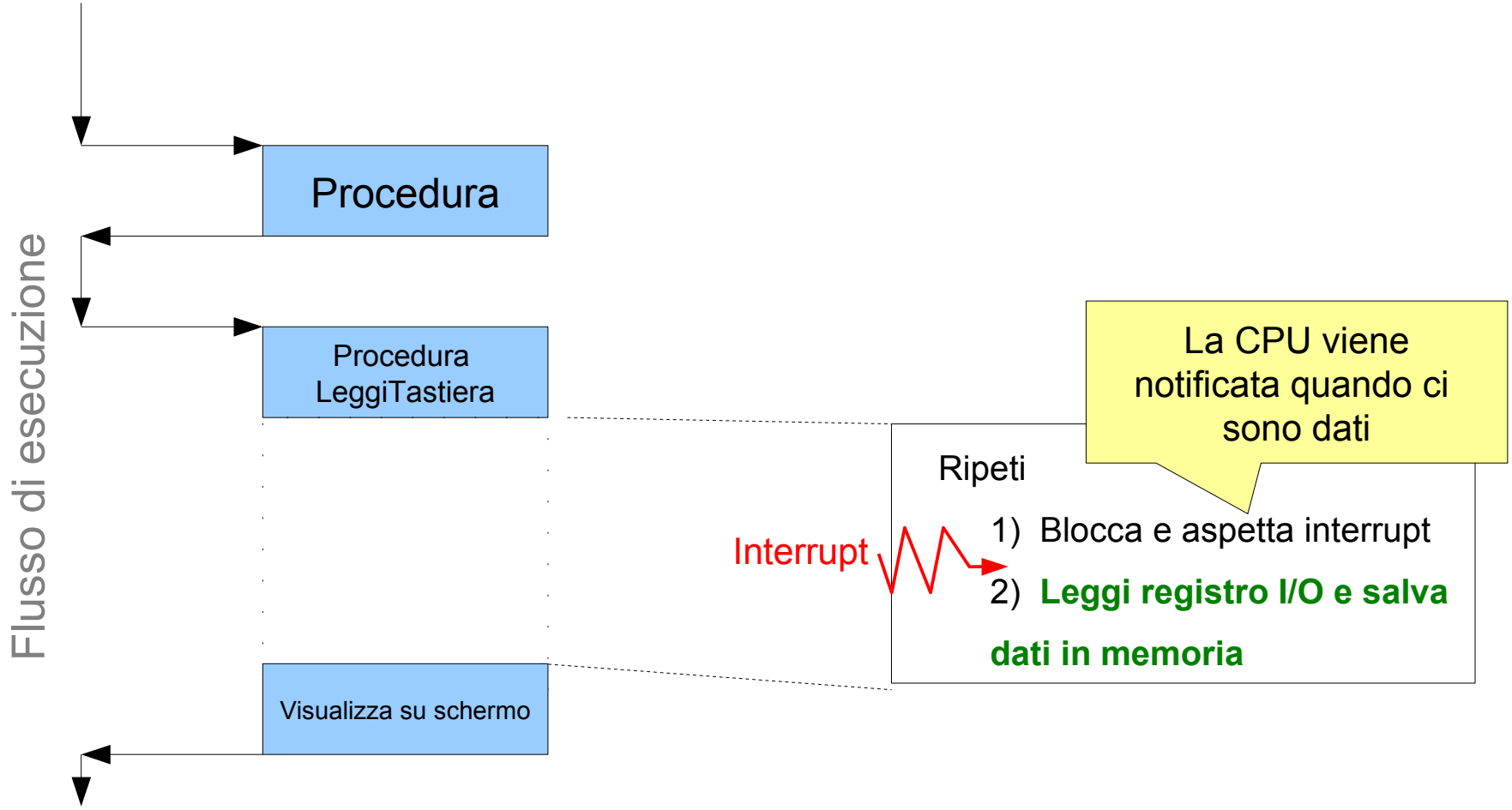


Interrupt multipli

- Se più interrupt vengono generati contemporaneamente, a quale rispondere per primo?
  - priorità per ogni interrupt

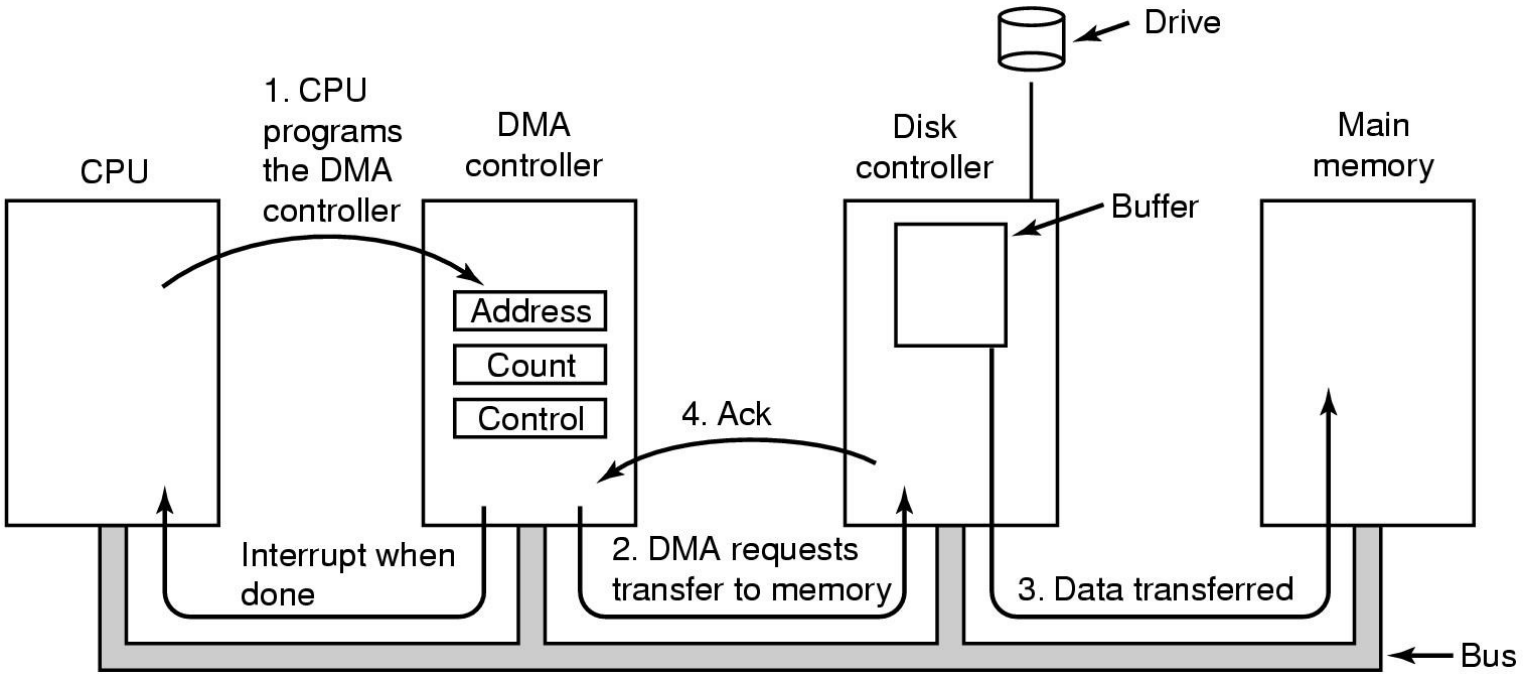
- Se durante l'esecuzione di un gestore di interrupt, un altro interrupt viene generato cosa deve fare il sistema?
  - disabilito interrupt quando eseguo gestore

I/O Sincrono: Interrupt



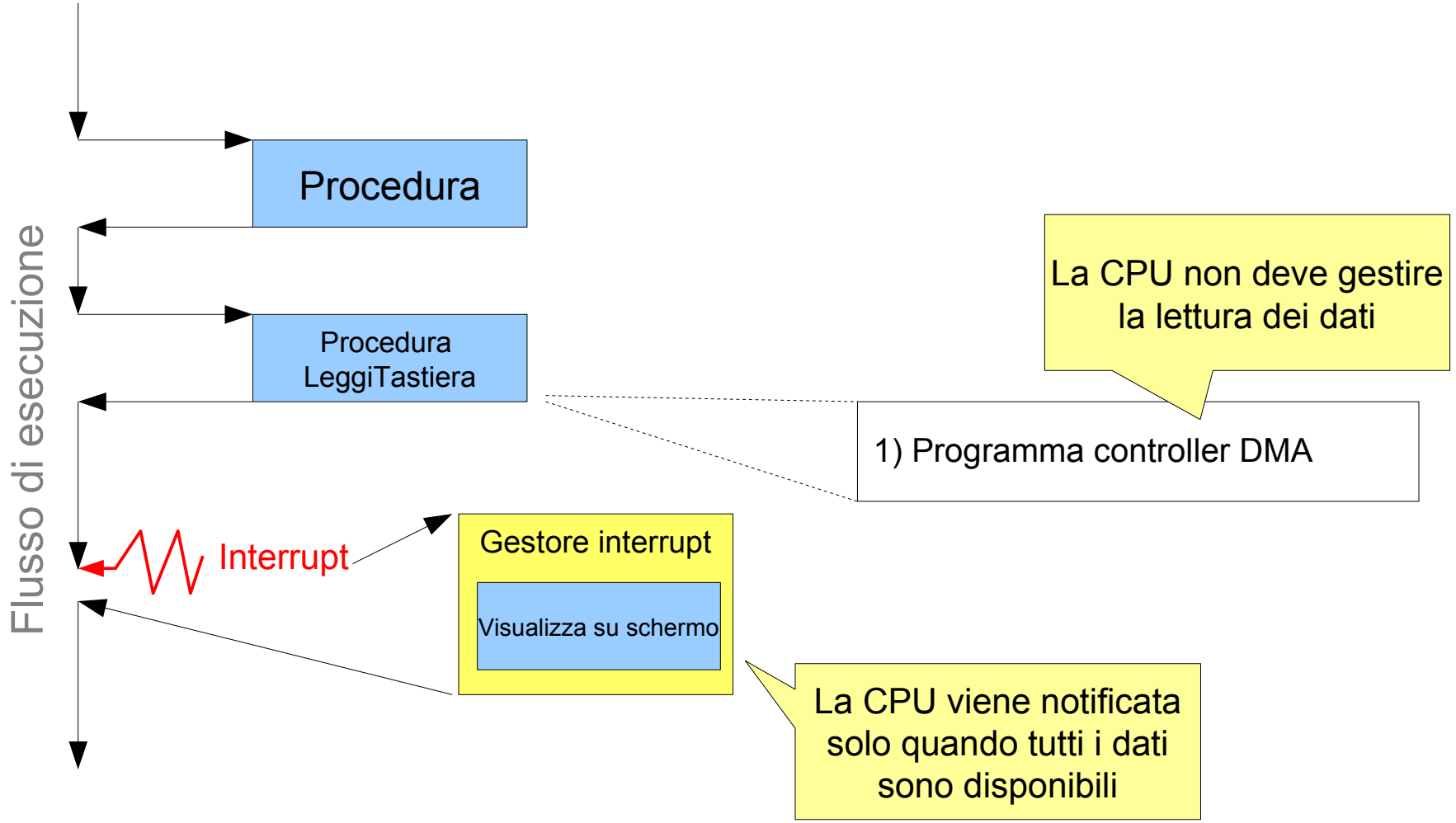
I/O Asincrono: DMA (Direct Memory Access)

- Per alleviare la CPU dal compito di gestire i trasferimenti di dati in input / output da e per la memoria, la maggior parte dei computer implementa un meccanismo aggiuntivo: il **Direct Memory Access**



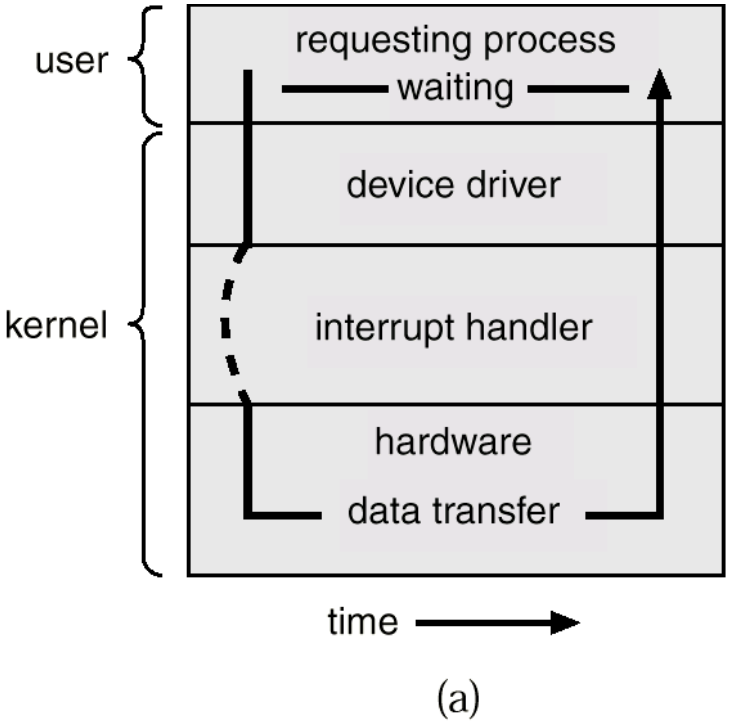
Tratto da da [TAN01]

DMA (Direct Memory Access)

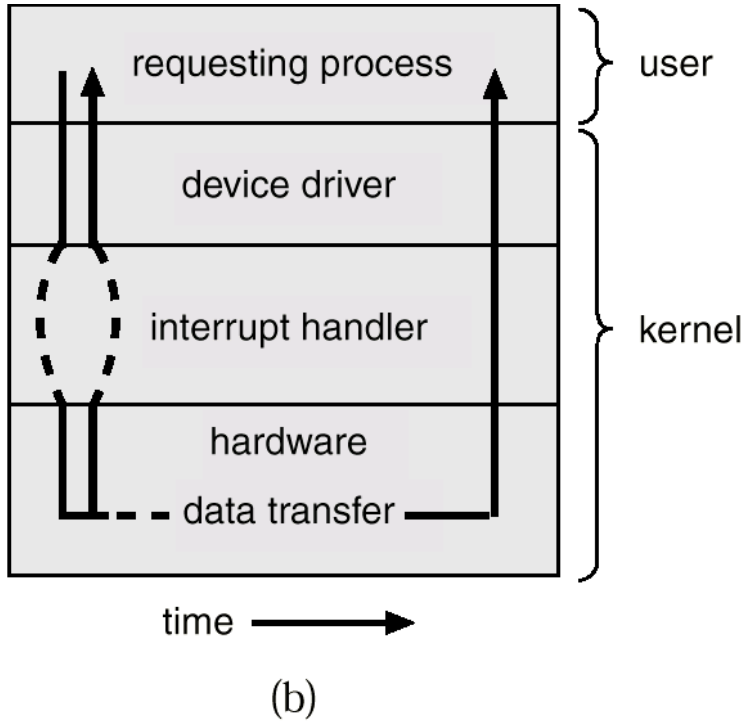


I/O Sincrono vs Asincrono

Sincrono (Polling, Interrupt)

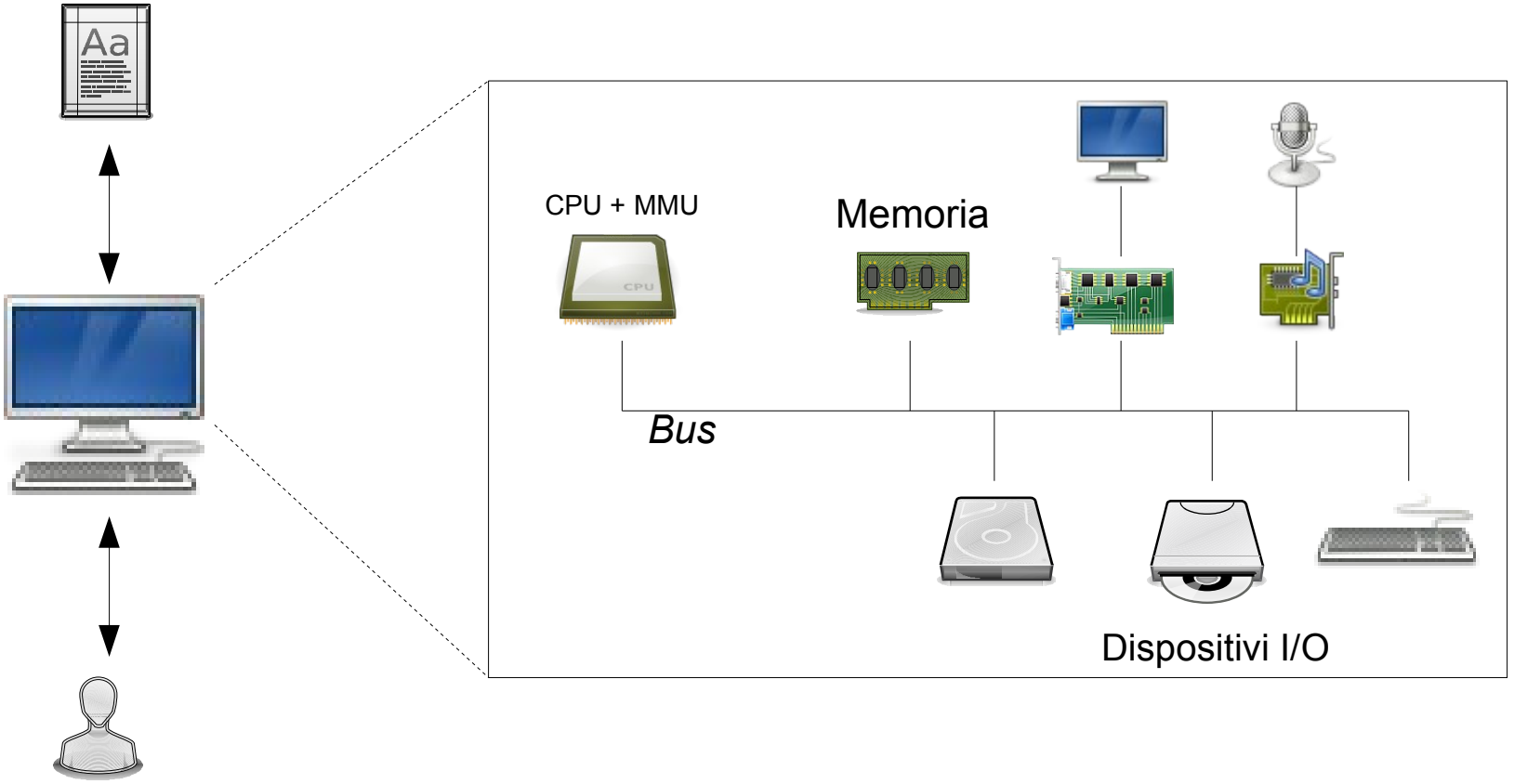


Asincrono (DMA)

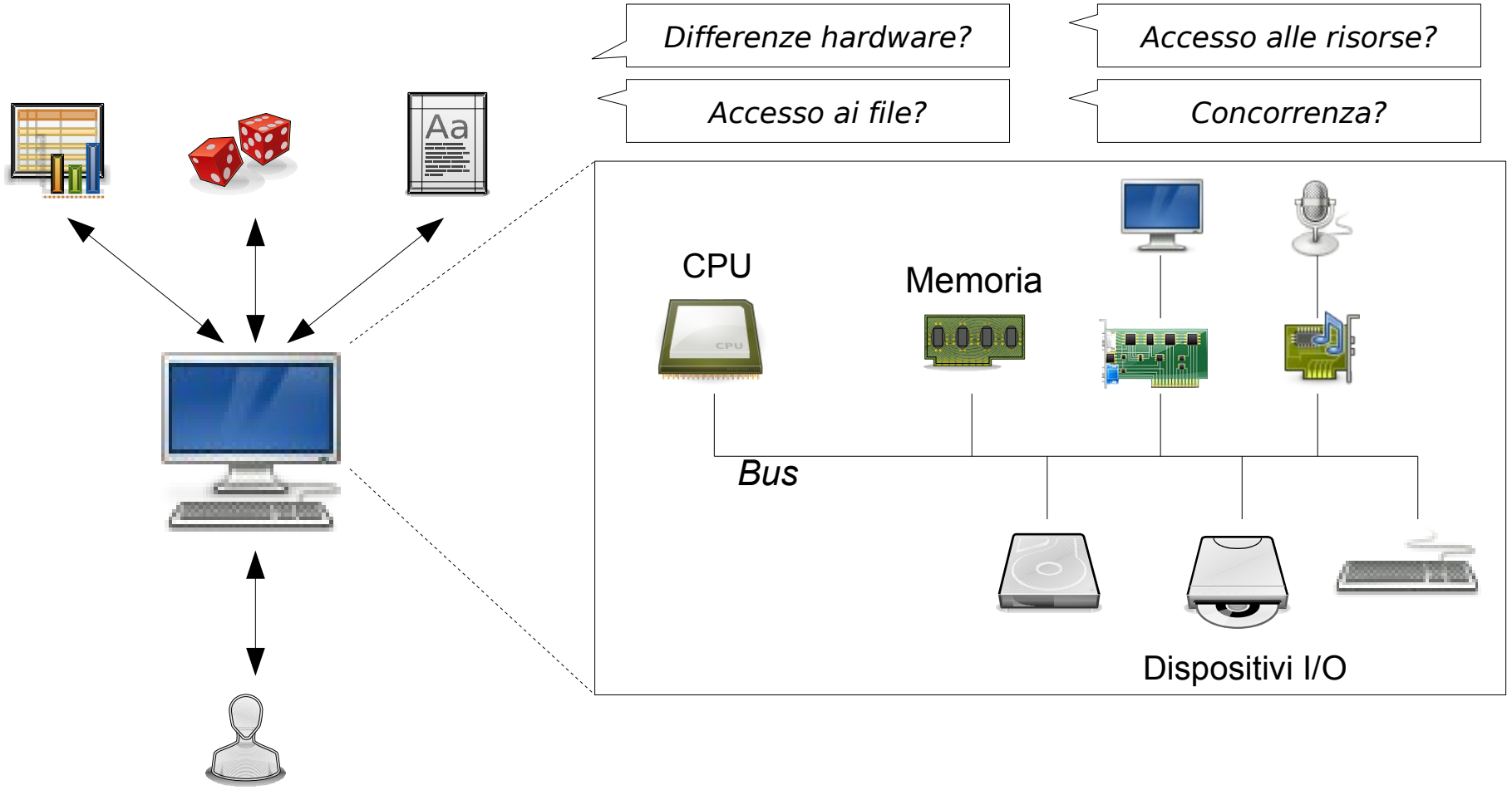


Tratto da da [SIL07]

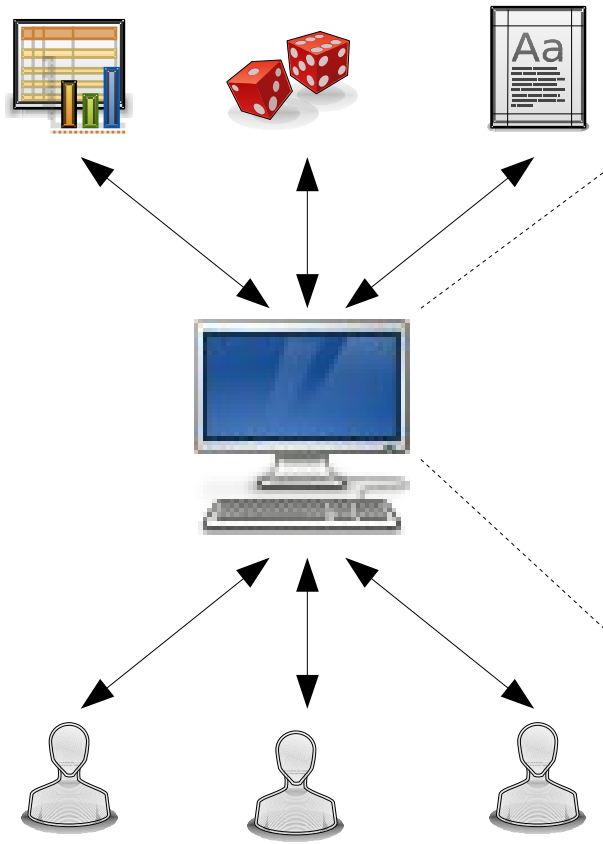
Un sistema complesso



Un sistema complesso, multiprogrammato



Un sistema complesso, multiprogrammato, multiutente

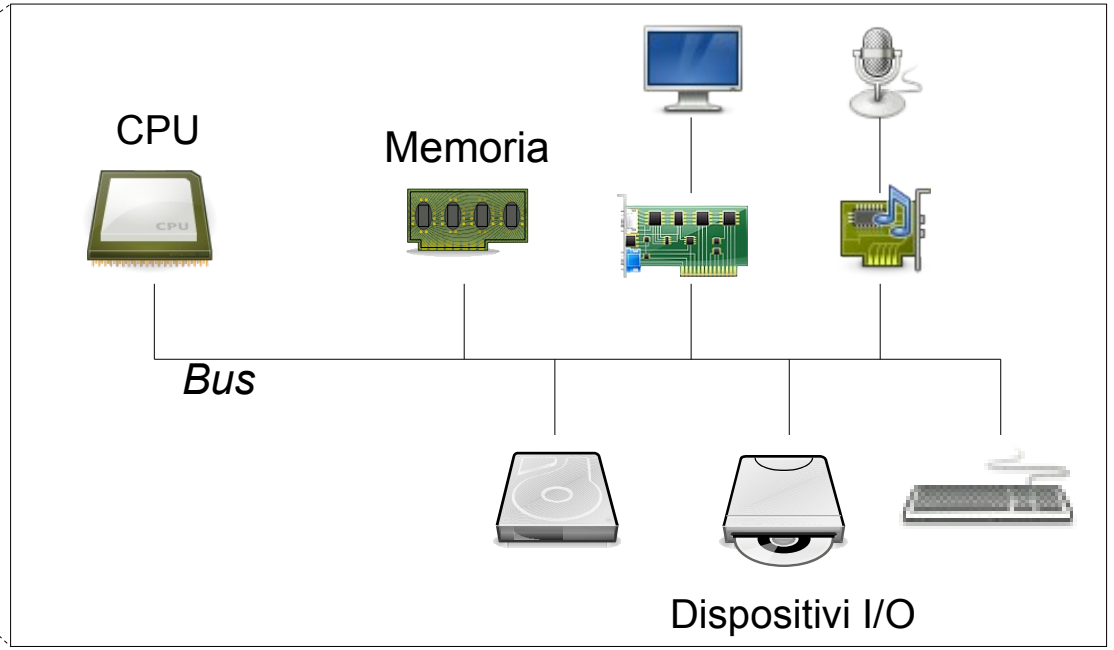


Differenze hardware?

Accesso alle risorse?

Accesso ai file?

Concorrenza?



Organizzazione dei dati?

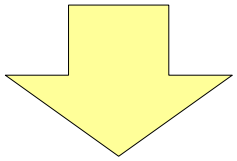
Utilizzo delle periferiche?

Esecuzione dei programmi?

Sicurezza?

Semplificare la programmazione, gestire le risorse

- **Cosa vogliamo:**
  - **Astrazione (macchina estesa) – *Visto dal basso verso l'alto***
    - un livello di astrazione che nasconde i meccanismi di gestione e le limitazioni del calcolatore
    - più facile da programmare
  - **Un gestore di risorse – *Visto dall'alto verso il basso***
    - per far funzionare insieme tutte le risorse e fornisce una visione globale e coerente del sistema
    - per eseguire più programmi contemporaneamente
    - per proteggere la memoria, le periferiche e le altre risorse



Sistema operativo

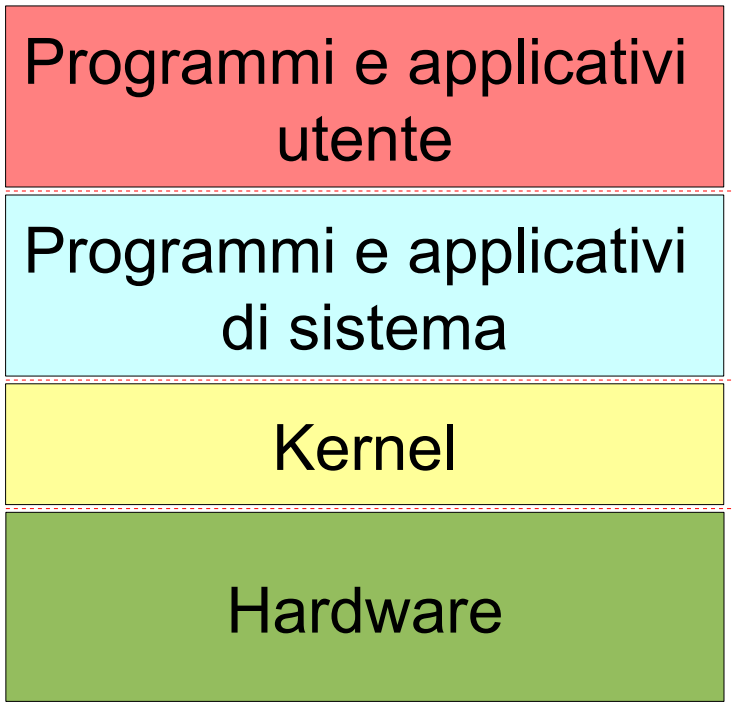
Definizione

- Il software può essere diviso in due grandi classi:
  - i **programmi di sistema**, che gestiscono le funzionalità del sistema di calcolo
  - i **programmi applicativi**, che risolvono i problemi degli utenti

- L'insieme dei programmi di sistema viene comunemente identificato con il nome di Sistema Operativo (SO)

- *Definizione*
  - *Un sistema operativo è un programma che controlla l'esecuzione di programmi applicativi ed agisce come interfaccia fra le applicazioni e l'hardware del calcolatore*

Astrazione



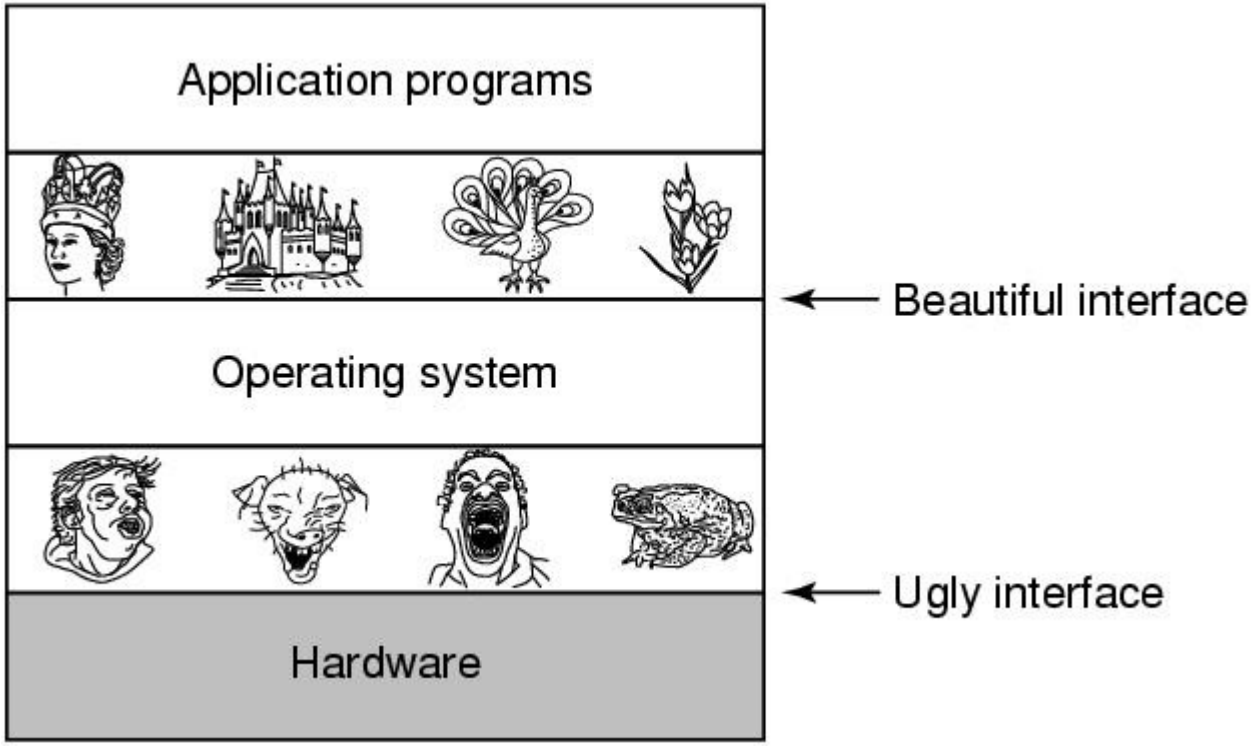
Risolvono problemi degli utenti

Definiscono il modo in cui le risorse di calcolo sono utilizzate per risolvere i problemi degli utenti

Controlla e coordina l'uso dell'hardware tra applicazioni e utenti attivi sul sistema, implementa le astrazioni principali

Fornisce le risorse di calcolo e di memorizzazione (CPU, memoria, periferiche I/O)

Astrazione

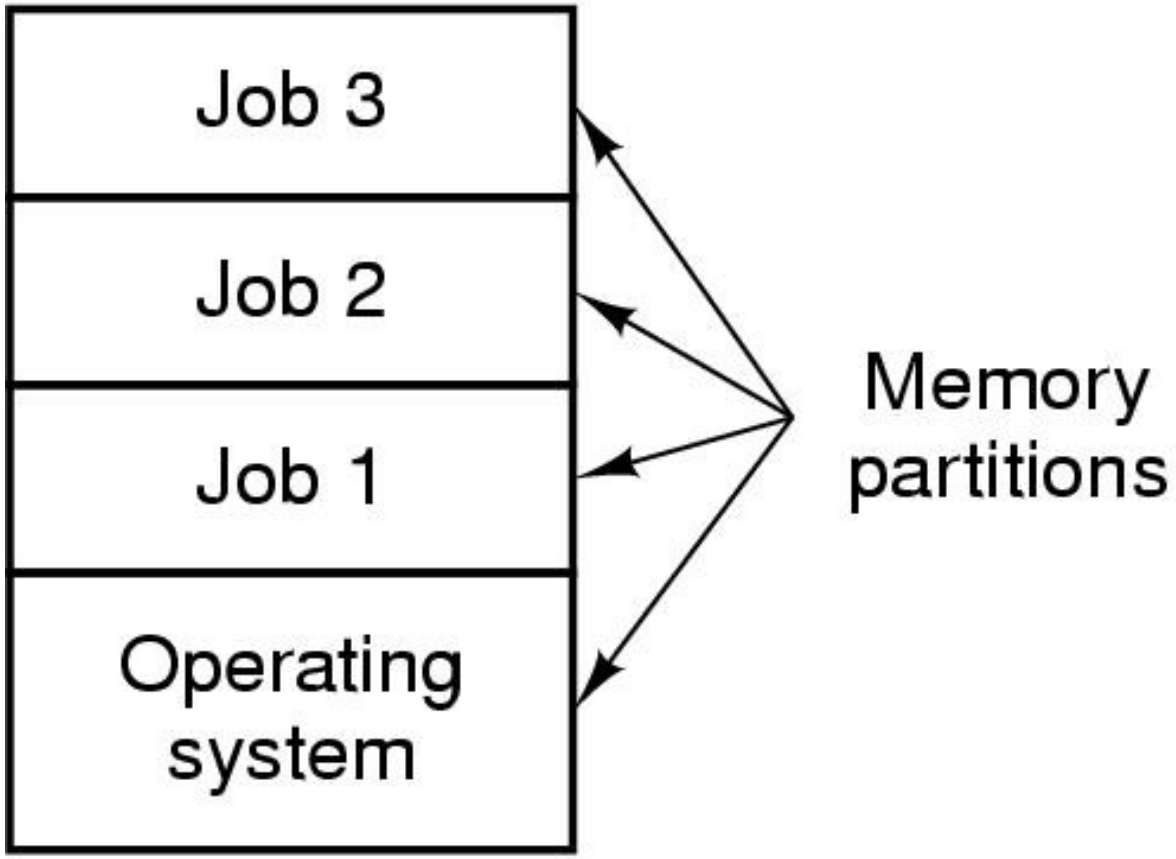


Tratto da da [TAN09]

Kernel

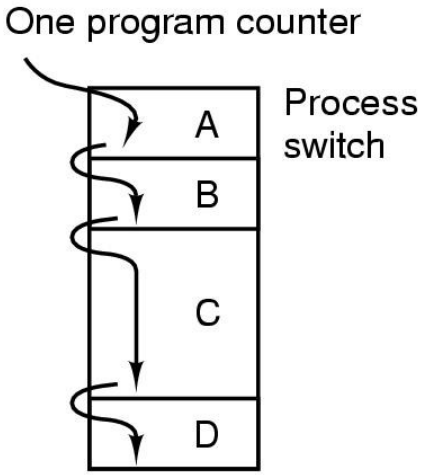
- Componente centrale del sistema operativo
  - Si occupa di coordinare le funzioni principali (è il cuore del sistema)
  - Implementa e mette a disposizione le principali astrazioni:
    - **Processi e thread**
    - **Memoria virtuale**
    - **File e directory**

Multiprogrammazione



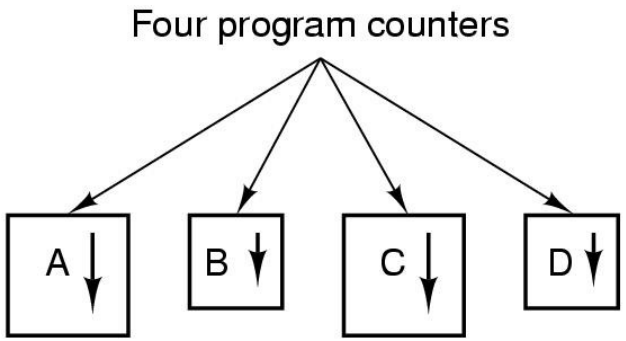
Tratto da da [TAN01]

Processi e Timesharing



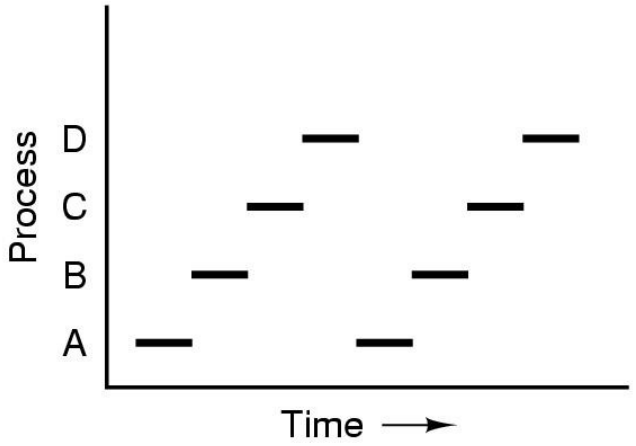
(a)

Un processo in esecuzione



(b)

Più processi in esecuzione parallela

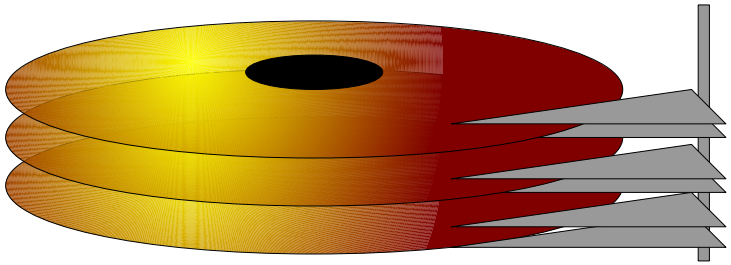


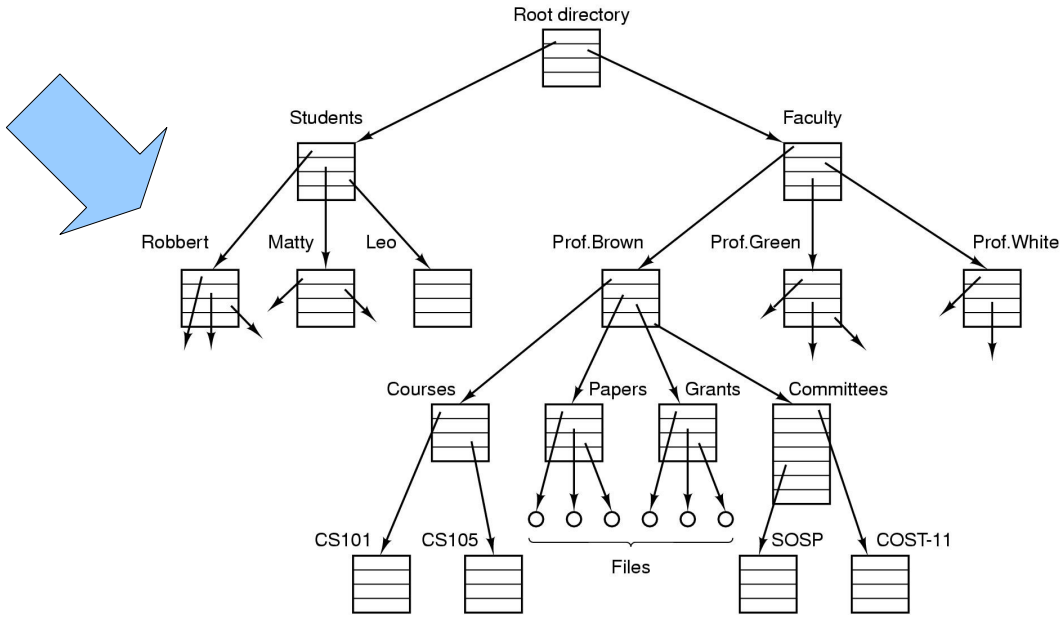
(c)

Più processi in esecuzione pseudo-parallela

Tratto da da [TAN01]

File e directory

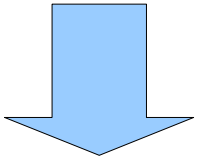




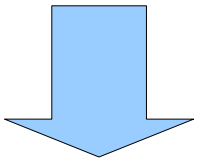
Tratto da da [TAN01]

Isolazione

- Il sistema operativo deve fare in modo che i processi non possano interferire tra di loro
  - Evitare che un processo possa scrivere/modificare i dati di un altro processo
  - Evitare che un processo possa curiosare nei dati di un altro processo

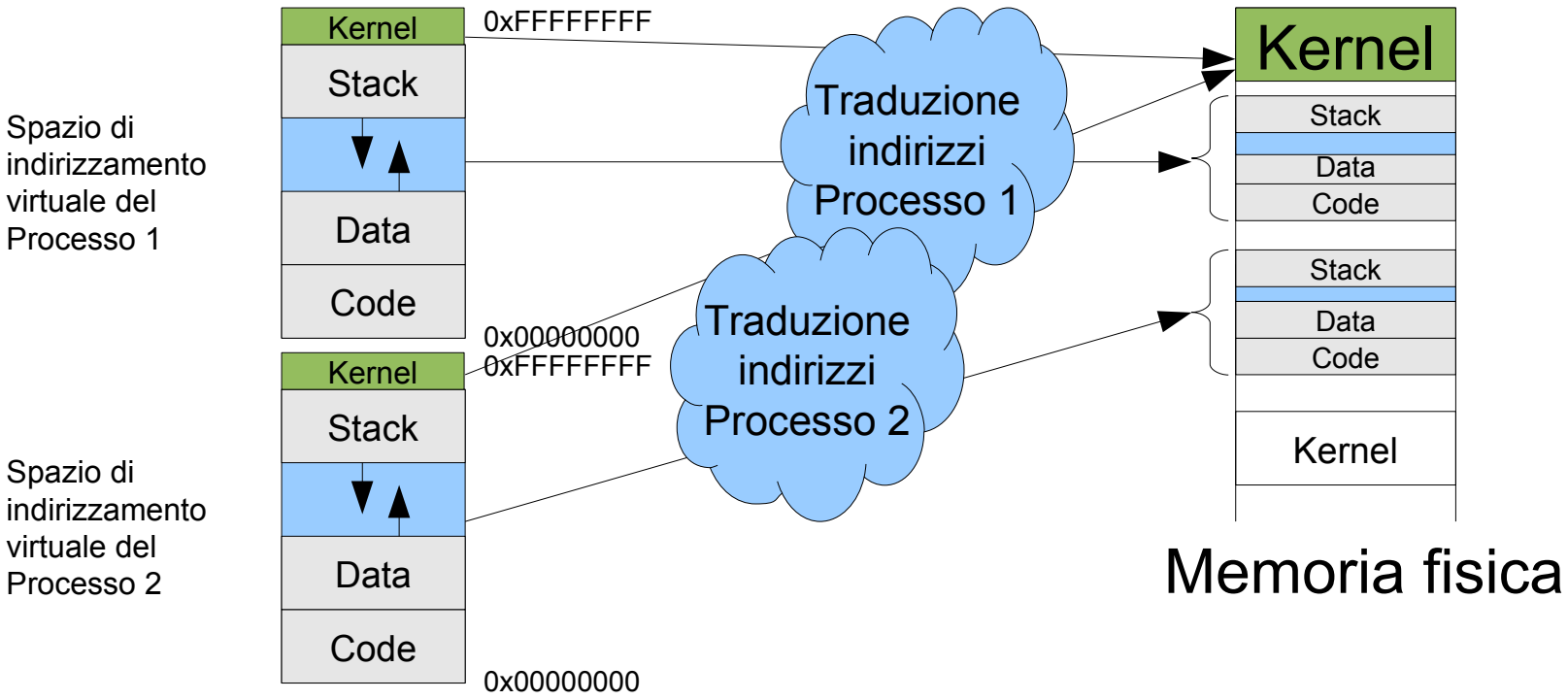


Isolazione dei processi



Spazi di indirizzamento in memoria separati per ogni processo

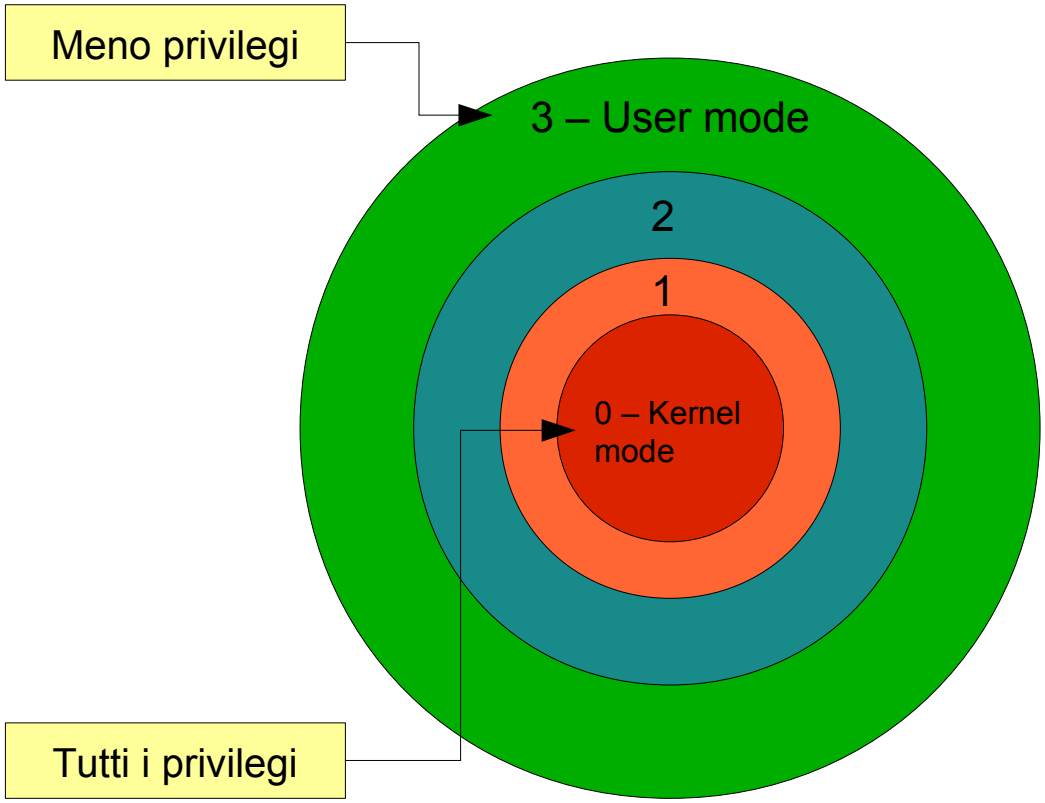
Spazi di indirizzamento e traduzione degli indirizzi



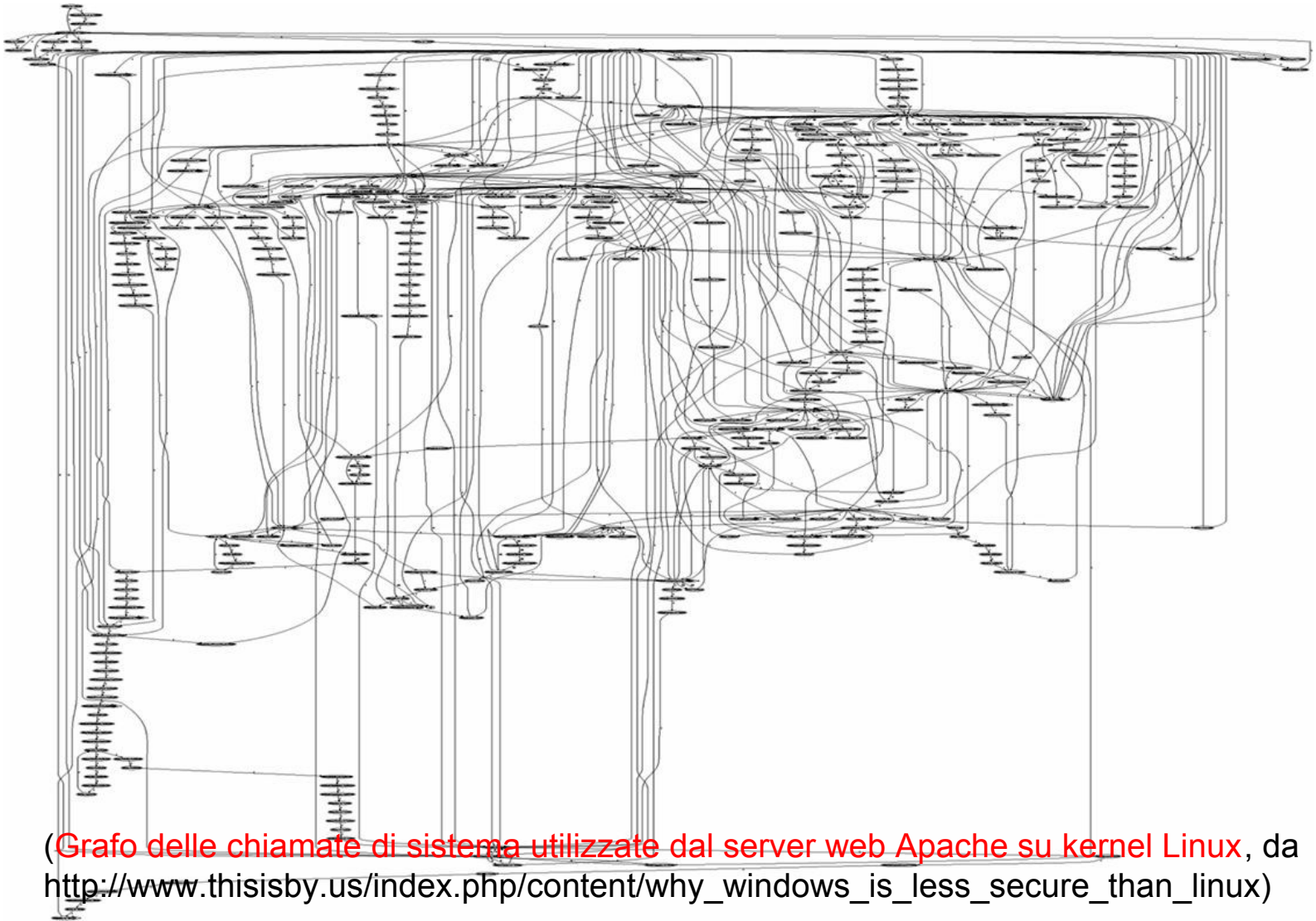
Protezione (kernel mode, user mode)

- Alcune istruzioni (per esempio quelle per gestire la traduzione degli indirizzi di memoria) non devono poter essere accessibili ai processi utente
  - È necessario differenziare tra due modalità di esecuzione:
    - **Priviligiata** (kernel mode)
      - Ha il controllo completo dell'hardware
      - “Da grandi poteri derivano grandi responsabilità”: un errore può causare un crash dell'intero sistema
    - **Non privilegiata** (user mode)
      - Isolazione
      - Un errore deve interessare solo il singolo processo
      - Accesso all'hardware controllato:
        - » Tramite **chiamate di sistema** (system call)

Struttura a cipolla

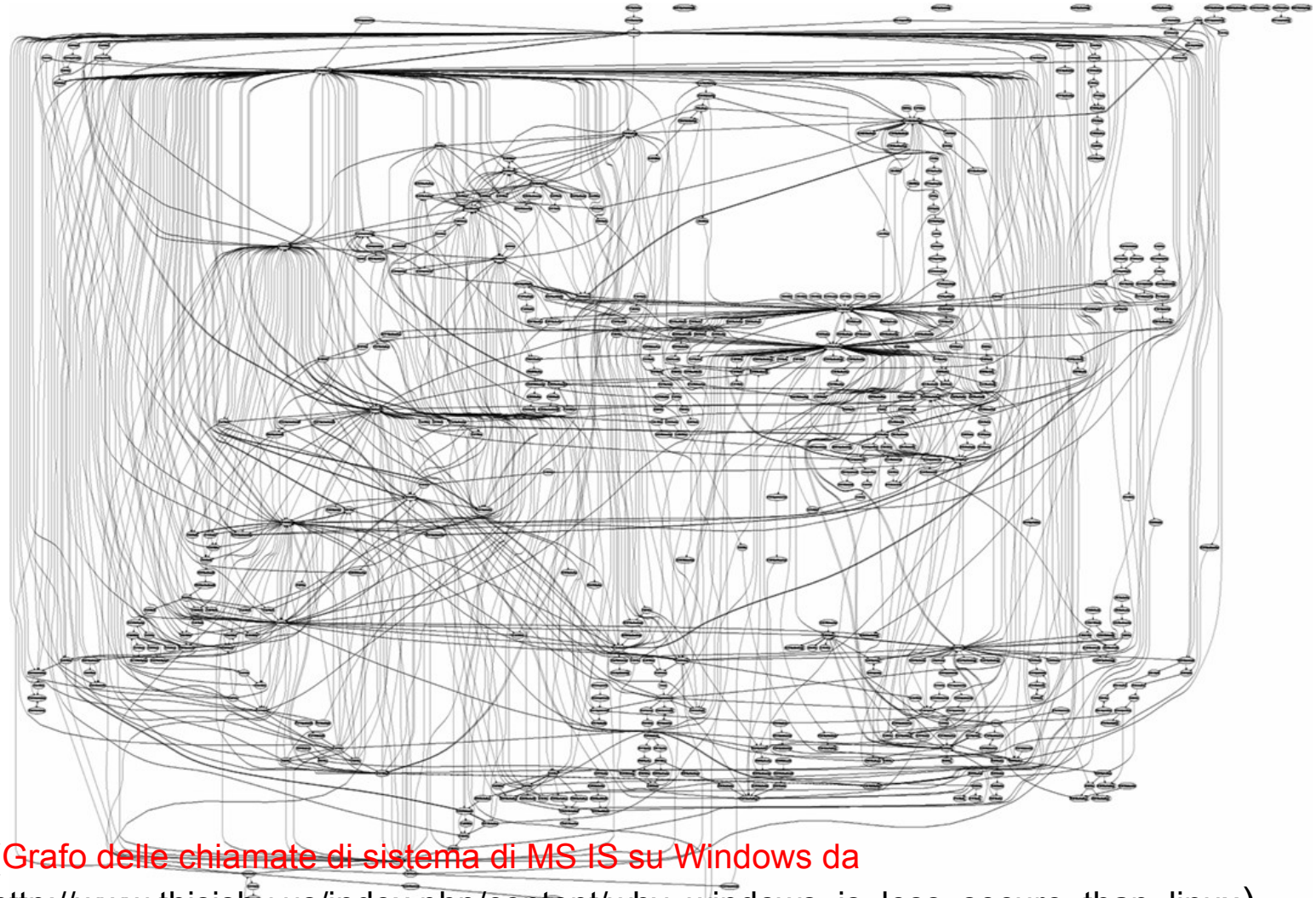


Interazione tra le applicazioni e il kernel: chiamate di sistema



(Grafo delle chiamate di sistema utilizzate dal server web Apache su kernel Linux, da http://www.thisisby.us/index.php/content/why_windows_is_less_secure_than_linux)

Interazione tra le applicazioni e il kernel: chiamate di sistema



(Grafo delle chiamate di sistema di MS IS su Windows da
http://www.thisisby.us/index.php/content/why_windows_is_less_secure_than_linux)

Struttura dei sistemi operativi

- Passare da modalità utente a modalità kernel è “più costoso” che chiamare una funzione nello stesso livello di protezione:
  - quindi perché non eseguire la maggior parte delle operazioni in modalità kernel?
    - sicurezza, robustezza
    - vale il “*principio del privilegio minimo*”
      - concedere ad ogni funzione del S.O. solo i privilegi di cui ha realmente bisogno

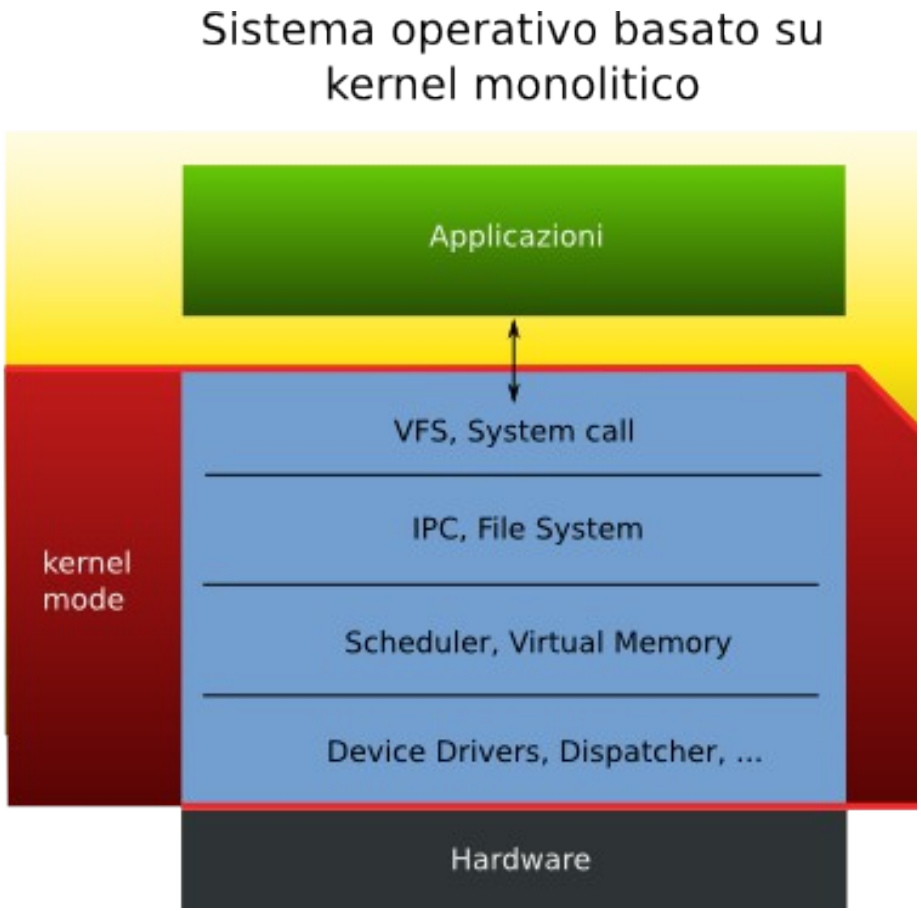
Tipologie

- Diverse tipologie di sistemi operativi possono definire diversamente quali funzionalità sono da eseguire in modalità “utente” e quali in modalità “kernel”
  - Non tutto *può* essere eseguito in modalità utente
  - Non tutto *deve* essere eseguito in modalità kernel

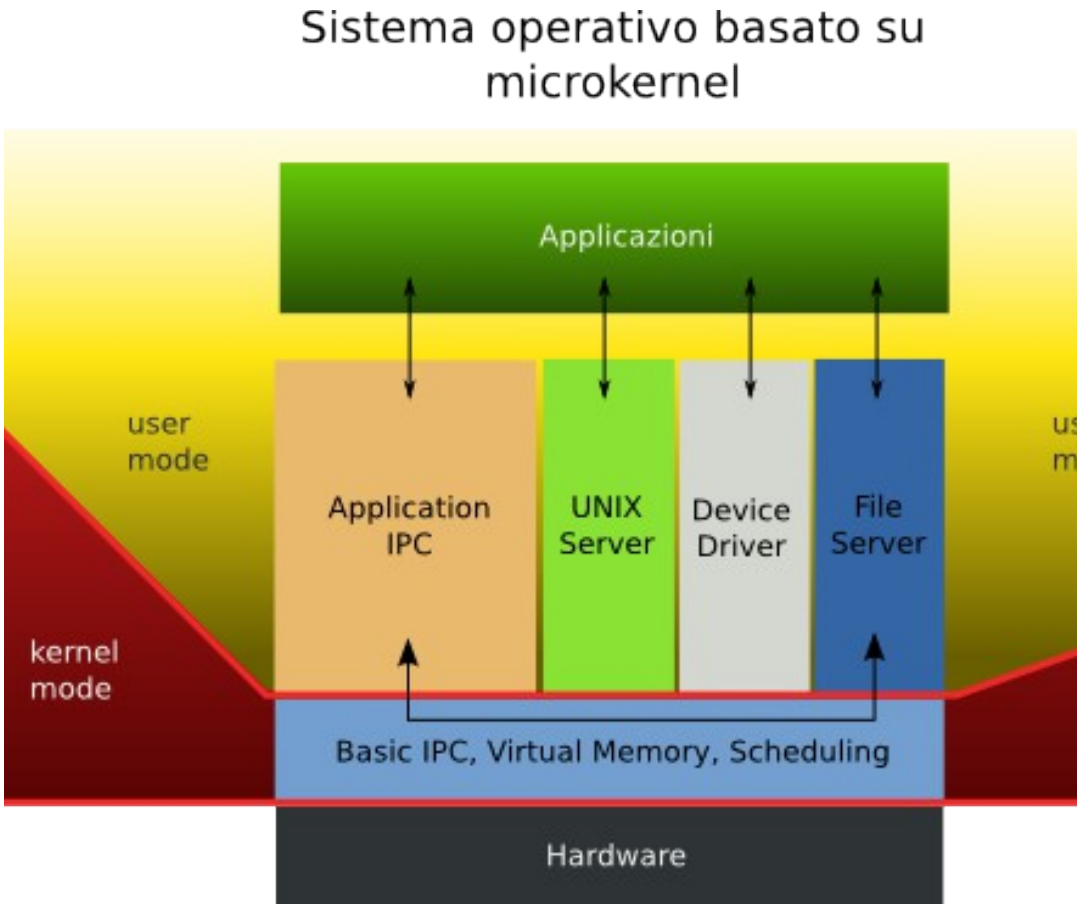
Tipologie

- **Sistemi micro-kernel**
  - Solo le funzionalità strettamente necessarie vengono eseguite in modalità kernel
  - Basato su server indipendenti eseguiti in modalità utente che comunicano tramite messaggi (più robusto)
- **Sistemi monolitici**
  - Tutte le funzionalità sono eseguite in modalità kernel
- **Sistemi ibridi**
  - Struttura modulare simile a un microkernel ma eseguita in modalità kernel

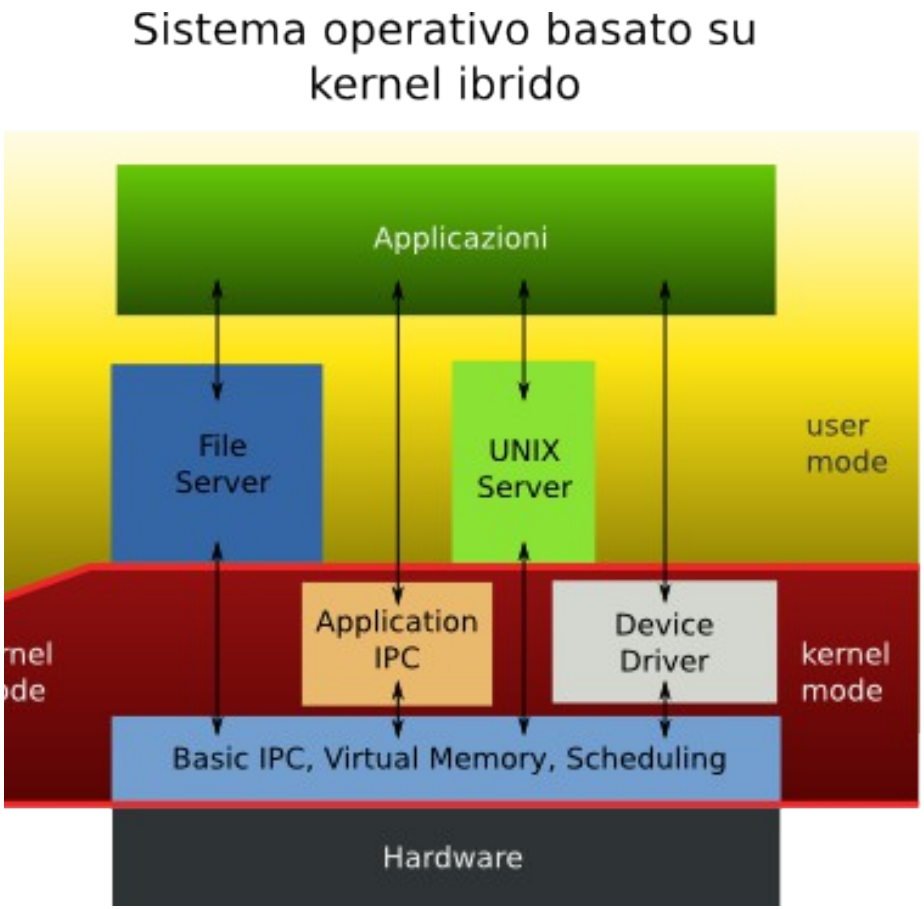
Kernel Monolitici



Microkernel



Kernel Ibridi



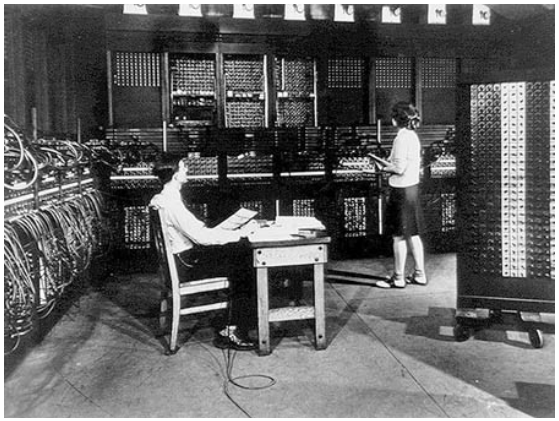
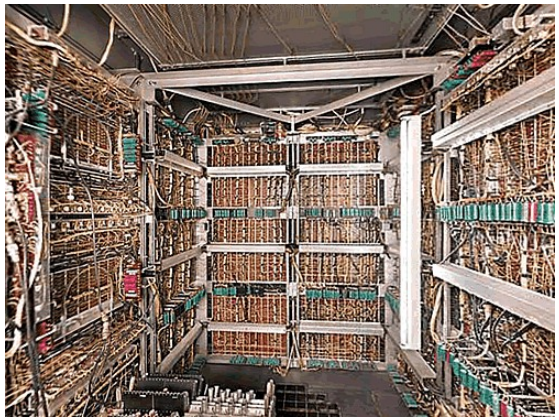
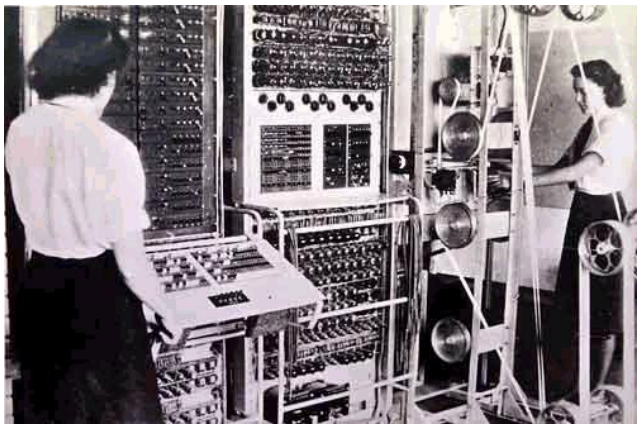
Storia dei sistemi operativi

- L'evoluzione dei sistemi operativi
  - è stata spinta dal progresso tecnologico dell'hardware
  - ha guidato il progresso tecnologico dell'hardware
- Perché analizzare la storia dei sistemi operativi?
  - Perché permette di capire l'origine di certe soluzioni presenti nei SO attuali
  - Perché è l'approccio migliore per capire come certe idee si sono sviluppate
  - Perché alcune delle soluzioni più antiche sono ancora utilizzate

Storia dei sistemi operativi: Prima generazione (1945-1955)

- I primi elaboratori elettronici erano progettati con valvole termoioniche
  - Occupavano intere stanze
  - Costosi: soltanto grossi centri di calcolo o università potevano permetterseli
  - Usati solo per calcoli numerici (calcolatori)
  - Inaffidabili (guasti frequenti)

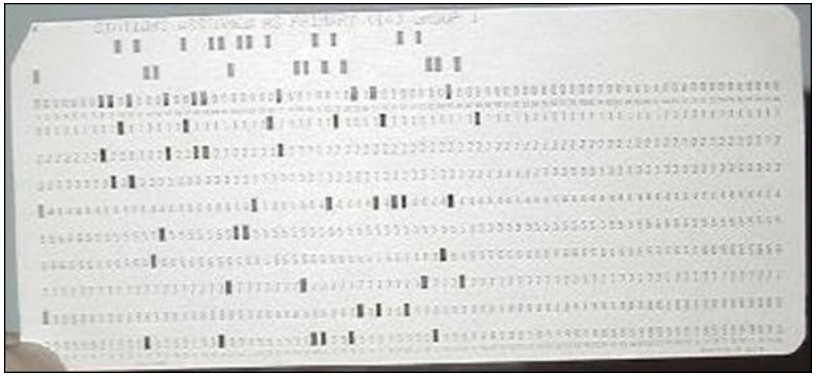
- In questo periodo non esisteva ancora il concetto di sistema operativo
- Venivano programmati in linguaggio macchina (stringhe di 0 e 1)
  - La programmazione avveniva su tavole di commutazione

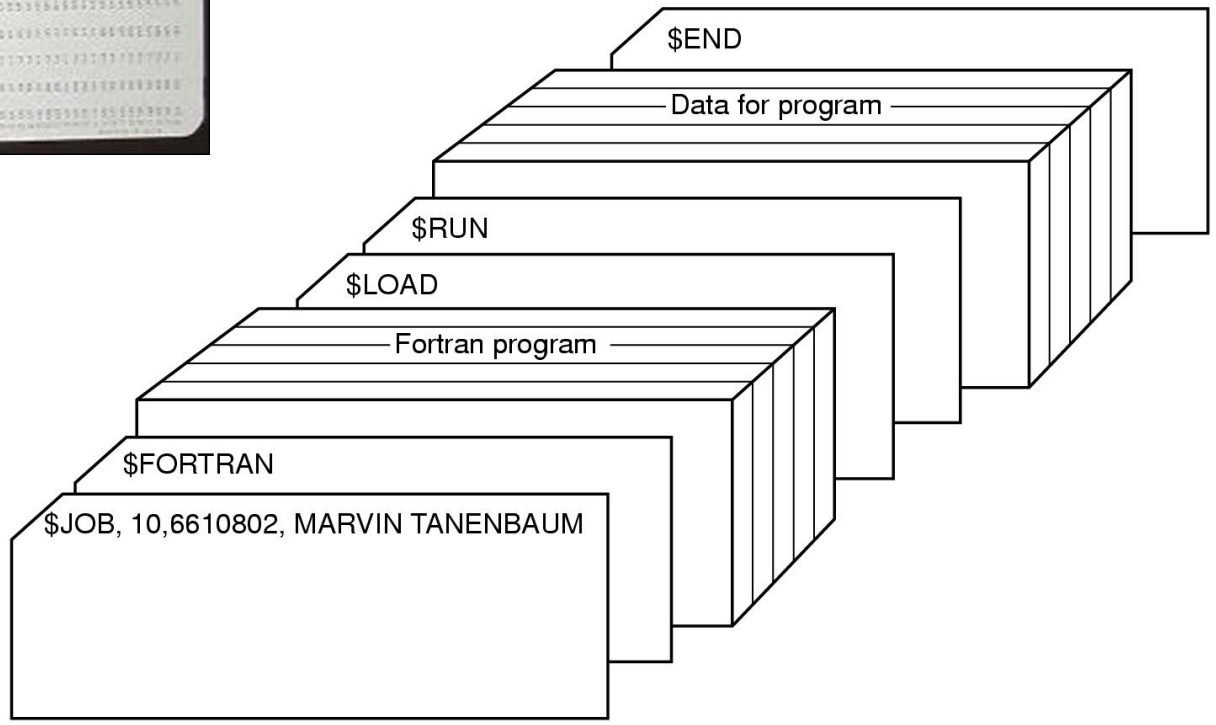




Storia dei sistemi operativi: Seconda generazione (1955-1965)

- Transistor
  - Più affidabilità
  - Minor costo
- Programmazione tramite schede perforate
  - Esecuzione di un programma (job):
    - Scrittura del programma su carta
    - Trasferimento su schede perforate
    - Caricamento nel computer
    - Esecuzione e stampa del risultato

Storia dei sistemi operativi: Seconda generazione (1955-1965)

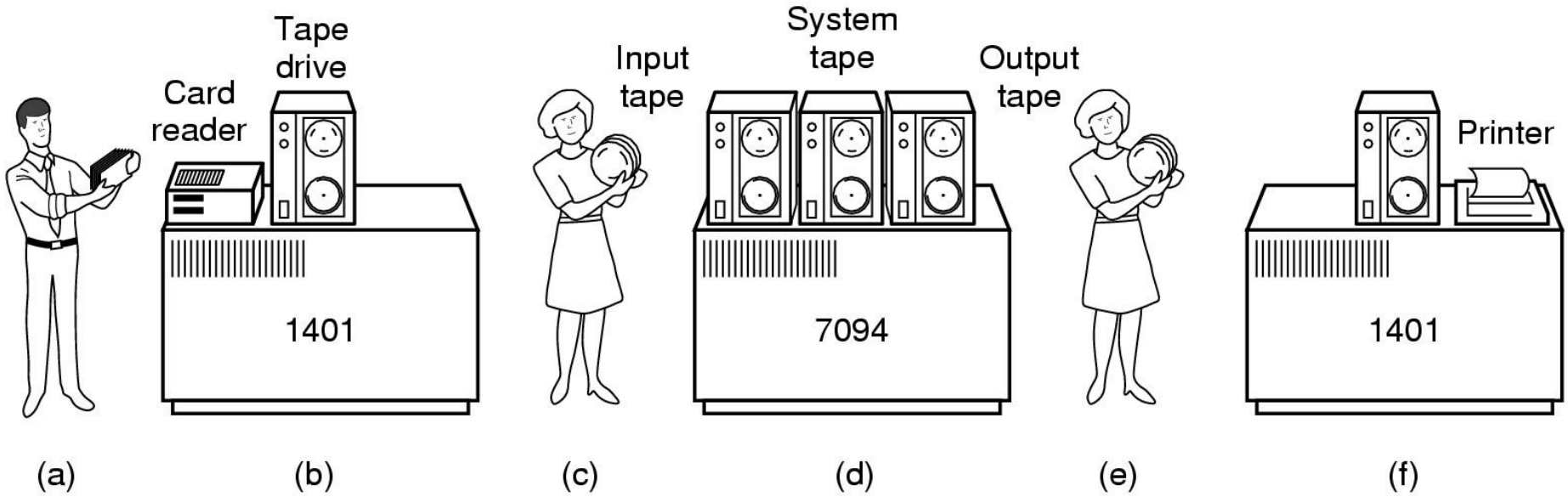




Tratto da [TAN01]

Storia dei sistemi operativi: Seconda generazione (1955-1965)

- Sistema batch (a lotti)
  - dividere i tre lavori, ovvero il caricamento dei dati, il calcolo e la stampa su macchine distinte
  - permette di aumentare la capacità di processing del sistema



Tratto da [TAN01]

Storia dei sistemi operativi: Seconda generazione (1955-1965)

- I sistemi operativi tipici per questi elaboratori, per lo più programmati in FORTRAN e in Assembler erano il FMS (Fortran Monitor System) e IBSYS
- Il sistema operativo residente è in grado di eseguire una sequenza di job, trasferendo il controllo dall'uno all'altro in successione
  - Controllo iniziale del computer al sistema operativo
  - Il controllo viene ceduto al job corrente
  - Una volta terminato il job, il controllo ritorna al sistema operativo

Storia dei sistemi operativi: Terza generazione (1965-1980)

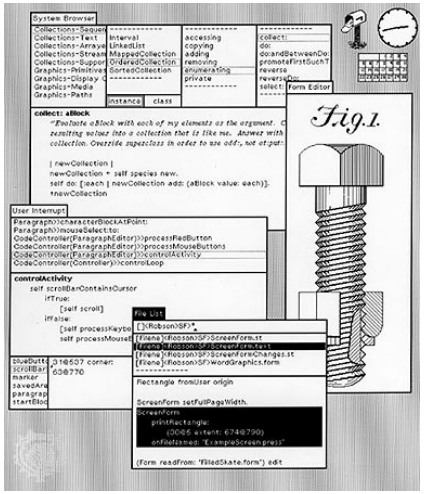
- Nel 1964 IBM presenta una famiglia di computer chiamata IBM System/360:
  - Distinzione tra architettura hardware e implementazione
  - Serie di sistemi compatibili
  - Le differenze erano solo nelle performance (memoria, processore, numero di periferiche) e nel prezzo
  - Possibilità di scrivere programmi capaci di funzionare su macchine diverse della serie 360
- Sistemi basati su circuiti integrati
- Per la prima volta è introdotta la multiprogrammazione
  - più programmi in memoria contemporaneamente
  - necessità di hardware specializzato per proteggere i programmi dalle reciproche interferenze

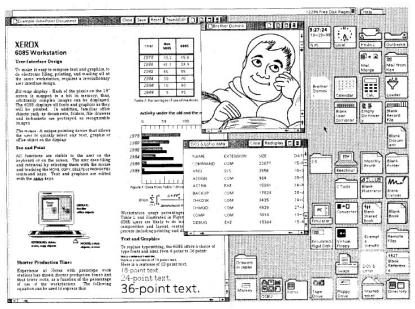
Storia dei sistemi operativi: Quarta generazione (1980-oggi)

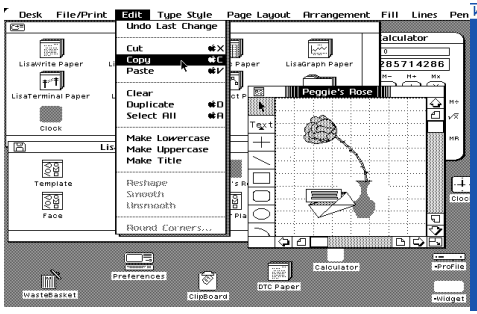
- Tecnologia LSI (large scale integration)
  - chip integrati
  - diminuzione prezzi dell'hardware
  - personal computer
- I personal computer sono dedicati ad utenti singoli:
  - L'obiettivo primario per i SO diventa la facilità d'uso; diminuisce l'interesse per la gestione ottima delle risorse
  - I SO per PC sono in generale più semplici; non implementano la protezione (almeno fino all'avvento di Internet)
  - Creazione di interfacce grafiche user–friendly
  - Tuttavia, tecnologie sviluppate per SO più complessi possono comunque essere adottate
- CP/M-80 della Digital Research per le CPU 8080 / 8085 / Z-80
- MS-DOS (o PC-DOS da IBM)

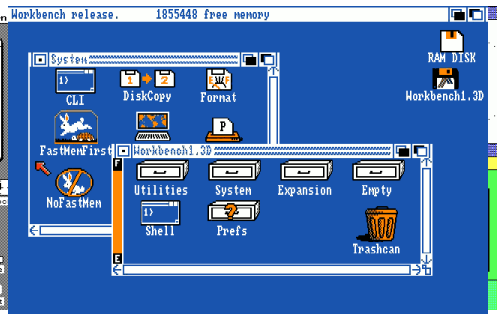
Storia dei sistemi operativi: Quarta generazione (1980-oggi)

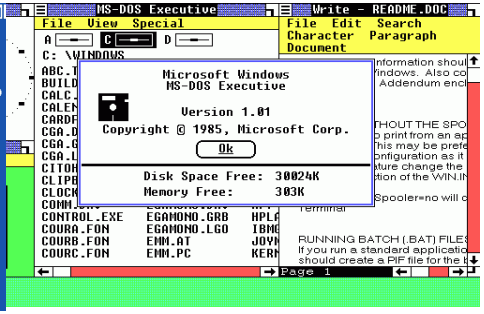
- Interfacce grafiche
  - Xerox Alto (1973) Star (1981)
  - Apple Lisa (1983)
  - AmigaDOS / Workbench (1985)
  - Microsoft Windows (1985)











Storia dei sistemi operativi: Quarta generazione (1980-oggi)

- Nuovi scenari
  - Internet
    - sistemi operativi di rete
    - sistemi operativi distribuiti
  - Mobile computing
    - dispositivi handheld (Android, Symbian OS, iOS,...)
  - Microcontrollori (FreeRTOS, Contiki,...)
  - Sistemi multiprocessore/multicore
- Nuove problematiche
  - Sicurezza
  - Efficienza
  - Facilità di utilizzo