Lezione 2 (18-02-25)

Storage Structure

All'interno di un calcolatore abbiamo:

- **memoria centrale**: unica memoria accessibile tramite CPU. È volatile.
- **memoria secondaria**: estensione della memoria centrale. Non è volatile.
- **hard disk drives (HDD)**: rigidi di metallo o in fibra di vetro. Più lenti della RAM.
- **non-volatile memory (NVM)**: memorie non volatili, prive di organi meccanici, più veloci degli HDD, sempre più diffusi.

La CPU carica le istruzioni esclusivamente dalla memoria, perciò tutti i programmi da eseguire devono esservi caricati.
In generale, i computer eseguono la maggior parte dei programmi da una memoria riscrivibile, ovvero la **memoria centrale** detta anche memoria ad accesso casuale (*random access memory* RAM)
La **memoria centrale** è realizzata tipicamente con una tecnologia basata sui semiconduttori, detta anche memoria *dinamica* ad accesso casuale (*dynamic random access memory* DRAM).

Vengono utilizzati anche altri tipi di memoria, infatti dal momento che la RAM è volatile, ovvero perde il suo contenuto quando la corrente viene a mancare, non può essere utilizzata per il programma di *bootstrap*.

Bootstrap :=
È il primo programma eseguito all'avvio del computer. Si trova nel firmware (BIOS o UEFI) ed è memorizzato su una ROM della scheda madre. Quando il computer viene alimentato, la CPU inizia a eseguire istruzioni da un indirizzo di memoria fisso, che contengono il codice necessario per inizializzare il sistema e avviare il sistema operativo.

Idealmente vorremmo che i programmi e i dati gestiti da essi risiedessero nella memoria centrale, ma questo non è possibile in quanto:

- la capacità della memoria centrale non è sufficiente a contenere in modo permanente tutti i programmi e i dati richiesti;
- la memoria centrale è un dispositivo di memorizzazione volatile, perde il proprio contenuto una volta interrotta l'alimentazione.
  Perciò viene prevista la presenza di una **memoria secondaria** come estensione della memoria centrale; ha come caratteristica fondamentale quella di conservare in modo permanente grandi quantità di dati.
  I dispositivi più utilizzati a questo scopo sono:
- **hard-disk drive** (*hdd*), disco magnetico
- dispositivi **non-volatile memory** (*nvm*)
  La maggior parte dei programmi è mantenuta in un disco sino al momento del caricamento nella

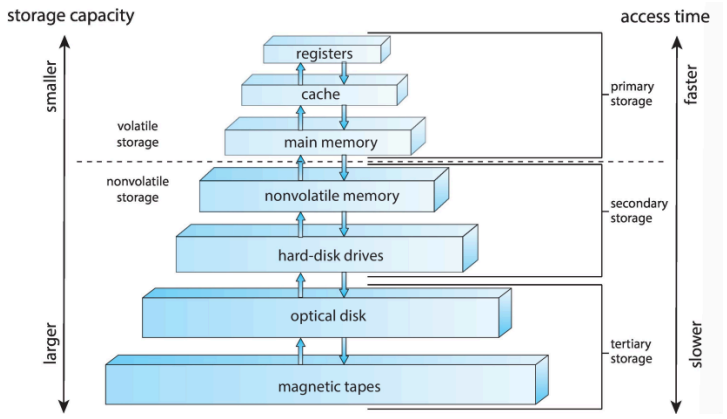
memoria. Molti programmi utilizzano il disco come sorgente e destinazione delle informazioni elaborate.

Esistono altri tipi di memorie:

- memorie cache
- cd-rom
- nastri magnetici (memoria di archiviazione terziaria o di backup)
- ...
  Le caratteristiche che differenziano i sistemi di memorizzazione sono: velocità, dimensioni e volatilità.

Gerarchia di memoria

I sistemi di memorizzazione possono essere ordinati seconda una scala gerarchica, in base alle **capacità** e al **tempo d'accesso**. Inoltre si differenziano in **volatili** e **non volatili**.
La memoria *volatile* perde il suo contenuto quando viene tolta l'alimentazione; quindi i dati vengono scritti su una memoria *non volatile* quando devono essere memorizzati in maniera sicura.
Caching :=
operazione che sposta dalla memoria più lenta a quella più veloce un po' di dati. Carico i dati che uso più spesso → ottimizzo prestazioni del computer, fondamentale per il calcolatore.
Ogni tipologia di memoria ha il suo driver per gestire il caching.
Device Driver := provvede ad interfacciare il dispositivo di I/O con il kernel



La memoria più veloce (e costosa) (statica) è quella utilizzata per implementare i registri all'interno del processore.
Successivamente, troviamo la cache.

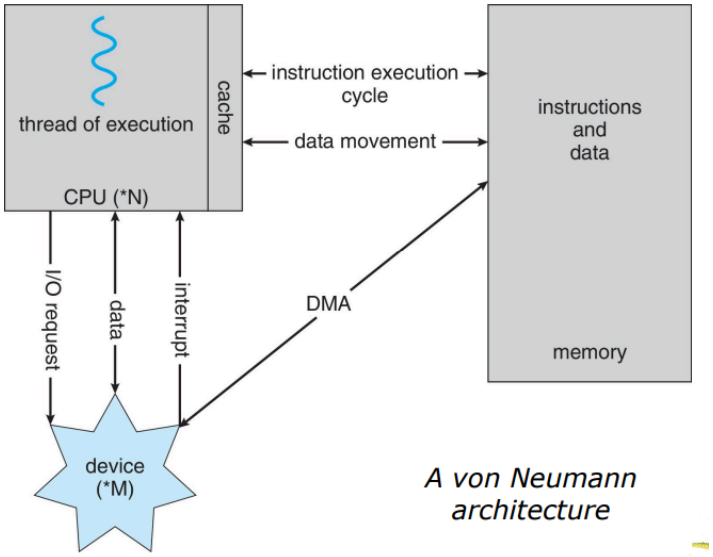
I primi quattro livelli sono **memorie a semiconduttore** (ovvero, sono composte da circuiti elettronici basati su semiconduttori).
I dispositivi NVM (quarto livello), hanno diverse varianti, ma sono in generale più veloci dei dischi magnetici (es. memoria flash).
Man mano che scendiamo i livelli, le memorie diventano meno veloci ma più grandi.
Ad esempio, il tempo di accesso alla RAM può essere decine di cicli di clock, mentre il tempo di accesso ad un registro della CPU è un ciclo di clock.

Se accedo a una cella di memoria, è probabile che accederò a celle vicine (**principio di località spaziale**). Analogamente, se ho recentemente scritto o letto da una cella, è probabile che accederò nuovamente alla stessa o a celle vicine (**principio di località temporale**).

Tipicamente la CPU ha più livelli di cache (L1, L2, L3) che precedono la memoria centrale, organizzate in una struttura gerarchica per ottimizzare l'acceso ai dati. La CPU si "rivolge" alla memoria per ottenere i codici operativi delle istruzioni macchina da eseguire e i dati su cui devono lavorare queste istruzioni.
I dispositivi di I/O prelevano/trasmetto dati dal/al mondo esterno, la CPU indirizza delle richieste di input/output ai dispositivi, scambia dati con i dispositivi e questo processo e guidato da interrupt

Struttura di I/O

Una percentuale del codice di un SO è dedicata alla gestione dell'I/O. L'I/O guidato dalle interruzioni è adatto al trasferimento di piccole quantità di dati, ma in caso di trasferimenti molto più grandi può generare un sovraccarico (*overhead*), per risolvere questo problema si utilizza la tecnica dell'accesso diretto alla memoria (DMA). Una volta impostati i buffer, i puntatori e i contatori necessari al dispositivo I/O, il controller trasferisce un intero blocco di dati dal proprio buffer direttamente nella memoria centrale, o viceversa, senza interventi da parte della CPU. In questo modo l'operazione richiede una sola interruzione per ogni blocco di dati trasferito, piuttosto che per ogni byte.



Operating-System Operations

L'avviamento del sistema, così come il riavvio, richiede la presenza del programma di avviamento (*bootstrap program*). Normalmente è memorizzato nel *firmware* che fa parte dell'hardware di un calcolatore. Il programma di avviamento ha lo scopo di inizializzare i componenti del sistema, dai registri della CPU ai controller dei dispositivi, fino al contenuto della memoria centrale; inoltre, deve avviare il sistema operativo e avviarne l'esecuzione: perciò deve individuare e caricare nella memoria il kernel del sistema operativo. Una volta caricato e in esecuzione, il kernel può iniziare a offrire servizi al sistema e agli utenti.
Alcuni servizi vengono forniti all'esterno del kernel da programmi di sistema caricati in memoria durante l'avviamento. Questi *processi di sistema*, anche detti **demoni**, restano in esecuzione per tutto il tempo in cui è in esecuzione il kernel.
(es. in Linux il primo demone è `systemd`, che avvia diversi altri demoni).
Se i dispositivi I/O non richiedono alcun servizio, non ci sono processi da eseguire o altro, il sistema rimane in attesa di un evento. Un evento è quasi sempre segnalato da un'interruzione:

- interruzioni hardware ⇒ *interrupt*;
- interruzioni software ⇒ *trap* o *exception* (es. divisione per zero)

Multiprogramming

In generale, un singolo programma non è in grado di tenere costantemente occupata la CPU e i dispositivi di I/O; inoltre, vorremmo fosse possibile eseguire più comandi contemporaneamente. A questo proposito esiste la **multiprogrammazione** che consente di aumentare la percentuale di utilizzo della CPU, organizzando i programmi in modo tale da mantenerla in continua attività. In un sistema multiprogrammato un programma in esecuzione è chiamato **processo**.

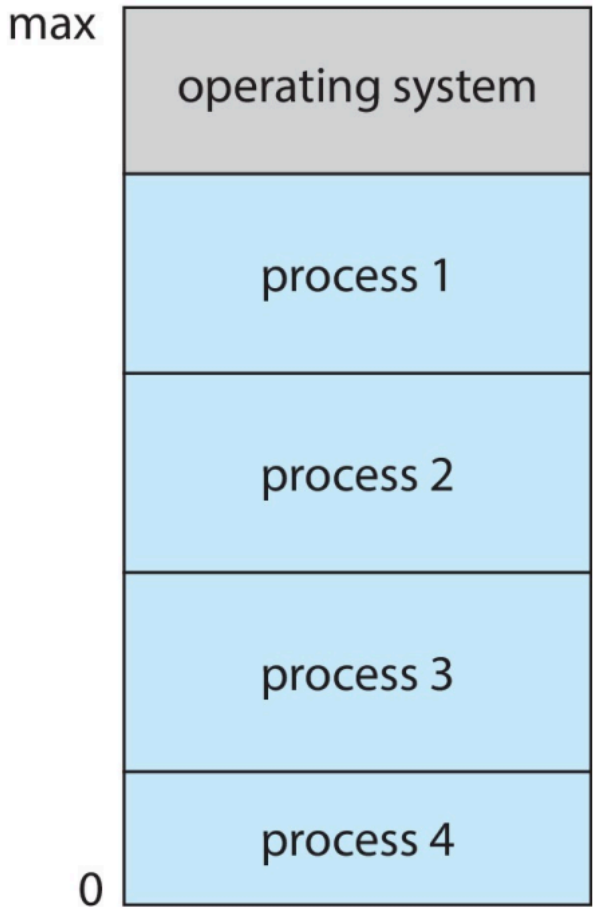
Multiprogrammazione := capacità di un sistema di elaborazione di eseguire più programmi contemporaneamente, in modo da tenere la CPU occupata.
Processo := istanza di un programma in esecuzione

Il sistema operativo mantiene in memoria centrale diversi processi e ne esegue uno alla volta. Se un processo entra in stato di attesa (ad esempio per un'operazione di I/O), il sistema operativo passa all'esecuzione di un altro processo per ottimizzare l'uso della CPU. Quando il primo processo ha terminato l'attesa, la CPU ne riprende l'esecuzione. Finché c'è almeno un processo da eseguire, la CPU non resta mai inattiva.

Multitasking

Il multitasking è un'estensione logica della multiprogrammazione; la CPU esegue più lavori commutando le loro esecuzioni con una frequenza tale da permettere a ciascun utente di usufruire di tempi di risposta rapidi.
Talvolta i processi possono richiedere operazioni *interattive*, la velocità delle operazioni di immissione dipendono dagli esseri umani (es. digitare caratteri sulla tastiera), nel frattempo che queste operazioni vengono eseguite, la CPU non è inattiva, il sistema operativo commuta rapidamente la CPU a un altro processo.
La coesistenza di programmi in memoria, nello stesso lasso di tempo, richiede una gestione corretta della memoria; inoltre, l'esistenza di diversi processi pronti per l'esecuzione nello stesso istante impone al sistema di scegliere quale processo viene eseguito per primo, il modo in cui questo processo viene scelto viene detto **scheduling della CPU**.
In caso di processi che devono essere eseguiti raramente, è comodo avere in esecuzione un insieme di processi che hanno una necessità di memoria complessiva maggiore di quella della memoria principale, soluzione:

- **swapping**: quando un processo non può/non ha bisogno di essere eseguito, per fare spazio ad un altro processo la sua immagine viene copiata sul disco (memoria secondaria) per fare spazio al processo che deve essere eseguito; ha svantaggia e non è quasi più in uso;
- **virtual memory**: la memora è divisa in pagine di dimensione fissa (4kb, 16kb), le pagine vengo trasferite dalla memoria principale al momento del bisogno; tutti i processi hanno un po' di pagine sul disco (quelle poco usate) e un po' di pagine sulla RAM;
  In un sistema multitasking, il sistema operativo deve garantire tempi di risposta accettabili. Un metodo comune per ottenere questo risultato è la **memoria virtuale**, che consente l'esecuzione di programmi anche non interamente caricati nella memoria.



Rappresenta una memoria principale dove: una parte è riservata per il sistema operativo e una parte viene divisa tra i vari processi (programmi in esecuzioni).