ARCHITETTURA DEI COMPUTER

Un calcolatore elettronico è una macchina estremamente complessa, costituita da centinaia di componenti hardware. Per componenti software si intende, invece, l'insieme di programmi in grado di far funzionare l'hardware o di eseguire particolari applicazioni. Il componente software più importante per il funzionamento della macchina è sicuramente il Sistema Operativo. I più importanti componenti hardware sono, invece, i seguenti:

- **processore**,
- **memoria principale (o RAM)**,
- **memoria secondaria (o di massa)**
- **dispositivi di input/output**.

L'esecuzione di un programma è subordinata a un input, in genere fornito attraverso qualche dispositivo, per esempio la tastiera. I programmi e i dati che devono essere elaborati risiedono nella memoria secondaria (per esempio il disco rigido). Quando un programma deve essere eseguito viene copiato, insieme ai dati che deve elaborare, nella memoria principale, detta anche RAM (Random Access Memory – cioè memoria ad accesso casuale). Il processore preleva quindi le istruzioni e i dati, eseguendo il programma. Infine, il risultato dell'elaborazione è in qualche modo reso noto attraverso qualche dispositivo di output, per esempio il monitor o la stampante.
Schematicamente in base alle funzioni che svolgono, i componenti principali hardware possono essere classificati nel seguente modo:

Componenti per l'**elaborazione** dei dati
Processore (Central Processing Unit - CPU)

Componenti per la **memorizzazione** dei dati
Memoria principale (o RAM)
Memoria secondaria (o di massa)

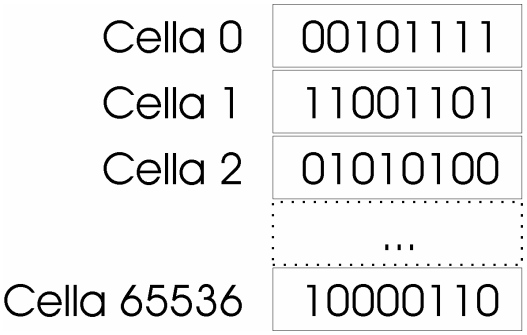
Componenti per il **trasferimento** dei dati
Dispositivi di input/output

Dispositivi di Input/Output

I dispositivi di Input/Output consentono l'interazione uomo-macchina. Rispetto ai primi anni dell'era informatica, in cui ad esempio si usavano la "schede perforate" per la specificazione dei programmi, gli attuali dispositivi di I/O sono abbastanza evoluti e user-friendly. Tra i dispositivi di input ricordiamo la tastiera, il mouse e lo scanner (che permette la digitalizzazione delle immagini), mentre tre i dispositivi di output ricordiamo il video e la stampante.

La memoria principale (RAM)

La memoria principale, o RAM (Rndom Access Memory – cioè memoria ad accesso casuale), contiene le istruzioni del programma e i dati che la CPU può direttamente elaborare. Essa è fisicamente composta da numerosi componenti elettronici (transistors) miniaturizzati ed è generalmente disposta su banchi (banchi di memoria). Dal punto di vista della rappresentazione dei dati in formato binario, ogni "unità elementare" della memoria può trovarsi in due diversi livelli di tensione elettrica, corrispondenti ai bit 0 e 1.
La memoria principale può essere vista come una **sequenza di celle**, ognuna caratterizzata da un **indirizzo** (numerico) e da un **valore** espresso con un **singolo byte**.



Specificando l'indirizzo di una cella, la CPU è in grado di leggere e/o scrivere il valore del byte memorizzato in quella cella. Il nome RAM indica la caratteristica di questo tipo di memoria per cui ogni cella è direttamente "indirizzabile". Cioè, per accedere al valore contenuto nella Cella 2, non è necessario "scorrere" le celle precedenti. In contrapposizione alla memoria ad accesso casuale sono le memorie ad accesso sequenziale, come i nastri magnetici. In questo tipo di memoria, è necessario scorrere tutte le locazioni di memoria che precedono quella d'interesse.
Si definisce **spazio di indirizzamento**, l'insieme (o il numero) delle celle indirizzabili direttamente. Il numero di celle indirizzabili è una potenza di due. Con:

- 16 bit si indirizzano 2^{16} celle = 65.536 celle
- 32 bit si indirizzano 2^{32} = 4.294.967.296 celle
- 64 bit si indirizzano 2^{64} celle
- …

La dimensione della memoria è generalmente misurata tramite multipli di byte. I principali sono i seguenti:

- kilobyte (KB) = 2^{10} byte = 1024 byte =
- megabyte (MB) = 2^{20} byte = 1000 KB
- gigabyte (GB) = 1 miliardo di byte = 1000 MB

Quindi:

- con 16 bit si indirizzano 64KB di memoria
- con 32 bit si indirizzano 4GB di memoria
- …

La parola (word) di un computer rappresenta quanti bit possono essere letti/scritti dalla cpu con un unico accesso alla memoria. Dimensioni tipiche per la parola sono: 16, 32, 64 e 128 bit. In genere, più grande è la parola, maggiore è la potenza del computer.
Alcune caratteristiche della memoria principale sono:

- La **RAM è abbastanza veloce**: per leggere/scrivere una cella sono richiesti, in media 5-30 nanosecondi (millesimi di milionesimi di secondo; $5\text{-}30 \cdot 10^{-9}$ secondi). Questo, tra l'altro, è un valore destinato a scendere per via dei continui progressi tecnologici.
- La **RAM è volatile**: è fatta di componenti elettronici, e se togliete la corrente si perde l'informazione in essi contenuta.

La memoria secondaria

Programmi e dati risiedono normalmente in memoria secondaria, per differenti motivi: (1) la memoria secondaria è più economica della memoria principale (2) **la memoria secondaria non è volatile** e pertanto è in grado di conservare l'informazione in essa contenuta anche in assenza di corrente elettrica. Quando si lancia un programma questo viene copiato dalla memoria secondaria (di solito un hard disk) in memoria primaria. Questa operazione si chiama caricamento ed è eseguita dal sistema operativo.
Benché il floppy disk (dalla capacità di1.44 Megabyte) sia stato uno dei primi esempi di dispositivi di memoria secondaria e sia incredibilmente ancora in uso, l'**hard disk** è sicuramente il dispositivo

più importante. In genere, esso e composto da supporti magnetici permanenti, generalmente dischi, gestiti mediante dispositivi meccanici. Ad esempio, un motore consente ai dischi di girare a velocità costante (es. 5400, 7200 o 10000 rpm - giri al minuto), mentre delle testine "leggono" lo stato di carica.
Il tempo d'accesso dei moderni hard disk sono dell'ordine dei millisecondi, mentre la capacità supera abbondantemente gli 80 GB.
Nell'hard disk la memoria è organizzata in blocchi di dimensione fissa (512B, 1KB,2KB,..) indirizzabili direttamente. La lettura/scrittura del disco avviene sempre in blocchi, per risparmiare tempo (pensate al tempo perso se si dovesse leggere un byte per volta). Il disco è quindi suddiviso o formattato in blocchi.
Altri esempi di dispositivo di memoria secondaria sono nastri magnetici e i CD/DVD. I primi sono caratterizzati da un accesso sequenziale ai dati, e pertanto sono molto lenti. Vengono generalmente usati per i backup di grosse quantità di dati. I secondi, sono supporti "ottici" abbastanza simili agli hard disk: sono anch'essi organizzati in blocchi ma con la differenza che i dati sono scritti/letti tramite un laser. La capacità di un CD arriva a 700 MB, mentre quella di un DVD a 4.7 GB.

Il processore

Il processore (CPU - Central Processing Unit) è senz'altro la parte più importante e complessa di un calcolatore elettronico. Esso controlla il flusso dei dati in tutto il sistema ed esegue i programmi. E' possibile scomporlo in una **unità di controllo (CU - Control Unit)** e in una **unità aritmetico-logica (ALU - Arithmetic Logic Unit)**. L'unità di controllo gestisce la sequenza delle operazioni che il calcolatore deve compiere, l'ALU ha il compito di eseguirle. Di seguito sono brevemente descritte le componenti principali della CPU.

La Control Unit (CU)

Come già accennato, l'unità di controllo gestisce (controlla) la sequenza delle operazioni che il calcolatore deve compiere, alla velocità del clock (2, 3, 4 GHz). Tali operazioni sono:

- prelievo dalla memoria centrale della prossima istruzione da eseguire;
- prelievo degli gli operandi specificati nell'istruzione;
- esecuzione dell'istruzione.

Registri

Sono piccole unità di memoria (2, 4, 8 byte) con tempi di accesso molto più bassi rispetto alla memoria primaria. Ospitano le informazioni necessarie per eseguire l'istruzione corrente. Sono in numero limitato (10, 20, 64) e si dividono in registri speciali e generali.

Registri Speciali

Il Program Counter (PC)
Il nome è poco significativo. Infatti il PC contiene l'indirizzo in memoria centrale della prossima istruzione da eseguire. All'inizio dell'esecuzione di un programma viene caricato con l'indirizzo della prima istruzione di quel programma. Ad ogni istruzione eseguita il PC viene modificato per contenere l'indirizzo della istruzione successiva.
L'Instruction Register (IR)
Contiene l'istruzione correntemente in esecuzione. La CU legge l'istruzione contenuta nell'instruction register e la esegue.
IL Registro di stato (PS - Program status)
Descrive lo stato corrente della esecuzione segnala eventuali errori (ad esempio il risultato di un confronto o il riporto di una somma).
Registro Indirizzi Memoria (MAR – Memory Address Register)
Contiene l'indirizzo della cella da cui leggere o in cui scrivere un dato.

Registro dati Memoria (MDR – Memory Data Register)
Contiene il dato letto dalla memoria o da scrivere in memoria.

I registri generali

Sono generalmente indicati con lettere progressive (A, B, …) o da una numerazione progressiva (R0, R1, …). Il loro numero varia in base alla specifica CPU, da alcune unità ad alcune decine. Sono usati come memorie temporanee per contenere gli operandi delle istruzioni e i risultati parziali durante l'esecuzione delle istruzioni.

Unità Logico Aritmetica (Arithmetic Logic Unit - ALU)

Si occupa di eseguire le operazioni di tipo aritmetico/logico: somme, confronti, eccetera. Preleva gli operandi dai registri generali e deposita il risultato delle operazioni ancora nei registri generali.

Quando la CPU deve eseguire una sequenza di istruzioni (programma):

1. **Fase di fetch (caricamento).**
    a. L'unità di controllo (CU) scrive l'indirizzo della cella di memoria contenente la prima istruzione del programma nel registro indirizzi memoria (MAR).
    b. L'istruzione da eseguire è copiata nel registro dati memoria (MDR) che si interfaccia con la RAM e, successivamente, nel registro istruzione (IR).
    c. La CU modifica il contenuto del PC in modo che esso contenga l'indirizzo della prossima istruzione del programma.
2. **Fase di decodifica.** La CU esamina l'istruzione contenuta nell'IR e determina le operazioni da svolgere.
3. **Fase di esecuzione.** Le unità interessate all'esecuzione dell'istruzione vengono attivate. Eventualmente vengono caricati gli operandi necessari (come nella fase di fetch) e/o scritti i risultati dell'operazione.

I punti 1-3 si iterano fino al raggiungimento di un istruzione di stop (es. la fine del programma) o fino al verificarsi di qualche errore.

La memoria cache

E' il livello di memoria intermedio tra i registri e la RAM, utilizzata per memorizzare i dati usati più spesso senza doverli recuperare in memoria centrale, il cui accesso risulterebbe più lento. A seconda del processore, la memoria cache può essere di varie dimensioni. Valori tipici sono: 64 KB, 128 KB, 256 KB, 512KB, 1MB, 2MB.
La memoria cache influisce moltissimo sulle prestazioni e il costo della CPU (e quindi del computer). I computer attuali hanno spesso più livelli di cache, ad esempio di primo livello (direttamente all'interno della CPU) o di secondo livello (esterna alla CPU, ma più veloce della RAM).

Il set di istruzioni macchina

Esistono differenti tipi di processore (**RISC** - Reduced Instruction Set Computer, **CISC** - Complex Instruction Set Computer), ognuno dei quali è in grado di eseguire un certo insieme di istruzioni base o elementari, detto **set di istruzioni macchina,** generalmente in numero non superiore a100. Esempi di processori RISC sono gli IBM Power 4 (utilizzati su supercalcolatori) o i PoerPC (utilizzati nei PC Apple), mentre esempi di processori CISC sono i Pentium IV della Intel o gli Athlon dell'AMD.
I più comuni tipi di istruzioni sono i seguenti:
Aritmetico/Logico. Rientrano in questa categoria anche le operazioni logiche tipo AND e OR. Nei calcolatori più complessi possono includere operazioni in virgola mobile. In ogni caso sono istruzioni che utilizzano l'ALU (Arithmetic Logic Unit).

Confronto. Eseguono il confronto tra due dati, per es. per decidere se effettuare un salto ad un altro punto del programma. In pratica non producono un risultato vero e proprio e nemmeno effettuano un'azione: si limitano ad aggiornare solo alcuni bit di stato. Dalla lettura di questi bit si ricava l'esito del confronto.
Salto. Può essere assoluto o condizionato. Se è condizionato è preceduto da un'altra istruzione che verifica la condizione, di solito un'istruzione di confronto.
Salto a subroutine, ritorno da subroutine. Il salto a subroutine è un'istruzione che deve salvare tutte le informazioni correnti relative al programma in esecuzione ed andare ad eseguire un altro segmento di programma (subroutine). Quando incontra invece un'istruzione di ritorno da subroutine il calcolatore riprende ad eseguire il programma interrotto a partire dal punto (istruzione) dove lo aveva lasciato precedentemente.
Trasferimento dati. Si tratta per es. del caricamento di registri dalla memoria e viceversa (si parla di istruzioni LOAD per i trasferimenti da memoria a registro e di istruzioni STORE per i trasferimenti da registro a memoria), del trasferimento di dati da registro a registro.
Istruzioni di I/O. Sono istruzioni che servono a trasferire dati all'esterno o ad acquisire dati dall'esterno del calcolatore. Si possono realizzare in due modi:

(A) quando incontra un'istruzione di I/O, l'unità centrale informa le periferiche che intende interagire con dispositivi esterni e, fornendo l'indirizzo, indica anche con quale dispositivo intende interagire; in questo caso si utilizzano codici specifici e indirizzi particolari;
(B) in alternativa è possibile riservare una zona nello spazio di indirizzamento della memoria in modo che ogni volta che si va ad agire in questa zona (con un'istruzione di trasferimento dati con la memoria) in realtà si interagisce con l'esterno

L'Assembler

Le istruzioni che costituiscono il linguaggio macchina sono rappresentate da stringhe binarie, generalmente di lunghezza prefissata (32, 64 bit). Per scrivere un programma direttamente in questo linguaggio è necessario avere la possibilità di utilizzare un linguaggio dello stesso livello ma con istruzioni di tipo simbolico più maneggevoli per il programmatore: l'Assembler ha precisamente questo ruolo e le sue dichiarazioni sono in corrispondenza biunivoca con le istruzioni del linguaggio macchina.
Per alcuni linguaggi ad alto livello la traduzione può essere fatta non da un compilatore ma da un interprete: questo e' per es. il caso dei linguaggi BASIC e JAVA. La differenza tra questi due tipi di traduttori e' la seguente:

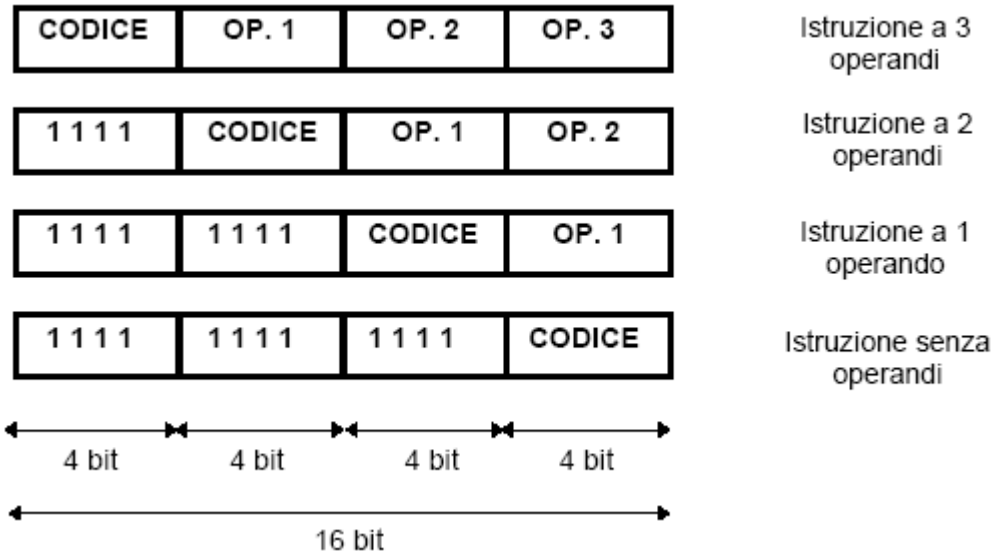
- **Compilatore.** Traduce l'intero programma sviluppato dall'utente (denominato **codice sorgente**) in Assembler e successivamente in linguaggio macchina o direttamente in linguaggio macchina lasciando "traccia" della traduzione, ossia producendo una copia fisica (in memoria) del programma tradotto (si tratta del cosiddetto **codice oggetto**). Se non si modifica il codice sorgente, ogni volta che si vuole eseguire il programma e' necessario avere a disposizione il solo codice oggetto (non serve piu' la traduzione).
- **Interprete.** E' un programma tramite il quale si esegue un altro programma. Esso preleva una dichiarazione alla volta, la esamina, la "interpreta" e la fa eseguire dal calcolatore, quindi passa alla successiva dichiarazione, senza lasciare "traccia" del programma tradotto. Ogni volta che si vuol fare eseguire il programma di partenza e' quindi necessario sia il programma sorgente che l'interprete.
- Se per uno stesso linguaggio sono disponibili entrambi i tipi di traduttori, ma non e' un caso frequente, si può osservare che l'interprete è generalmente più lento nel fare girare un programma, ma risulta più semplice da realizzare; il compilatore in compenso può produrre un codice oggetto di migliore qualità. Gli interpreti in generale hanno il vantaggio di rilevare subito eventuali errori e di poter eseguire lo stesso programma su macchine diverse.

Ogni istruzione macchina è costituita da un insieme di bit suddivisi solitamente in due parti: il **codice operativo** e gli **operandi.** Il codice operativo specifica il tipo di operazione che si vuole

eseguire sugli operandi il cui numero varia tipicamente da 0 a 3. Di seguito sono riportati esempi di istruzioni con i diversi numeri di operandi:

ADD R1, R2, R3 R1: destinazione R1, R3: sorgenti	Istruzione a 3 operandi che coinvolge 3 registri: in R1 si inserisce la somma dei contenuti di R2 ed R3; ossia: R1 ← [R2]+ [R3]
ADD R1, R2 R1: sorgente e destinazione R2: sorgente	Istruzione a 2 operandi che coinvolge 2 registri: in R1 si inserisce la somma dei contenuti di R1 ed R2; ossia: R1 ← [R1]+ [R2]
INC R3	Istruzione ad un operando: incrementa di una unità il contenuto del registro R3 ; ossia: R3 ← [R3] + 1
RETURN	Istruzione senza operandi (ritorno da subroutine)

Nelle istruzioni con 2 o 1 operando i registri possono essere sostituiti da indirizzi di memoria. Le istruzioni a 3 operandi sono tipiche delle macchine RISC, mentre quelle a 2 operandi sono tipiche delle macchine CISC.
Ogni istruzione che l'unità centrale deve eseguire viene divisa in più fasi. In generale si può al momento osservare che le unità centrali di tipo RISC hanno fasi molto semplici per ogni istruzione, mentre le CISC hanno fasi abbastanza complesse. Sarebbe inoltre opportuno che le istruzioni avessero tutte la stessa lunghezza (stesso numero di bit), per non complicare la struttura interna della CPU. In effetti le unità centrali RISC utilizzano istruzioni a lunghezza fissa (32 o 64 bit), mentre le unità centrali di tipo CISC, soprattutto per i vincoli posti dalla necessità di preservare la compatibilità con calcolatori precedenti, utilizzano istruzioni a lunghezza variabile (16/32/64).
Se si intendessero utilizzare istruzioni a lunghezza fissa, ad esempio di 16 bit, ma con numero di operandi variabili, è possibile utilizzare una codifica a **espansione del codice operativo** che sfrutta al meglio i bit disponibili.



Come si ricava dalla figura:

- Per le istruzioni a 3 operandi esistono 15 codici operativi possibili: infatti si possono utilizzare i codici da 0000 a 1110 mentre il codice 1111 non può essere usato perchè, nel

caso di istruzioni con meno di 3 operandi i primi 4 bit valgono 1111 e i secondi 4 diventano una prosecuzione del codice operativo;
- Per le istruzioni a 2 operandi il codice operativo e' costituito dai primi 8 bit: sono pertanto 15 i codici possibili, da 1111 0000 a 1111 1110, e la configurazione 1111 dei secondi 4 bit serve per espandere ulteriormente il codice operativo;
- Per le istruzioni a 1 operando il codice operativo e' costituito dai primi 12 bit: 15 codici sono i codici possibili, da 1111 1111 0000 a 1111 1111 1110, sempre per consentire la successiva espansione;
- Per le istruzioni senza operandi il codice operativo e' costituito da tutti e 16 i bit: 16 sono i codici possibili, da 1111 1111 1111 0000 a 1111 1111 1111 1111.

Complessivamente si arriva a 61 istruzioni (15+15+15+16).

Altri componenti hardware

Altri componenti hardware sono la scheda madre e la scheda video. La scheda madre ospita molti componenti hardware come la CPU e la memoria principale. Inoltre ospita il BUS di sistema, tramite il quale possono avvenire gli scambi di dati tra processore, memoria principale e secondaria. La scheda video si occupa, invece, della gestione del segnale video e della sua rappresentazione.

Prestazioni

Esistono vari criteri: velocità di calcolo, affidabilità del sistema, fiducia nel rivenditore, assistenza tecnica, ecc. Tuttavia, il criterio più utilizzato è il tempo di CPU.
Il tempo di CPU di un processo è espresso dal prodotto del numero di cicli di clock necessari per il suo completamento per il periodo di clock:

$$T_{CPU} = n_{CK} \cdot T_{CK}$$

La frequenza di clock è definita come $f_{CK} = 1/ T_{CK}$, da cui: $T_{CK} = 1/ f_{CK}$
Il tempo di CPU può essere allora definito come:

$$T_{CPU} = n_{CK} / f_{CK}$$

Con *CPI* si indica il numero medio di cicli necessari a eseguire una data istruzione in linguaggio macchina tra quelle che compongono il programma. Se n_I rappresenta il numero di istruzioni, allora:

$$CPI = n_{CK} / n_I \Rightarrow n_{CK} = CPI \cdot n_I$$

Il tempo di CPU può essere allora riscritto come:

$$T_{CPU} = CPI \cdot n_I \cdot T_{CK}$$

oppure

$$\boldsymbol{T_{CPU} = CPI \cdot n_I / f_{CK}}$$

Questa espressione del tempo di CPU evidenzia come le prestazioni dipendano dai seguenti fattori:
1. la frequenza di clock cui funzionano le CPU (f_{CK});
2. il numero di cicli di clock necessari in media per completare un'istruzione (*CPI*);
3. il numero di istruzioni macchina che devono essere eseguite per completare un processo (n_I).

La frequenza di clock a cui funziona la CPU dipende principalmente dalla tecnologia adottata per la realizzazione dei circuiti e dall'architettura della CPU stessa. Più grande è la frequenza, minore risulta il tempo di CPU. Tuttavia, bisogna tenere in considerazione anche i termini *CPI* ed n_I.
I processori CISC sono basati sull'idea di implementare a livello hardware funzioni sempre più complesse. Questo permette di ridurre il numero di istruzioni necessarie al completamento del processo, n_I, ma al contempo il tempo medio d'esecuzione, *CPI*, cresce. Al contrario, i processori RISC sono basati sull'idea di implementare a livello hardware funzioni semplici, a fronte di un numero maggiore di istruzioni necessarie al completamento del processo, n_I, ma più efficienti in termini di tempo medio d'esecuzione, *CPI*.
Le due filosofie sono entrambe valide. Tuttavia, quando si scrive un programma in un linguaggio ad alto livello, il compilatore (o l'interprete) si occupa della sua traduzione in linguaggio machina, che

sarà poi eseguito dalla CPU. Alcuni studi condotti verso la fine degli anni 80 hanno evidenziato che anche i migliori compilatori e interpreti non erano in grado di sfruttare al meglio istruzioni complesse tipiche dei processori CISC. Al contrario, si è dimostrato più semplice realizzare un buon codice macchina sulla base di istruzioni semplici come quelle dei processori RISC. Per tale motivo i processori RISC sono ritenuti, a parità di clock, più veloci dei processori CISC e, per questo motivo hanno trovato larga diffusione su macchine dedicate espressamente al calcolo scientifico. L'architettura CISC sopravvive essenzialmente nel mercato dei personal computers per questioni legate alla compatibilità delle applicazioni.
Infine, per quanto riguarda i supercalcolatori, un'altra misura è utilizzata diffusamente per indicare la potenza di calcolo: il **MFLOPS** – Millions of FLOating point operations Per Seconds (milioni di operazioni in virgola mobile al secondo). In questi sistemi, infatti, i programmi di calcolo operano pesantemente su numeri frazionari. Il computer più veloce della nostra università ha una potenza di ben 160000 MFlops = 160 GFlops (160 miliardi di operazioni in virgola mobile al secondo).

Il software di base

Come abbiamo visto è possibile specificare sequenze di comandi (programmi) che il calcolatore può eseguire. Tali programmi, scritti in un qualsiasi linguaggio di programmazione, o direttamente in linguaggio macchina, costituiscono il software. A differenza dei primi elaboratori elettronici, i moderni computer sono dotati di software di base che si occupano della gestione dell'hardware e facilitano l'interazione uomo macchina.
Il software di base comprende il Sistema Operativo e, in alcuni casi, uno o più compilatori ad alto livello. Un generico sistema operativo moderno si compone di alcune parti ben definite: un gestore di file system che si occupa di esaudire le richieste di accesso alle memorie di massa, un gestore di memoria virtuale che alloca pagine di memoria a richiesta e si assicura che questa sia presente nella memoria fisica al momento giusto, uno scheduler che assicura ai vari processi in esecuzione una ben definita quantità di tempo di elaborazione, uno spooler che accumula i dati da stampare e li stampa in successione, una interfaccia utente (shell o GUI) che permette agli esseri umani di interagire con la macchina ed un kernel, fulcro del sistema, che gestisce il tutto. A seconda dei casi, un particolare sistema operativo può avere tutti questi componenti o solo alcuni. Vediamo ora una serie di sistemi operativi possibili, dal più semplice al più complesso.

Monitor

Praticamente il solo kernel, con una minima interfaccia interattiva per impartire i comandi. Permette di scrivere in memoria il programma da eseguire e di lanciarlo, non ha nessuna altra caratteristica. È semplicissimo (per un computer), spesso i suoi comandi sono semplici chiamate dirette a subroutine in linguaggio macchina, è stato anche il primo tipo di sistema operativo mai implementato su un computer.

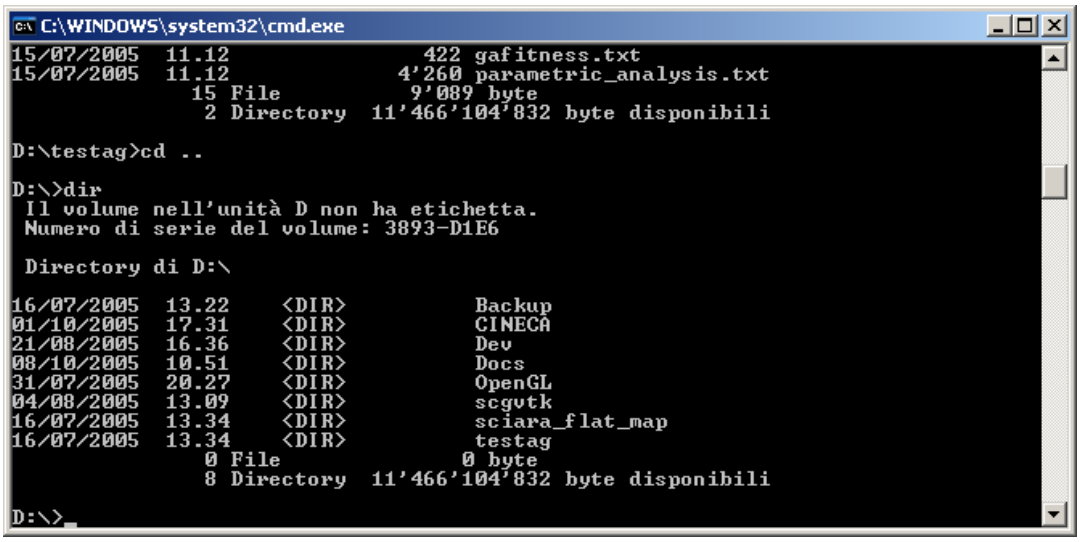
Interprete

Il secondo passo verso una migliore gestione del computer si ha con lo sviluppo di una interfaccia utente separata dal kernel, un interprete di comandi che funga anche da interfaccia utente, da shell. Questa shell primitiva di solito funge anche da interprete per un linguaggio di programmazione: a seconda delle scelte dei progettisti del software può essere un vero linguaggio oppure un più semplice linguaggio di scripting con cui creare comandi batch.

DOS

Un computer diventa molto più utile se dotato di una memoria di massa: per gestirla serve un gestore di file system, cioè un insieme di funzioni che permetta di organizzare i dati sulla superficie dei mezzi di memorizzazione secondo una struttura ben precisa. I sistemi operativi capaci di gestire un file system sono detti genericamente Disk Operating Systems, cioè DOS appunto. L'esemplare

più famoso è senz'altro il MS-DOS della Microsoft. Esiste anche una versione libera del dos, denominata: FreeDOS.



Sistema multitask

Alcuni programmi non hanno sempre realmente bisogno della CPU: a volte, invece di eseguire istruzioni stanno aspettando che arrivino dei dati da un file, o che l'utente prema un tasto alla tastiera. Quindi si può, in linea di principio, usare questi tempi "morti" per far girare un altro programma. Questa idea, sorta fin dai primi anni 50, si concretizzò nei sistemi operativi multitasking, cioè dotati di uno scheduler che manda in esecuzione più processi (esecuzioni di programmi) contemporaneamente, assegnando a turno la CPU ad ognuno e sospendendo l'esecuzione dei programmi in attesa di un evento esterno (lettura sulla/dalla memoria di massa, stampa, input utente ecc.) finché questo non si verifica.
Dovendo ospitare in memoria centrale più programmi nello stesso tempo, i sistemi multitask hanno bisogno di più memoria rispetto a quelli monotask: questo porta questo tipo di sistemi operativi a fare quasi sempre uso di un gestore di memoria virtuale.

Sistema multiutente

Se un computer può far girare più programmi contemporaneamente, allora può anche accettare comandi da più utenti contemporaneamente: in effetti dal multitasking alla multiutenza il passo è molto breve tecnicamente, ma fa sorgere una serie di nuovi problemi dal punto di vista della sicurezza del sistema: come distinguere i vari utenti tra loro, come accertarsi che nessun utente possa causare danni agli altri o alla macchina che sta usando ecc.
Questi problemi si risolvono assegnando un account univoco per ogni utente, assegnando un proprietario ai file ed ai programmi e gestendo un sistema di permessi per l'accesso ad essi, e prevedendo una gerarchia di utenti (cioè di account) per cui il sistema rifiuterà tutti i comandi potenzialmente "pericolosi" e li accetterà soltanto se impartiti da un utente in cima alla gerarchia, che è l'amministratore del sistema (generalmente l'account root nei sistemi Unix, Administrator nei sistemi Windows).

Parti del sistema operativo

Kernel

Il kernel è il cuore di un sistema operativo. Si tratta di un software con il compito di fornire ai programmi in esecuzione sul computer e agli altri moduli componenti il sistema operativo le funzioni fondamentali ed un accesso controllato all'hardware. Quali funzioni sia opportuno che il

kernel debba fornire e quali possano essere demandate a moduli esterni è oggetto di opinioni divergenti: se il kernel di un sistema operativo implementa soltanto un numero molto ristretto di funzioni, delegando il resto ad altre parti, si parla di microkernel. Il vantaggio di un sistema operativo microkernel è la semplicità del suo kernel; lo svantaggio è l'interazione più complessa fra il kernel e le altre componenti del S.O. stesso, che rallenta il sistema.

File system

Il file system è il modo in cui i file sono immagazzinati e organizzati su un dispositivo di archiviazione, come un hard disk o un CD-ROM. Esistono molti tipi di file system, creati per diversi sistemi operativi, per diverse unità di memorizzazione e per diversi usi. Esistono due grandi classi di file system: quelli per unità locali, destinate ad organizzare fisicamente i dati su un disco, e i file system distribuiti, nati per condividere i dati fra più computer collegati attraverso una rete, superando le differenze fra sistemi operativi e filesystem locali delle varie macchine.

Filesystem per unità locali

* Amiga FileSystem
* CFS
* Ext2
* Ext3
* FAT
* HFS
* HPFS
* ISO 9660
* JFS
* Minix
* NTFS
* ReiserFS
* UFS
* XFS

Filesystem distribuiti

* Nfs
* Coda
* Andrew

Scheduler

Lo scheduler è un componente fondamentale dei sistemi operativi multitasking, cioè quelli in grado di eseguire più processi (task) contemporaneamente. Lo scheduler si occupa di fare avanzare un processo interrompendone temporaneamente un altro, realizzando così un cambiamento di contesto (context switch). Generalmente computer con un processore sono in grado di eseguire un programma per volta, quindi per poter far convivere più task è necessario usare lo scheduler. Esistono vari algoritmi di scheduling che permettono di scegliere nella maniera più efficiente possibile quale task far proseguire.

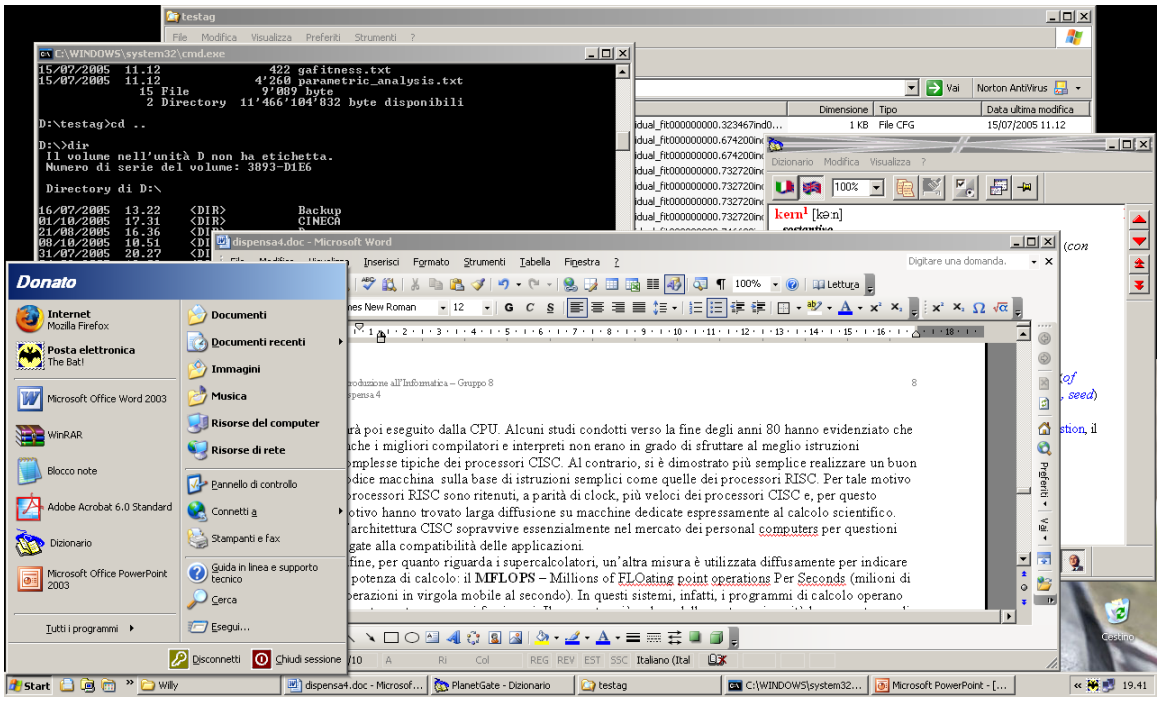
Gestore di memoria virtuale

Il gestore di memoria virtuale è la componente del sistema operativo che si occupa di gestire ed assegnare la memoria ad un processo che ne fa richiesta. Questo è necessario in quanto gli OS che utilizzano un gestore di memoria virtuale forniscono ai processi un'astrazione della memoria realmente presente.

Interfaccia utente

L'interfaccia grafica (in inglese Graphical User Interface, abbrev. GUI) è un paradigma di sviluppo che mira a consentire all'utente di interagire col calcolatore manipolando graficamente degli oggetti,

svincolandolo dal dover imparare una serie di comandi da impartire con la tastiera come invece avviene con le interfacce testuali CLI (Command Line Interface). È lo strato di un'applicazione software che si occupa del dialogo con l'utente del sistema utilizzando un ambiente grafico.
L'interfaccia grafica nei sistemi operativi moderni è concepita come la metafora di un piano di lavoro rappresentato dallo schermo (detto scrivania o desktop), con le icone a rappresentare i file (di cui alcune a forma di cartellina per le directory) e le finestre a rappresentare le applicazioni.
Tale ambiente di lavoro, in cui si opera attraverso il puntatore comandato con il mouse, è stato concettualizzato nei laboratori Xerox (progetto Alto) e implementato (in bianco e nero) per la prima volta da Apple con il suo rivoluzionario personal computer Macintosh nel 1984. La prima versione a colori della GUI venne introdotta da Commodore con il suo Amiga nel 1985.
La GUI, con la sua semplicità d'utilizzo, per anni ha contrastato le interfaccie ostiche e complicate basate ancora sul vecchio principio della linea di comando (Command Line Interface), come UNIX e DOS. In seguito al successo del Macintosh e dell'Amiga, queste caratteristiche innovative sono state mutuate da Microsoft con la creazione del proprio sistema operativo Windows.
Attualmente tutti i sistemi operativi diffusi nel settore dei personal computer sono dotati di una GUI che opera secondo gli stessi principi di quella originariamente studiata da Xerox. Ciò ha causato una evoluzione significativa nell'interazione tra computer e utente: grazie all'interfaccia grafica è possibile compiere molti compiti comuni e complessi senza il bisogno di un'approfondita conoscenza del funzionamento del computer.



Spooler di stampa

Lo spooler di stampa è stato, storicamente, uno dei primi moduli esterni del sistema operativo ad essere implementato, per risolvere il problema della gestione delle stampe su carta.
Infatti, essendo le stampanti elettromeccaniche dei dispositivi molto lenti, i primi programmi per elaboratore dovevano necessariamente sprecare molto tempo di CPU, estremamente prezioso all'epoca, per controllare la stampante ed inviarle i dati. Quindi venne ideato un programma separato, che girava con una priorità molto bassa e che era visto dagli altri programmi come una normale stampante: in realtà invece lo spooler accumulava i dati che un programma doveva stampare in una apposita area di memoria RAM, e poi si faceva carico del processo di stampa vero e proprio lasciando gli altri programmi liberi di continuare la loro esecuzione.

Il meccanismo fondamentale dello spooler di stampa è rimasto sostanzialmente invariato dai suoi albori fino ad oggi: con gli anni e con il progredire della tecnologia le modifiche più rilevanti sono state la capacità di gestire più stampanti selezionabili a piacere, e la capacità di gestire anche stampanti remote, collegate cioè non direttamente al computer su cui gira lo spooler ma ad altri elaboratori connessi via rete.