Sistemi Operativi

Alexej Nardovičk

September 28, 2022

1 Contenuti del corso

- **Funzione** e **struttura** di un Sistema Operativo
- **Sistemi a processi**: proprioetà di processi e thread
- Gestione dell'**unità centrale** - Alg.s di scheduling
- Gestione della **memoria centrale**
- Gestione del **file system**
- Gestione della **memoria secondaria** e i dispositivi di I/O
- **Casi di studio** - Unix/Linux. Linee generali di Windows e MacOS.

2 Introduzione

2.1 Cos'è un Sistema Operativo?

- macchina estesa
- gestore di risorse

3 Storia dei Sistemi Operativi

3.1 I generazione – 1945 - 1955

La prima generazione degli elaboratori elettronici è caratterizzata dall'utilizzo della tecnologia delle valvole e delle tavole di commutazione (*plug boards*), insieme ad una programmazione avvenuta direttamente in linguaggio macchina.
Alcuni antecedenti di queste macchine furono la *Macchina analitica* del matematico Charles Babbage (1837 ca.) e il lavoro di Lady Ada Lovelace, considerata la prima programmatrice della storia.
La programmazione in questa epoca avveniva senza alcun sistema operativo di riferimento, bensì direttamente in linguaggio macchina o cablando manualmente i circuiti del calcolatore.
Gli anni '40 videro la nascita di alcuni dei primi calcolatori *general purpose*, (*i.e.* macchine senza un unico programma di esecuzione ma programmabili per eseguire calcoli generici), tra i quali il *MARK I* della Harvard University costruito da IBM, l'*ENIAC* (*Electronic Numerical Integrator and Calculator*) e successivamente l'*EDVAC*, (*Electronic Discrete Variable Automatic Computer*), la prima macchina basata sulla tecnologia dei transistor che segue l'architettura di von Neumann.
Queste macchine erano molto ingombranti fisicamente, e offrivano una potenza di calcolo minima rispetto a quello che ci offrono oggigiorno i Personal Computer: erano noti infatti i problemi relativi all'affidabilità, alla complessità d'uso e alla lentezza, dovuti tra le altre cose ad una mancata distinzione tra i ruoli di costruzione, progettazione, programmazione e manutenzione delle macchine.
La memorizzazione di dati e programmi avveniva prima attraverso schede perforate, successivamente tramite transistor.

3.2 II generazione – 1955 - 1965

A partire da metà anni '50 prese piede lo sviluppo di linguaggi di programmazione ad alto livello, tra i quali ricordiamo sicuramente il linguaggio assembly e Fortran. La memorizzazione di dati e programmi avveniva ancora tramite schede perforate e transistor, e i calcolatori erano caratterizzati dalla cosiddetta **monoprogrammazione**: i programmi utente memorizzati su schede perforate e in attesa di esecuzione venivano caricati ed eseguiti nelle macchine **uno alla volta**. Questi programmi o insiemi di programmi prendono il nome di **job**, e la loro esecuzione avveniva in differita attraverso un operatore che eseguiva dei **comandi *batch***.
Questa generazione vede anche la distinzione tra i seguenti ruoli:

- Costruttori
- Programmatori (programmavano in *high level language*)
- Operatori (gli effettivi utilizzatori dei calcolatori; gestivano le schede di programma, le schede JCL (Job Control Language) e l'I/O)

In questi anni cominciano a nascere i primi sistemi operativi per semplificare l'uso dei calcolatori, che funzionavano appunto secondo il concetto di *batch*:

1) I programmi e i dati venivano caricati in sequenza su un nastro
2) Questi nastri venivano spostati sul computer ed eseguiti a lotti (da qui *batch*)
3) L'output veniva stampato su nastro
4) Altri computer più piccoli si occupavano della lettura/scritta da/verso nastro

3.3 III generazione – 1965 - 1980

La terza generazione di calcolatori conserva dagli anni precedenti l'utilizzo di sistemi di elaborazione *batch*, tuttavia con una grande novità: quella della **multiprogrammazione**: secondo questo principio, un job può usare il processore, mentre altri job usano le unità periferiche di cui hanno bisogno. Questo portò ad una partizione della memoria, assegnandone le varie parti

Figure 1: Struttura tipica di un job - Fortran

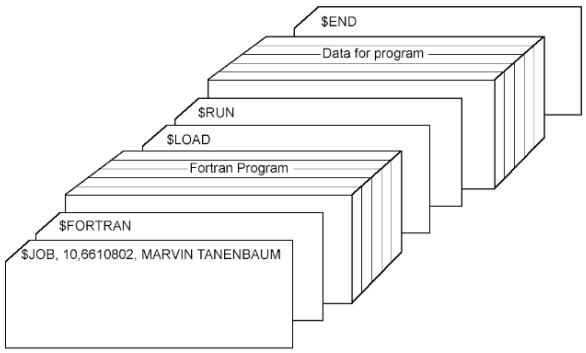
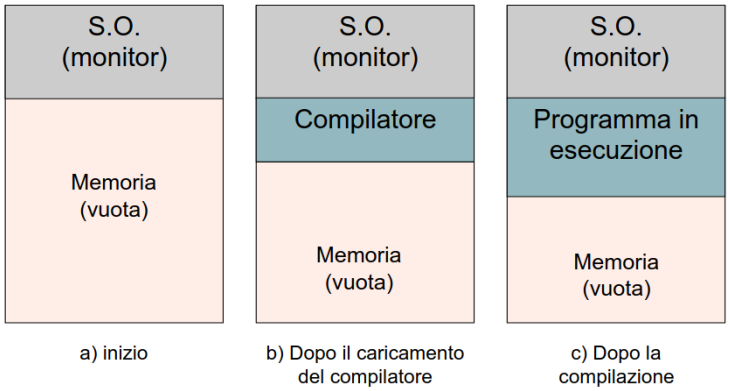


Figure 2: Organizzazione della memoria durante l'esecuzione di un sistema *batch*



a diversi job, in favore di un'esecuzione contemporanea di diversi job e una potenza di calcolo decisamente superiore.
Per gestire questo *pool* di job da eseguire, nacque la necessità di algoritmi di **scheduling** della CPU: algoritmi efficienti grazie ai quali il processore sa come, quando e quanto allocare risorse ai programmi che più ne hanno bisogno in un determinato momento. Vennero quindi progettati i primi ***scheduler***, software del sistema operativo adibiti a questo compito.
In questi anni si progettarono nuovi linguaggi di programmazione ad alto livello (*e.g.* C) e si diffuse l'uso di editor testuali e grafici.
Un'altra novità tecnologia di questa generazione è lo sviluppo dei **circuiti integrati**, che permisero una riduzione dei costi ottenendo anche una mag-

gior velocità, e la nascita di sistemi operativi più complessi.
Tra le macchine degne di nota ricordiamo la famiglia di elaboratori *System/360* di produzione IBM e finalizzata all'uso per *mainframe* e i minicalcolatori PDP, caratterizzati dai costi contenuti e dalle prestazioni limitate.

Studiamo più nel dettaglio alcuni concetti della multiprogrammazione:

- **SPOOL** (*Simultaneous Peripheral Operation On Line*): tecnologia che permette l'esecuzione di operazioni concorrenti e l'esecuzione di job parallela alle operazioni I/O e al trasferimento dati.
- **Sistemi timesharing**: variante della multiprogrammazione, sviluppata per supportare molti utenti interattivi simultanei ai terminali. Questa tecnologia prevedeva di dividere il tempo della CPU in **quanti di tempo**, di lunghezza arbitraria, al termine dei quali il job viene interrotto e si assegna la CPU al job successivo (*prelazione*).
  Questo concetto comportava dunque:
  - Cambi di contesto frequenti (**context switch**, che avvengono tuttora nelle architetture moderne)
  - Attenzione alla protezione e sicurezza dei dati e della memoria (un job non può poter accedere ai dati di un altro job)
  - Principio di sviluppo della tecnologia della **memoria virtuale**
  - Tempo di risposta ridotto a minuti/secondi

  Per quanto questo concetto possa sembrare estremamente efficiente, bisogna selezionare la dimensione del quanto di tempo con una certa cognizione di causa: un quanto troppo breve causerebbe un overhead troppo grande dovuto alla maggior frequenza di cambi di contesto, mentre un quanto troppo lungo rallenterebbe job di minore entità in favore di job più complessi.

Alcuni esempi di tecnologie di timesharing introdotte in questo periodo furono il sistema **CTSS** (*Compatible Time Sharing System*) da parte dell'MIT, evoluto poi nel **MULTICS** (*Multiplexed Information and Computer Service*), in cui venne introdotto il concetto di **processo**. Questi sistemi, insieme al

TSS (*Time Sharing System*, su cui si basa *Unix*, di cui parleremo in seguito) e al **CP/CMS** (*Control Program/Cambridge Monitor System*) includono il concetto di **memoria virtuale**, per dare ai processi l'illusione di avere maggior memoria di quella fisicamente disponibile.
Nacquero anche i sistemi **real-time**, atti a fornire una risposta entro un dato periodo di tempo limitato.

Abbiamo già menzionato *Unix*, sistema operativo derivato da CTSS e MULTICS che implementa le tecnologie di timesharing multiprogrammati che abbiamo visto finora. Nel 1974 *Unix* fece registrare una doppia licenza, commerciale e libera, e venne pubblicato il suo codice sorgente: per rendere compatibili le diverse versioni che si vennero a creare di questo sistema operativo, fu introdotto lo standard *POSIX*, appartenente allo standard *IEEE*, dal momento che disponendo del codice sorgente ognuno aveva la possibilità di creare una nuova versione del sistema operativo. Questa possibilità fu anche favorita dallo sviluppo dei primi semplici Personal Computer, costituiti da una tecnologia a microprocessori.

Infine, in questa generazione si iniziò a sviluppare anche la tecnologia Internet e i protocolli TCP/IP da parte del Dipartimento della Difesa americano, usati in ambienti militari e accademici per una massiccia e veloce trasmissione dei crescenti volumi di dati.

3.4 IV generazione – 1980 - presente

A partire dagli anni '80 crebbe fortemente lo sviluppo delle **workstation** e dei **personal computer**, il cui utilizzo venne fortemente semplificato grazie all'introduzione delle **interfacce grafiche (GUI)**. Il trasferimento di informazioni tra computer tramite **reti** divenne molto più pratico ed economico, e l'elaborazione dei dati diventò man mano sempre più **distribuita** ai siti in cui veniva richiesta.
Alcuni esempi di sistemi operativi introdotti in questa generazione furono:

- *CP-M80 Digital*, divenuto poi *MS-DOS*
- *LisaOS* (1983), primo SO per PC con GUI, seguito da *MacOS* (1984) e *MacOSX* (1999)
- *Windows* (1985), *Windows 3* (1990) che adottò la tecnologia della

memoria virtuale, *Windows 95* (1995), *Windows NT* (1998), *Windows XP* e *Windows Millennium Edition* (2001), *Windows 7* (2017), *Windows 8* (2012) per sistemi touch, *Windows 10* (2015).
Tutti i sistemi operativi di *Microsoft* accennati sopra sono stati inizialmente pensati per girare su PC, ma vennero poi anche adattati per le *workstation*.

- *Linux*, *Unix* - dotati di interfaccia *Xwindows* (basato su *X11* del MIT), poi *BSD*, *Xenix*, *SunOS*, *Solaris*, *FreeBSD* (da cui derivò anche *MaxOSX*)

Nacque in questa generazione la distinzione tra **cliente** e **servente**, concetti chiave su cui si svilupperà poi il relativo modello di elaborazione:

- I **clienti** richiedono diversi servizi ai serventi
- I **server** eseguono le richieste di servizio

Subì una forte evoluzione anche l'area dell'ingegneria del software, promossa anche dal governo degli Stati Uniti per un controllo rigoroso dei progetti software del Dipartimento della Difesa, realizzando alcuni traguardi della storia dell'informatica quali i concetti di **riusabilità del codice**, un maggior grado di **astrazione nei linguaggi di programmazione** e il **multithread** di istruzioni da poter eseguire in modo indipendente.

Il decennio successivo vide la nascita delle applicazioni e delle **reti di calcolatori**, tra cui ricordiamo il *World Wide Web* e *Microsoft Office*. In questo periodo sistemi operativi e calcolatori erano già molto simili a quelli che usiamo al giorno d'oggi, infatti è a partire dagli anni '90 che si fondarono le basi dei dispositivi che usiamo tuttora: PC, smartphone e tablet (con relativi sistemi operativi), sistemi distribuiti, cloud computing...
Tutte queste innovazioni furono favorite dall'importante **decrescita del costo** della capacità di elaborazione e della memoria, che rese possibile l'esecuzione di programmi grandi e complessi su PC e ampliò la disponibilità di sistemi economici per la memorizzazione di grandi database e l'elaborazione di job, riducendo quindi la necessità di mainframe centralizzati: diretta conseguenza di ciò fu lo sviluppo di metodi e tecniche per l'elaborazione di calcolo distribuito, tramite appunto i cosiddetti **sistemi distribuiti**, cioè sistemi di elaborazione indipendenti che cooperano per raggiungere un obbiettivo comune.

- **Sistemi paralleli**
  - Un sistema con un ampio insieme di unità di elaborazione
  - Accoppiamento stretto, comunicazione rapida
  - Risorse condivise (*e.g.*, la memoria)
  - Alta affidabilità e prestazioni
- **Sistemi distribuiti**
  - Sistema costituito da un insieme di unità di elaborazione complete interagenti e cooperanti, collegate da linee di comunicazione
  - Omogeneo o eterogenei
  - Architetture
  - Condivisione di risorse, prestazioni, affidabilità e trasparenza

Sempre in questo decennio vennero creati degli **standard** di sistemi operativi che supportano **networking tasks**, per l'aumento della **produttività e comunicazione**, ed è in questo periodo che la **Microsoft Corporation** divenne dominante, grazie all'introduzione nei sistemi operativi *Windows* di tecnologie presenti nei primi Macintosh che permettevano agli utenti di navigare più applicazioni concorrenti con una certa facilità.

Importante fu anche lo sviluppo della **tecnologia a oggetti**:

- Molte applicazioni si scrivevano in **linguaggi di programmazione orientati agli oggetti** (*e.g.*, C++ o Java)
- **Sistemi operativi orientati agli oggetti**, oggetti appunto che rappresentano componenti del SO
- Concetti come **eredità** e interfacce sfruttati per creare sistemi operativi **modulari** e che permettevano di mantenere ed estendere il software rispetto alle tecniche precedenti

Va adesso fatta una digressione storica sulla questione della filosofia *open-source*: negli anni '90 la maggior parte dei software commerciali era venduta sotto forma di **codice oggetto** già compilato, non includeva il codice sorgente vero e proprio; questo serviva ai produttori di sistemi operativi per

nascondere le tecniche di programmazione e informazione proprietaria.
In questo decennio si diffuse sempre di più l'ideologia del software libero, ***open-source*** appunto, distribuito insieme al codice sorgente, che ne permette l'esame e la modifica. Alcuni software estremamente famosi utilizzati tutt'ora, come **Linux** e **Apache Web Server**, sono *open-source*.
Una pietra miliare della storia dell'*open-source* fu la nascita del progetto ***GNU***, a cura di Richard Stallman, che consisteva in un progetto di **software libero** che si poneva l'obiettivo di ricreare ed estendere gli strumenti per il sistema operativo **Unix** di *AT&T*, schierandosi contro il concetto di costo per l'uso di software.
Sulla scia di Richard Stallman nacquero altre iniziative a favore della promozione dell'*open-source*, tra le quali ricordiamo la ***Open Source Initiative***, ***OSI***:

- Organizzazione nata per promuovere i benefici della programmazione *open-source*
- Facilita il rapido miglioramento dei prodotti software, permettendo a chiunque di **testare**, eseguire il **debug** e **migliorare le prestazioni**
- Aumenta la probabilità di riconoscere e risolvere i *bugs* per risolvere anche problemi relativi alla sicurezza dei programmi
- Gli individui e le aziende possono modificare il codice sorgente per creare software personalizzato secondo l'esigenza del relativo ambiente (**adattabilità**)

Possiamo concludere questa rassegna sulla IV generazione dei sistemi operativi ribadendo alcuni concetti chiave:

- I sistemi operativi diventano sempre più di facile uso (*user friendly*)
- Le caratteristiche delle ***GUI*** introdotte da *Apple* e adottate poi da tutte le altre grandi aziende sono sempre più diffuse e migliorate
- Le funzionalità "**Plug-and-play**" sono integrate nei sistemi operativi: permettono agli utenti di aggiugnere e rimuovere dinamicamente componenti hardware, senza il bisogno di riconfigurare manualmente il sistema operativo
- I **servizi web** cominciano la loro ascesa verso l'utilizzo commerciale:

- Comprendono un insieme di **standard** correlati
- Componenti software ready-to-use su Internet
- Permette a due qualsiasi applicazioni di comunicare e scambiare dati
- Fanno largo uso di **middleware**: un insieme di software che collegano applicazioni separate e diverse (spesso tramite rete e anche fra macchine eterogenee, dotate di diverse architetture)

3.4.1 Cenni di storia di Internet e World Wide Web

Prima di affrontare l'ultima generazione della storia del sistemi operativi, è importante soffermarci su tempi e modalità che alla fine del II millennio portarono alla nascita del sistema di reti del *World Wide Web*.

Come abbiamo già accennato, gran parte dello sviluppo di Internet fu promosso dai governi, specialmente quello degli Stati Uniti d'America: alla fine degli anni '60, il Dipartimento della Difesa americano creò e implementò **ARPAnet**, una sorta di predecessore di Internet, una rete progettata per trasmettere dati tra i principali sistemi delle istituzioni finanziate da ARPA **senza controllo centralizzato**. Questo antecedente di Internet era già allora capace di una trasmissione di informazioni quasi istantanea tramite e-mail.

Successivo fu poi lo sviluppo del **Transmission Control Protocol/Internet Protocol** (**TCP/IP**),, cioè di un insieme di regole per comunicare su *ARPAnet*. Questi protocolli **gestiscono la comunicazione fra applicazioni**, assicurandosi che i messaggi siano **instradati** correttamente dal mittente al destinatario, limitando gli errori tramite **tecniche di correzione dell'errore**. Gli stessi protocolli vennero poi riutilizzati quando la rete venne aperta anche all'uso commerciale più generale.

Nel 1989, nel suo studio al CERN di Ginevra, Tim Berners-Lee progettò quella che diventerà poi la rete su cui, attualmente, verranno scambiate tutte le informazioni tramite Internet: il **World Wide Web**.
Venne ideato come rete per **individuate e visualizzare documenti multimediali** su argomenti qualsiasi, ma soprattutto per la **condivisione di informazioni** attraverso documenti di testo con **collegamenti ipertestu-**

ali. I documenti presenti in questa rete sono definiti secondo lo standard del linguaggio **HTML** (**HyperText Markup Language**), e il protocollo (cioè le regole di comunicazione) p definito dall'**HTTP** (**HyperText Transfer Protocol**).

3.5 V generazione – 1990 - presente

La caratteristica principale degli elaboratori moderni è sicuramente la **mobilità**: dagli anni '90 infatti lo sviluppo della tecnologia per sistemi mobili crebbe a ritmi esponenziali, partendo dai primi palmari quali il *Nokia N9000* per arrivare ai moderni *smartphone* (termine coniato nel 1997) dotati di sistemi operativi complessi, quali *Android*, basato su Linux e di proprietà di Google, e *iOS*, di proprietà di Apple.
Ciò che contraddistingue i dispositivi mobili moderni è una serie di fattori:

- Risorse limitate
- Alimentazione a batteria
- Diversi dispositivi di I/O
- Uso intensivo di applicazioni, reti e protocolli (*cloud computing*)
- Integrazione in altri oggetti
- Attenzione alla sicurezza

Tutti i moderni calcolatori possono interagire tra di loro attraverso l'***Internet of Things*** (***IoT***), grazie all'evoluzione della rete sulla quale possono scambiare informazioni oggetti statici e mobili (*e.g.*, elettrodomestici, abbigliamento, impianti, macchine, attrezzature...).
Alla base di questa tecnologia si trova il concetto di *ubiquitous computing*, in italiano *computazione ubiqua*, cioè un modello di interazione uomo-macchina secondo il quale l'elaborazione delle informazioni è integrata all'interno di oggetti e attività di tutti i giorni (gli oggetti citati sopra). Per permettere queste implementazioni bisogna chiaramente progettare e sviluppare un sistema operativo anche per questi oggetti, e anche per essi, come per un tradizionale calcolatore, bisogna preoccuparsi di garantirne sicurezza e privacy.
L'informazione può essere scambiata anche attraverso i cosiddetti *smart objects* (cioè tecnologie *RFID*, codici *QR*...).

4 Cos'è un sistema operativo

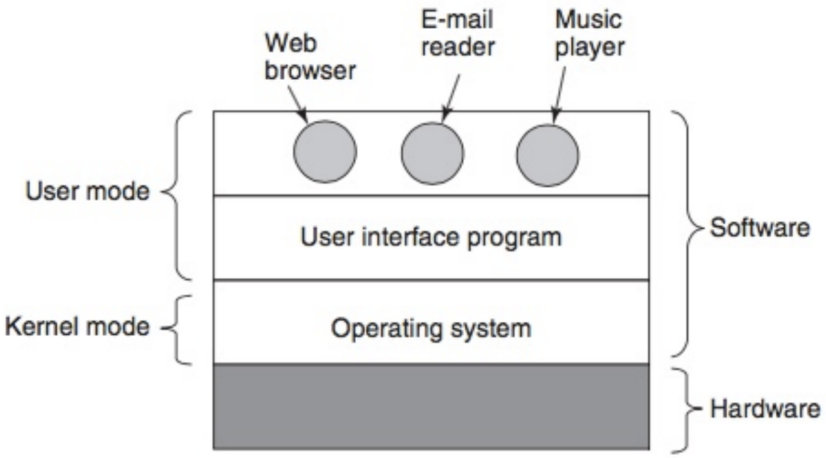
Un S.O. è un software che controlla l'hardware.
Un S.O. è un programma che gestisce e controlla l'esecuzione di un insieme di applicazioni, agisce come interfaccia tra le applicazioni e l'hardware del calcolatore e gestisce le risorse hardware.
Il S.O. è (prevalentemente) eseguito con il processore in modalità kernel Le applicazioni sono eseguite principalmente in modalità utente. Un sistema operativo ha applicazioni separate dall'hardware utilizzato:
- livello software
- gestione software e hardware per produrre i risultati desiderati

Un Sistema operativo innanzitutto è un gestore di risorse:
• Risorse harware;

- processore
- periferiche I/O
- memoria
- periferiche di comunicazione
- ecc.

• Applicazioni software

componenti di un sistema operativo:



5 Sistemi timesharing

- Variante della multiprogrammazione, fu sviluppato per supportare più utenti interattivi nello stesso momento sul terminale
- Il tempo di CPU è suddiviso in **quanti di tempo**
- Al termine del quanto il job viene interrotto e si assegna la CPU al job successivo (prelazione)
- Il tempo di risposta viene ridotto a minuti o secondi

Progettazione dei sistemi timesharing:
- Gestione del processore: scheduling
- Gestione della memoria: memoria virtuale
- Protezione delle risorse (memoria, filesystem, ...)

6 Componenti hardware

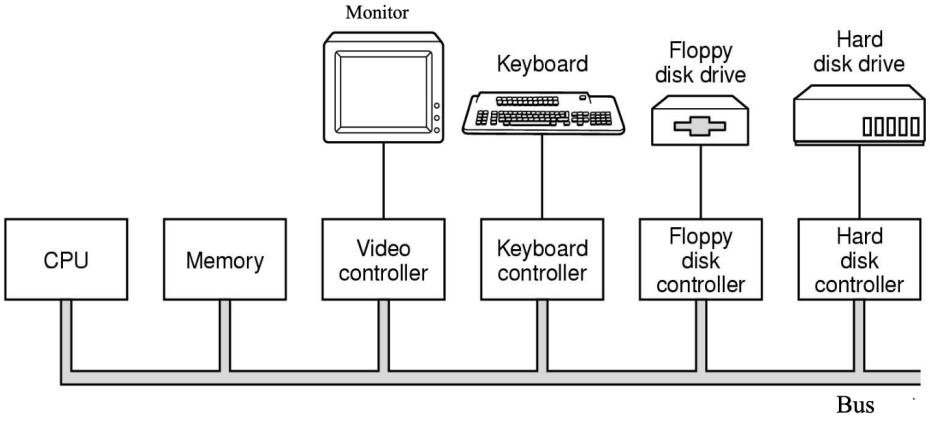
Per progettare un sistema operativo devo conoscere le risorse hardware e software che deve gestire:
- processori
- memoria interna / memoria secondaria
- altre periferiche I/O
- processi
- thread
- file
- database

6.1 componenti hardware - CPU

Il processore è quella parte dell'hardware che **esegue** in linguaggio macchina, la CPU esegue le **isrtuzioni** di un programma.
All'interno della CPU troviamo i **registri**, ovvero delle memorie ad alta velocità situati sui processori. I dati devono essere nei registri prima che un processore possa operarvi.

Il **tempo di elaborazione si misura in cicli**, fornito dal generatore di clock di sistema.
La velocità del processore è misurata in **GHz** (miliardi di cicli per secondo).

Componenti Hardware



Componenti di un semplice personal computer

Evoluzione dei microprocessori con architettura Intel

Microprocessore	Bit	Anno	Transistor	Produttore
4004	4	1971	2,25 K	Intel
8080	8	1974	5 K	Intel
Z80	8/16	1976	6 K	Zilog
8088	8/16	1979	29 K	Intel
80286	16	1982	134 K	Intel,Amd
80386	32	1985	275 K	Intel,Amd
80486	32/64	1989	1,2 M	Intel,Amd
Pentium	32/64	1993	3,1 M	Intel
Pentium III	32/64	1999	9,5 M	Intel
Athlon	32/64	1999	22 M	Amd
Pentium IV	32/64/128	2000	40 M	Intel
Opteron	32/64/128	2003	100 M	Amd
Itanium 2	32/64/128	2004	220M	Intel
Opteron quad core	32/64/128	2006	460 M	Amd
Core 2 Quad	32/64/128	2006	582 M	Intel
Xeon MP 6 core	32/64/128	2008	1.900 M	Intel
Itanium quad core	32/64/128	2008	2.000 M	Intel

La **CPU** si trova in diverse modalità, in mod **nucleo (kernel)** o **utente**. Quando viene chiamata una TRAP (istruzione) viene cambiata la modalità da utente a kernel.

Prestazioni di una CPU

$T = N_i / IPS$ $\quad\quad$ $IPS = F \times IPC = F / CPI$

T	tempo di esecuzione
N_i	numero di istruzioni di un programma
IPS	numero di istruzioni per secondo
F	frequenza di clock del processore
IPC	numero di istruzioni per ciclo di clock
CPI	cicli di clock per istruzione

T può migliorare con

- aumento di F → miniaturizzazione
- riduzione N_i, per lo stesso lavoro → CISC / RISC (Complex/Reduced Instr. Set)
- aumento di IPC ovvero riduzione CPI → pipeline
  → CPU superscalare

6.2 componenti hardware - Scheda madre

Scheda con circuiti elettronici stampati:

- Componente hardware hce fornisce collegamenti elettrici tra i dispositivi
- La scheda madre è il circuito stampato (**PCB**) centrale nel sistema:
  - I dispositivi come CPU e memoria principale sono attaccati
  - include **chip** per eseguire operazioni di basso livello (**BIOS – Basic Input Output System**).
  - BIOS controlla quanta RAM è disponibile, le componenti collegate, scansiona i bus e rileva i dispositivi, poi determina il dispositivo di avvio, carica e avvia il primo settore.

6.3 componenti hardware - Memoria