**Comparazione tra computer classici e computer quantistici**

Marco Agostino Palazzo

Anno Accademico

2023/2024

Relatore Tesi :Raffaella Radogna

**Indice**

1.Introduzione

2.Computer Classico

3.Componenti computer classico

3.1 Ram

3.2 HardDisk/SSD

3.3 Cache

3.4 Cpu

3.5 Gpu

3.6 Bus

3.7 Periferiche

3.8 Scheda Madre

3.9 ROM

4.Funzionamento complessivo componenti computer classico

5. Introduzione meccanica quantistica

6.Computer quantistico

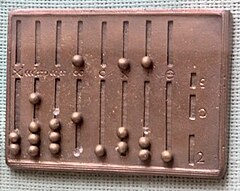
7.Componenti computer quantistico

8.Funzionamento complessivo componenti computer quantistico

**Capitolo 1**

Introduzione

Il primo computer di cui si hanno notizie risale all’era dei Babilonesi, i quali inventarono l’abaco, strumento che ha aperto la strada a una marea di altre invenzioni tecnologiche.

Il signor Gottfried Wihlelm Leibnitz, è un esempio lampante: inventò il sistema binario secoli e secoli dopo i Babilonesi, dando praticamente vita alla base della programmazione informatica odierna! 

La vera svolta nello sviluppo dei computer, tuttavia, arriva nemmeno 100 anni fa, durante la **Seconda Guerra Mondiale**.

Nel 1943, i nazisti, per comunicare, utilizzano messaggi criptati.

Il governo britannico, interessato al contenuto di quei messaggi, assolda **Alan Turing**, il quale crea il Colossus, il primo Computer digitale interamente elettronico e in grado di decifrare i messaggi dei tedeschi.

Turing fu una delle tante menti che contribuì a sviluppare macchine sempre più sofisticate, capaci di eseguire calcoli sempre più complessi sulla base della loro programmazione.

È sua la testa dietro al Test di Turing, un test volto a capire e valutare l’intelligenza delle macchine.

Facciamo un salto in avanti di più di 30 anni, e arriviamo al 1976. Due giovani , chiusi nel loro garage , costruiscono un microcomputer. I due ragazzi si chiamano **Steve Jobs e Steve Wozniak** e nel corso dello stesso anno fondano quella che sarebbe diventata una delle aziende più famose al mondo: la **Apple.**

Un anno dopo, migliorando il precedente modello, creano l’Apple II, probabilmente il primo computer davvero user friendly. Ma la concorrenza non tarda a fare la sua comparsa.

Nel 1981 **Bill Gates e Paul Allen** fondano la Microsoft e creano il sistema operativo MS-DOS, praticamente un semplice prompt di comandi con sfondo nero e linee di codice bianche.

L**’International Business Machines Corporation**, una delle aziende più antiche nel settore informatico, nel corso dello stesso anno, fiuta l’affare e produce il suo primo Personal Computer con sistema operativo MS-DOS.

Nel 1985 Microsoft rilascia Windows 1.0, la prima versione del Sistema Operativo più utilizzato al mondo e installato su tutti i computer delle nostre scuole.

Dieci anni dopo, nel 1995, Windows rilascia il capostipite dei sistemi operativi odierni, ovvero **Windows 95**, contenente per la prima volta nella storia Internet Explorer, il primo browser (insieme a Netscape Navigator) di sempre. Nasce così la moderna era di Internet.

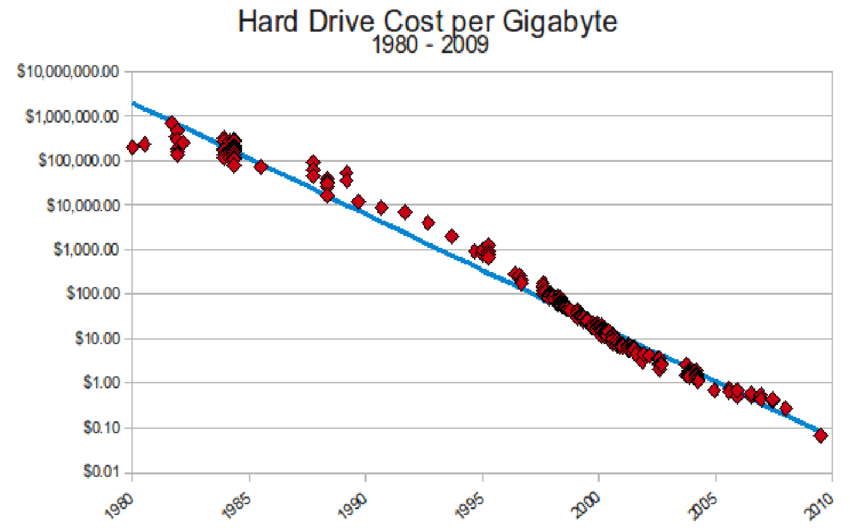
**Capitolo 2**

Computer Classici

Dal 1965 ad oggi i computer si sono evoluti in maniera esponenziale , questo viene spiegato perfettamente da Moore con la sua prima legge che qualsiasi informatico conosce.

**Prima legge di Moore** :

«La complessità di un microcircuito, misurata ad esempio tramite il numero di transistor per chip, raddoppia ogni 18 mesi (e quadruplica quindi ogni 3 anni).»

Un facilissimo esempio che spiega questa legge sono gli HardDisk(Spiegazione Capitolo 3).Essi vennero inventati nel giugno del 1956 , e come si evince dall’immagine il loro costo con il passare del tempo è diminuito in maniera esponenziale.

Questo grazie alla legge di Moore, in quanto ogni anno venivano rilasciati Hd sempre piu’ prestanti (a livello di throughput e di gb/componente),facendo diminuire il prezzo a quelli usciti precedentemente .

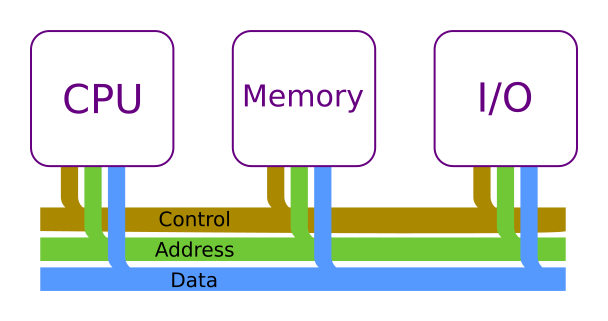
Se si parla di architettura di computer digitali, si pensa subito all’Architettura di von Neuman.

Essa consiste in tre principali elementi :

-Cpu(Control Process Unit)

-Ram(Random Access Memory)

-Bus(Binary Unit System)



Questi tre elementi compongono le parti essenziali di un qualsiasi computer digitale.

La cpu fetcha le istruzioni ed i dati dalla memoria, decodifica dati ed istruzioni, ed esegue l’istruzione sui dati inerenti a quell’istruzione, riportando il risultato in memoria.

Tutti questi passaggi di dati avvengono tramite i Bus.(Approfondimento Processo Capitolo 4)

Come ben sappiamo, oggi un computer ha ben altre componenti , come un’HardDisk(recentemente sostituiti dalle SSD),schede grafiche, periferiche di ogni genere(tastiere,mouse,webcam, tavolette grafiche etc...).

L’unità fondamentale con cui il computer comunica con le sue componenti è il bit.

Il bit non è altro che un’impulso elettrico , che associa uno zero, ed un’1 in base all’assenza o alla presenza di un segnale elettrico.

Da non confondere bit con byte.

I primi come già spiegato, possono essere rappresentati da uno zero o da un’1, mentre i byte, sono una sequenza di 8 bit .

Essendo composti da 8 bit, rendendo possibile avere 256 possibili configurazioni per quel byte.

Queste due misure vengono confuse molto spesso, un’esempio classico è quando andiamo a leggere la “velocità ” di una connessione internet.

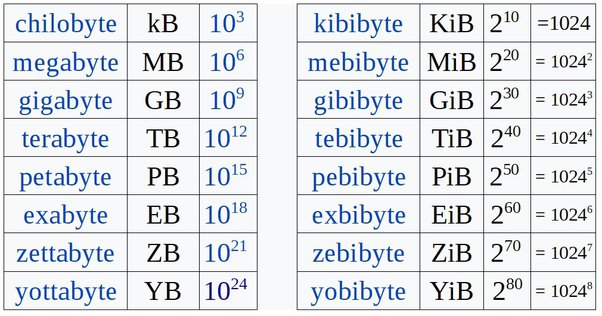
A volte può essere riportata in Mbs/s altre Mb/s.

La differenza???

La prima rappresenta i bit , la seconda i byte, dunque se dovessi leggere una connessione internet da 256 Mbs/s e una da 256 Mb/s, quale sarebbe la piu’ conveniente?

La seconda ovviamente, in quanto la prima 256b Mbs/s , convertiti in MegaByte da MegaBits, sarebbe di 2Mb/s.

I Byte vengono suddivisi in base all’esponente di base due multiplo di 10.



Il computer dunque comunica attraverso bit e byte, ma come fa a capire i numeri in diverse basi e tutto il resto dei dati?

Questo avviene tramite un processo iterativo attraverso l’uso di diversi linguaggi situati su diversi livelli.

Il livello più alto, è quello attraverso il quale noi diciamo al computer cosa fare, e nell’eventualità con quali dati.(Livello linguaggi di programmazione)

Queste informazioni, in linguaggi di alto livello , vengono Compilate , e vengono trasformate in linguaggio macchina (Livello Assembler).

Successivamente , questi dati in linguaggio macchina vengono a loro volte tradotti in linguaggio binario, ovvero il nostro adorato bit.

Per compilatore e Interprete di intende un tipo particolare di esecuzione di un programma partendo da un linguaggio di alto livello.

Il compilatore suddivide il suo processo in varie fasi.

* Analisi lessicale:Attraverso un analizzatore lessicale, spesso chiamato *scanner* o *lexer*, il compilatore divide il codice sorgente in tanti pezzetti chiamati token.I token sono gli elementi minimi (non ulteriormente divisibili) di un linguaggio, ad esempio parole chiave (*for*, *while*), nomi di variabili (*pippo*), operatori (*+*, *-*, *«*).
* Analisi sintattica o parser : L'analisi sintattica prende in ingresso la sequenza di token generata nella fase precedente ed esegue il controllo sintattico. Il controllo sintattico è effettuato attraverso una grammatica. Il risultato di questa fase è un albero di sintassi.
* Analisi semantica: L'analisi semantica si occupa di controllare il significato delle istruzioni presenti nel codice in ingresso. Controlli tipici di questa fase sono il type checking, ovvero il controllo di tipo, controllare che gli identificatori siano stati dichiarati prima di essere usati e così via. Come supporto a questa fase viene creata una tabella dei simboli (*symbol table*) che contiene informazioni su tutti gli elementi simbolici incontrati quali nome, tipo (se presente) etc. Il risultato di questa fase è l'albero sintattico astratto(AST).
* Generazione del codice intermedio: Dall'albero di sintassi viene generato il codice intermedio
* Ottimizzazione del codice intermedio.
* Generazione del codice oggetto.

Una volta generato il codice oggetto, verrà collegati ad altri codici oggetto se presenti attraverso il linker.

Per essere eseguito, questo blocco di codici oggetto, verrà caricato attraverso il loader nella Ram .(Processo completo spiegato nel Capitolo 4).

Ci accorgeremo della presenza di un’errore ,con la compilazione, solo alla fine della lettura dell’intero programma in linguaggio di alto livello.

A differenza del compilatore, l’interprete non traduce in linguaggio macchina , dunque non crea nessun file intermedio, in quanto esegue direttamente il programma dal codice sorgente.

L’interprete si ferma non appena incontra l’errore nel codice.

**Capitolo 3**

Componenti Computer Classico

3.1 RAM

L'acronimo RAM sta per Random Access Memory.E’ una memoria volatile.

Volatile sta a significare la sua capacità di salvare i dati su essa , ma eliminarli una volta che il pc viene spento.

Come discusso nel capitolo precedente , essa appartiene alla triade delle componenti principali dell’architettura di Von Neuman.

Essa svolge un ruolo molto importante , fetcha dalla memoria fisica (HD, SSD), molto più lenta , dati di programmi(non completi, ma andando a creare una memoria virtuale, spiegata più avanti nel capitolo) e li rende disponibili alla cpu.

La RAM è costituita di celle o allocazioni, nelle quale vengono salvati i dati.Ciascuna cella ha un numero, chiamato indirizzo.

Ogni cella è composta da un multiplo di 2 bit , e rappresenta la piu piccola unità indirizzabile

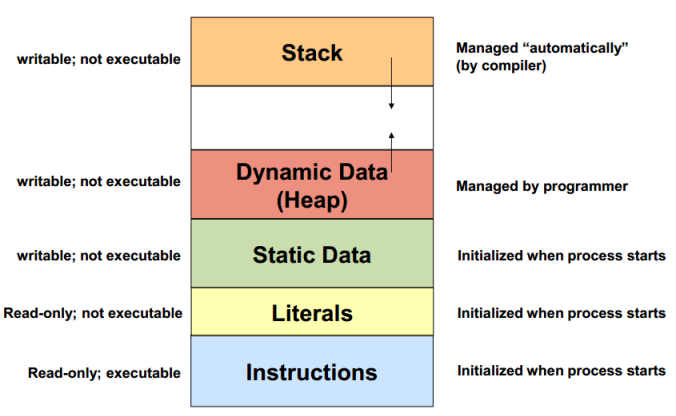
La parola (word) di un computer rappresenta quanti bit possono essere letti/scritti dalla CPU con un unico accesso alla memoria. Dimensioni tipiche per la parola sono: 16, 32, 64 e 128 bit. In genere, più grande è la parola maggiore è la potenza del computer.

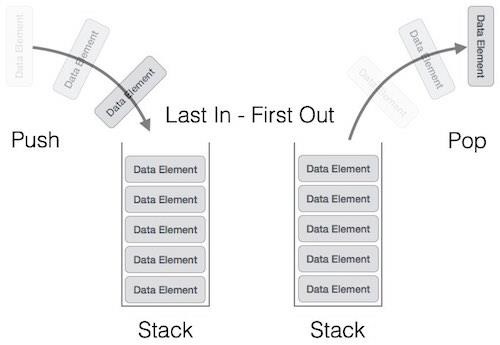
Un classico esempio è quando andiamo a scaricare un programma, vi sarà capitato di aver trovato due versioni dello stesso programma(stessa versione software, ma diverso modello x86 o 32bit e 64bit), questo è perche le istruzioni sugli attuali computer sono composte da 32 0 64 bit.

Detto questo, la capacita di una memoria è calcolata in base al numero di celle , per il numero di bit per singola istruzione.

Una memoria da 256 Mb(Ovvero 256 000 000 byte) che supporta le istruzioni a 64 bit(8 byte) avrà a disposizione 32 000 000 indirizzi.

La RAM divide la sua memoria principalmente in due macro parti : Stack ed Heap;



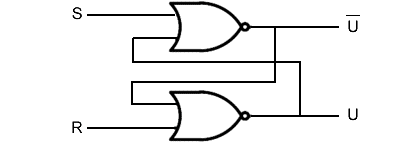
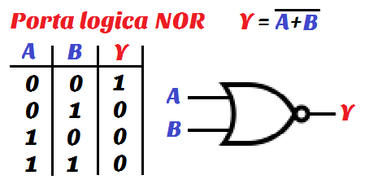
Nello stack i dati vengono alloca in maniera LIFO(Last in First out), ovvero l’ultimo dato ad essere stato introdotto, è il primo ad uscire. E’ la memoria allocata di default allocata da un programma, molto veloce ma con memoria limitata.

Al contrario, l’heap ha ampiezza maggiore a discapito di una velocità inferiore, e viene usato quando non si è a conoscenza di quanta memoria allocare per una particolare azione(In Java viene usato per allocare oggetti, in c per creare linked list tramite le funzioni malloc/realloc).

Come già sappiamo i dati hanno un dimensione espressa in multipli di byte , ma come vengono salvate queste informazioni nella memorie??

Per creare una memoria da 1 bit è necessario disporre di un circuito che in qualche modo “ricordi” i precedenti valori di input .

E’ possibile costruire un circuito di questo tipo , utilizzando due porte NOR.



Questo circuito è chiamato latch SR e ha due input ,S(Setting) per impostare il valore

Del latch e R(Resetting ) per azzerarlo .

Il circuito ha anche due output U e U(negazione U in quanto non potranno mai avere lo stesso risultato).

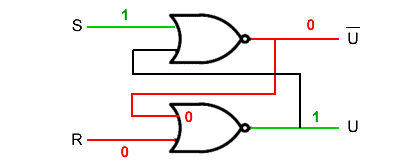
Esempio di utilizzo:

Settiamo S ad 1 ed avremo i seguenti passaggi:

-NOR Logico 1|0 darà come output 0

-Essendo circuiti sequenziali, l’output precedente sarà usato dalla seconda porta NOR

generando come secondo output 1.



Si definisce stato l’output generato in base all’input ricevuto(in base al settaggio del valore di input di S o R).

Uno stato in cui entrambi i valori di output valgono zero è instabile, in quanto obbliga entrambe le porte ad avere due zero come input , il che , se fosse vero,

Produrrebbe come risultato 1 e non 0.

Analogamente è impossibile che i due output valgano 1 , dato che forzerebbe gli input ad essere 0 e 1 , il cui risultato dovrebbe essere 0 e non 1 .

Le RAM , insieme alla memorie in generale , sono composte da flip-flop , particolare circuiti latch.

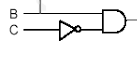
Prima di spiegare cosa sono i flip-flop ,è necessario spiegare cos’è il clock.

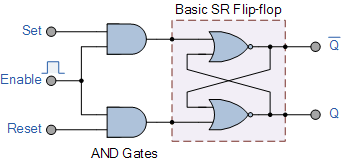
In molti circuiti digitali, l’ordine attraverso il quale si verificano gli eventi è cruciale.

Molti circuiti digitali utilizzano dei clock per gestire la sincronizzazione e permettere ai progettisti di ottenere le relazioni temporali desiderate .

Un clock è un circuito che emette una serie di impulsi di larghezza definita ad intervalli temporali costanti.L’intervallo temporale compreso tra le estremità dei due impulsi consecutivi è detto ciclo di clock.La frequenza degli impulsi è generalmente compresa tra i 100 MHz e 5 GHz(velocità di un processore).

I flip-flop utilizzano il clock per cambiare di stato, ma c’è bisogno di un modo di poter ritardare l’azione .Come vedremo nel capitolo 4 questo avviene in quanto le informazioni vengono salvate nelle memorie solo alla fine del ciclo di clok, ovvero nel fronte di salita.

Dunque è stato introdotto nel circuito latch un invertitore .Il quale ottiene come input il segnale , che viene ritardato dalla porta logica AND.



3.2 HardDisk/SSD

Essendo che non è possibile salvare grandi moli di dati sulla RAM , sono starte create delle memorie fisiche/secondarie , molto più lente nella fase di fetch rispetto alla RAM, ma molto più capienti.

L’Hard Disk è costituito da una serie di piatti coassiali (Disk–Pack) che ruotano intorno ad un asse comune con velocità tipiche di 5400 g/m e 7200 g/min. Ogni piatto è composto di materiale plastico ed è ricoperto su entrambe le facce da uno strato sottile (qualche micron) di materiale ferromagnetico. Tale strato presenta sulla superficie milioni di areole che possono essere magnetizzate da un campo magnetico. Un’areola magnetizzata corrisponde ad un 1 logico, un’areola smagnetizzata ad uno 0 logico. In prossimità di ciascuna faccia di ogni piatto si affaccia una testina di lettura/scrittura che può muoversi in senso radiale mentre il disco ruota a velocità costante. Il compito della testina è leggere o scrivere dati sulla superficie dei ciascuna faccia. Quando la testina deve memorizzare (scrittura) un bit emette un campo magnetico impulsivo che magnetizza l’areola, mentre, quando deve acquisire (lettura)un bit, passa attraverso l’areola: se la trova magnetizzata, si produce una corrente impulsiva indotta sul circuito della testina e questo corrisponderà ad un 1, altrimenti si avrà assenza di corrente e ciò corrisponderà ad uno 0.

Ogni testina è individuata da un numero intero a partire da 0.

Ogni faccia a sua volta è suddivisa in tracce circolari concentriche il cui numero dipende dalla capacità del disco. Le tracce, come le testine, sono numerate in successione a partire da 0 che individua la traccia più periferica.

Ogni traccia inizia con una speciale sequenza di bit chiamata preambolo e termina con un’altra sequenza di bit chiamata postambolo.

Tutte le tracce con lo stesso numero giacciono su una superficie cilindrica ideale del disco e questo permette di far riferimento al numero di traccia o di cilindro indifferentemente.

Ogni traccia è suddivisa in blocchi di bit chiamati settori. Un settore contiene 512 bytes ed è individuato mediante un numero a partire da 1. Il numero di settori per traccia è tipicamente 63.

Il settore è la più piccola unità di lettura/scrittura. In altre parole la lettura e la scrittura vengono effettuate per blocchi non minori di 512 bytes.

Esistono due modi per individuare un settore.

Il primo consiste nel fornire una terna di numeri: il numero della testina, il numero di cilindro e quello del settore. Questa terna è chiamata indirizzo CHS (Cylinder Head Sector).

Il secondo utilizza un solo numero per individuare un settore. In questa tecnica i settori vengono numerati utilizzando la successione 0,1,2....Così i settori della prima traccia e testina (C=0,H=0) andranno da 0 a 62, quelli con C=1, H=0 saranno 63,64,65.....125

Questo tipo di indirizzamento prende il nome di LBA (Linear Block Addressing) ed è quello utilizzato dal sistema operativo.

Un parametro importante dell'HD è il tempo impiegato per accedere ad un determinato settore che non sia sotto la testina. Esso consta di due parti:

* Seek Time : posizionamento della testina sulla traccia specificata
* Latency Time: accesso allo specifico settore della traccia

Le case costruttrici forniscono il tempo d’accesso medio in lettura e in scrittura

Nei computer moderni, si stanno iniziando sempre di più a montare SSD al posto dei classici HD .

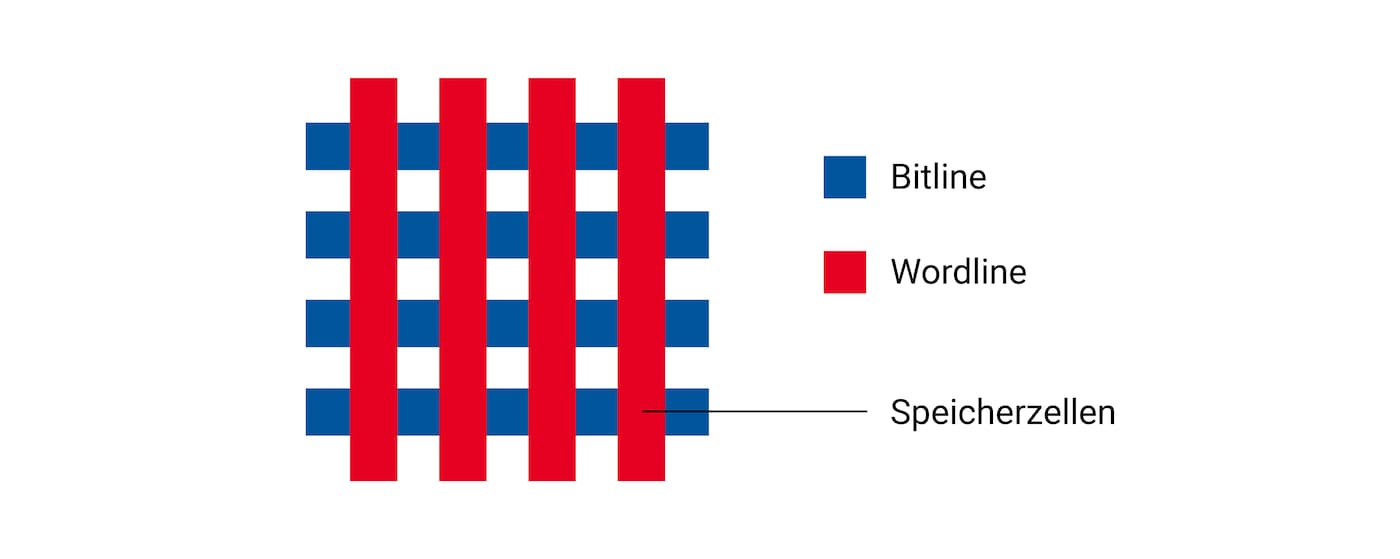
Questo è dovuto al fatto che il prezzo degli SSD sta diminuendo, è la loro velocità è maggiore rispetto ad un classico HD.

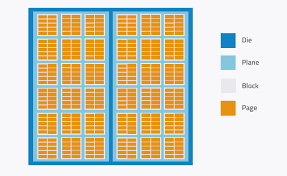
A differenza del disco rigido, le unità SSD non hanno testine di lettura/scrittura e piatti rotanti. Ogni blocco di un SSD è accessibile alla stessa velocità, indipendentemente che essi si trovino uno accanto all'altro o su un altro modulo. Questo spiega anche perché deframmentare un SSD non ha senso. I tempi di accesso sono più brevi e le velocità di lettura/scrittura più elevate.

Le SSD sono composte da memorie NAND.

A differenza dell'HDD, la memoria NAND flash legge/scrive elettronicamente i dati. La NAND flash è caratterizzata da un'elevata densità di memoria e dal backup non volatile dei dati. In pratica fa quello che fanno i piatti del disco rigido: memorizza i dati in modo permanente. Ciò avviene generando e leggendo la tensione.

Un blocco NAND è costituito Bit Line e Word Line. Le Bit Line corrono parallele l'una all'altra. Sono fatte di silicio e sono separate l'una dall'altra da una cosiddetta Shallow Trench Isolation («Isolamento di trincea superficiale», in breve STI).

Le Word Line sono perpendicolari alle Bit Line. Su di esse si trova il Control Gate e sono costituite da polisilicio. Nel punto dove Bit Line e Word Line si sovrappongono si trova la cella di memoria.

Le celle di memoria di una World Line sono concentrate in una pagina dati. Le pagine dati sono le unità più piccole di un SSD che possono essere lette o scritte. Queste possono avere dimensioni di 2 KB, 4 KB, 8 KB o 16 KB, a seconda del processo di produzione o del tipo di flash.

Più pagine vengono agglomerate in blocchi. Per blocco ci sono tra le 128 e le 256 pagine. Un blocco è la più piccola unità di memoria che può essere cancellata. Tecnicamente, questo sarebbe possibile anche a livello di pagina. Tuttavia, la tensione necessaria a tal fine graverebbe sulle celle di memoria a tal punto che il ciclo di vita sarebbe più breve che se le celle di memoria fossero cancellate a livello di blocco (di seguito segue una descrizione più dettagliata dell'usura delle celle di memoria).

In parole povere, la memoria flash memorizza i dati collocando gli elettroni o rimuovendoli . I bit vengono letti utilizzando la tensione. Se sono presenti elettroni, il bit ha il valore 0. Se non ci sono elettroni, il bit ha il valore 1. Le celle di memoria sono considerate vuote o cancellate se hanno il valore 1 e scritte se hanno il valore 0

I blocchi a loro volta sono solitamente divisi in due livelli per modulo flash (detto anche die). I moduli sono posizionati sulla scheda, dove sono legati al controller attraverso quattro fino a dieci canali.

3.3 Cache

Le cache sono memorie molto piccole , che si trovano al secondo posto nella graduatoria di velocità per scrittura e lettura dei dati.

Esse sono precedute dai registri della CPU , e superano di gran lunga RAM e memorie secondarie.

Le cache svolgono un ruolo fondamentale, in quanto negli ultimi anni i processori hanno continuato ad incrementare la loro velocità di elaborazione(aumento Hz clock).

Questo rendeva sempre più lente le memorie , in quanto i processori , chiedevano dati ad una velocità sempre maggiore di quanto effetivamente le memorie potessero offrire questi dati (collo di bottiglia).

Una delle tecniche per evitare questo collo di bottiglia, è l’introduzione di due cache , una per le istruzioni, e una per i dati.

Questo aumenta la larghezza di banda del sistema di memoria , avendo i due tipi di dati in due cache differenti.

Le CPU contengono cache (L1:16-64 KB) per le istruzioni e per i dati .

Le cache di secondo livello(L2: 512KB/1MB) , non sono incluse nel chip della CPU, ma possono essere all’interno del suo involucro,vicino al chip, collegata attraverso circuiti ad alta velocità.

La cache di terzo livello (L3) si trova sulla scheda del processore.

In generale le cache sono annidate, nel senso che l’intero contenuto della L1 è contenuto in L2, e l’intero contenuto di L2 è presente in L3.

Il funzionamento della cache si basa principalmente su due principi di località, la località temporale e la località spaziale.

* Località temporale (temporal locality): I dati recentemente usati hanno un'alta probabilità di essere riutilizzati a breve. Una cache memorizza solo un sottoinsieme dei dati - *the most recent-used - i dati più recentemente usati* . I dati letti sono temporaneamente memorizzati in cache. Se il processore richiede lo stesso dato, questo è fornito dalla cache. La cache è efficiente perché loop brevi di istruzioni e routine (piccoli programmi) sono strutture comuni di programmi e generalmente diverse operazioni sono eseguite sugli stessi dati e variabili.
* Località spaziale (spatial locality): se un dato viene referenziato, è molto probabile che dati adiacenti siano a breve a loro volta acceduti. Le istruzioni e i dati sono trasferiti nella cache in blocchi fissi (cache blocks), noti come cache lines (linee di cache). La dimensione della linea di cache è nel range di 4-512 byte, cosicché più di un dato da processare (4/8 byte) è memorizzato in ciascuna cache entry (indirizzo di cache). Molti programmi sono altamente sequenziali. La successiva istruzione generalmente è all'indirizzo successivo di memoria. I dati sono generalmente strutturati e i dati in queste strutture sono normalmente memorizzati in posizioni contigue di memoria (stringhe di dati, tabelle, ecc.).

3.4 Cpu

Secondo componente, non per importanza , dell’architettura di Von Neuman.

La CPU è il fulcro delle operazioni che compie il nostro computer.

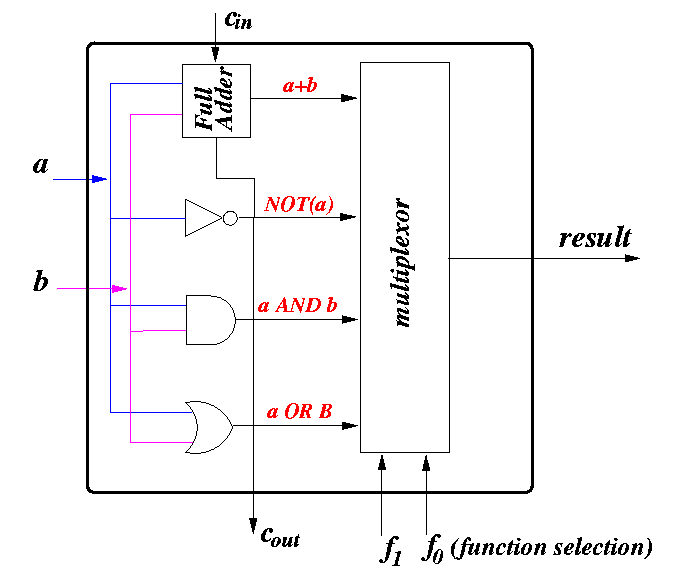
Essa è composta da varie parti:

-ALU(Arihmetic Logic Unit)

-Control Unit

-Registri

L’ALU è la parte fondamentale del nostro calcolatore, in quanto , come afferma il proprio nome , è la parte che attraverso le porte logiche , esegue le istruzioni sui dati.



L’ALU a sua volta è composta da 4 circuiti

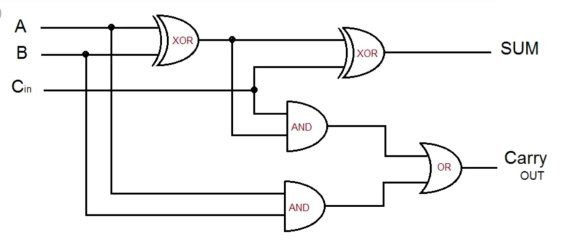
-Sommatore

-NOT

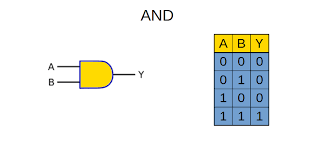
-AND

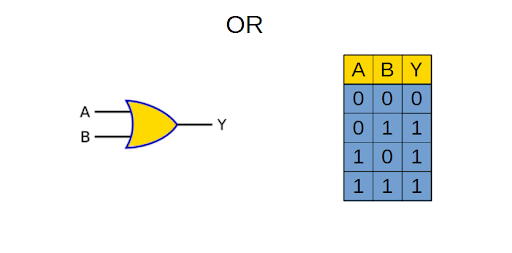
-OR

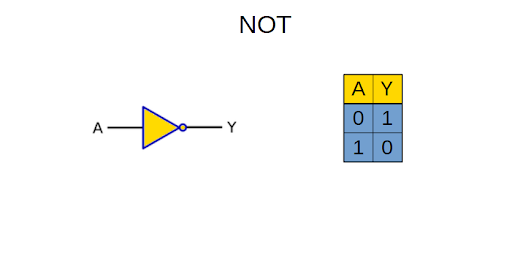
Il Sommatore è un circuito logico sequenziale, ovvero un circuito i cui valori di uscita non dipendono soltanto dai valori di ingresso correnti, ma anche da quelli precedenti.

Il Sommatore ha bisogno in input dei due dati da sommare, e del riporto del sommatore precedente(se presente).

Se per esempio inseriamo nel sommatore due valori da sommare 1|1 e il valore di riporto 0, avremo che la loro somma sarà uguale a 0 con il riporto di 1.

La porta AND è una delle fondamentali porte logiche che compongono l’architettura dei compinenti elettronici.Esse sfruttano le regole della logica classica(tabella nell’immagine).

Un’altra porta logica fondamentale è l’OR, che anch’essa segue le regole della logica classica.

Un’altra porta logica importante è il NOT, la negazione dell’input.

ALU decide quale circuito utilizzare per processare un particolare gruppo di dati, in base al tipo di input che riceverà dalla Control Unit .

L'unità di controllo è conosciuta anche con la sigla CU ( Control Unit ). È il dispositivo della CPU a cui spettano le funzioni di controllo. L'unità di controllo coordina il flusso di dati tra il processore e gli altri componenti del computer, legge ed esegue le istruzioni nella memoria centrale. Il funzionamento dell'unità di controllo del computer si basa sul processo fetch-execute ( lettura-esecuzione ).

-Fetch :nella fase fetch ( lettura ) l'unità di controllo legge dalla memoria centrale del computer le istruzioni da eseguire, scrivendole nel registro di istruzione. Le istruzioni sono registrate in particolari locazioni di memoria consecutive, al fine di agevolare la

successiva operazione di interpretazione ed esecuzione sequenziale delle istruzioni mediante la locazione del contatore di programma ( program counter ).

Execute. Nella fase execute ( esecuzione ) l'unità di controllo interpreta ed esegue in modo sequenziale le istruzioni situate nel registro di istruzione. Nel caso di calcoli tra dati numerici, l'unità di controllo trasferisce i dati dalla memoria centrale del computer all'unità aritmetico-logica ( ALU ) e attende il risultato.

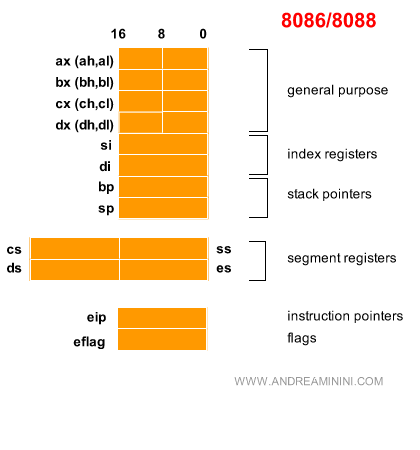
In base alla struttura di implementazione scelta , avremo due tipi di CU:

-cablata(Hardwired)

-microprogrammata

Per quanto riguarda quella cablata, avremo che a determinati impulsi, corrisponderanno determinate istruzioni. Il set di istruzioni è costante e non cambia. Pertanto, un’unità di controllo cablata viene utilizzata nei processori che utilizzano un semplice set di istruzioni noto come RISC (Reduced Instruction Set Computers).

L’unità di controllo microprogrammata ha una sequenza di microistruzioni memorizzate nella memoria di controllo. Queste microistruzioni contengono segnali di controllo per l’esecuzione di micro-operazioni. Inoltre, nei processori viene utilizzata un’unità di controllo microprogrammata basata su un complesso set di istruzioni noto come CISC (Complex Instruction Set Computer).

I registri, infine, sono utilizzati per contenere , durante un cliclo di clock ,i dati/istruzioni inerenti ad una determinata operazione. Questo rende la fase di fetch molto più veloce , in quanto i registro sono situate sul podio per quando riguarda la velocita di scrittura/lettura fra le memorie.

Lo scopo principale della CPU è possibile riassumerlo con 3 parole:

-Fetch

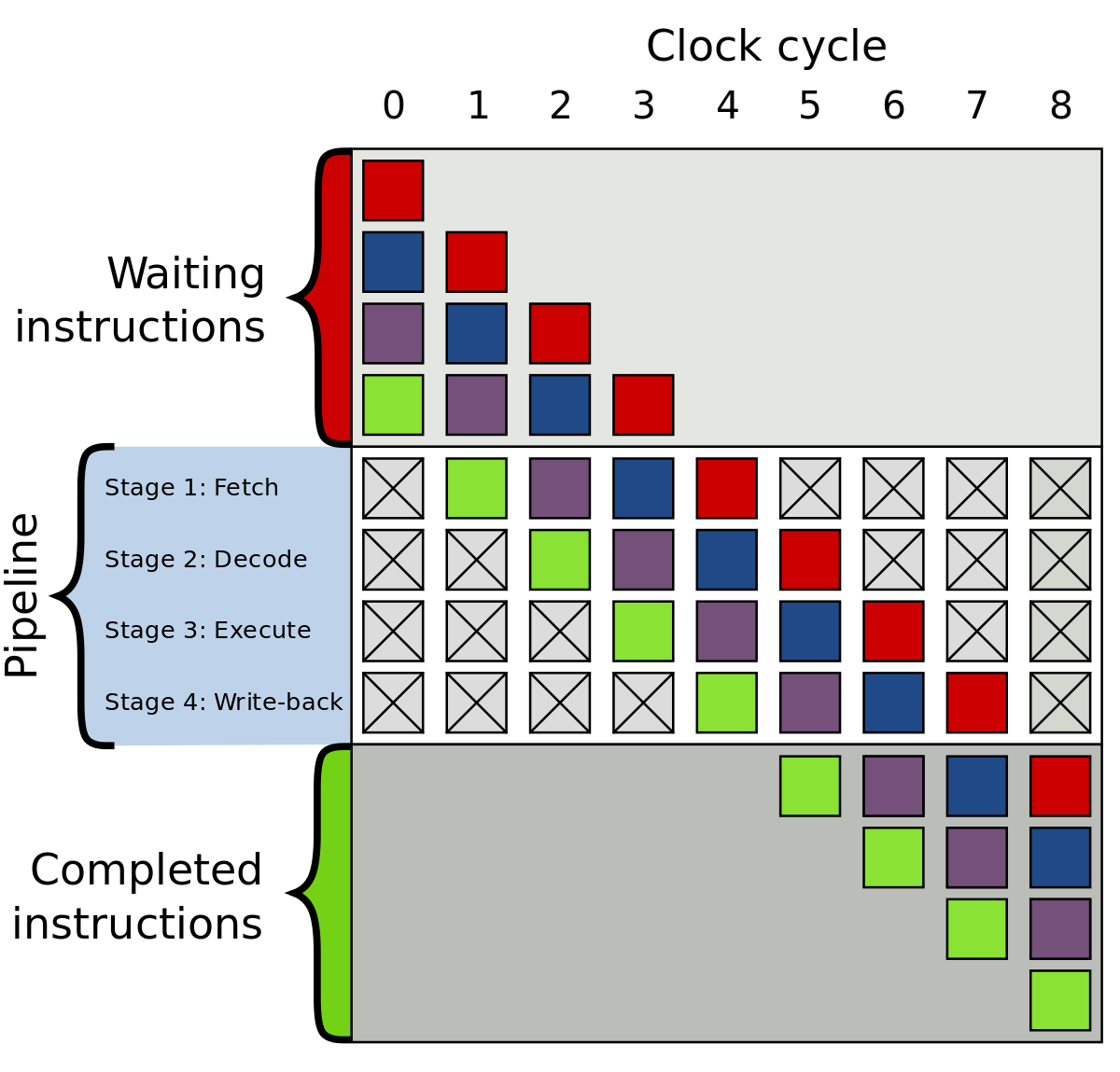
-Decode

-Excute

Infatti, essa fetcha le istruzione/dati, le decodifica ed infine esegue le operazioni sui dati fetchati.

Inizialmente le CPU svolgevano questo ciclo in maniera sequenziale, ma i produttori di questi componenti, idearono un nuovo modo per eseguire questo ciclo , che prende il nome di pipeline.

La pipeline non è altro che un modo parallelo di eseguire il ciclo di operazioni.



Come si evince dall’immagine, in ogni ciclo di clock la cpu esegue parallelamente i 3 stati del ciclo .Velocizzando di gran lunga il ciclo .

Il fattore velocità della CPU può essere incrementato anche andando a suddividere la CPU in più core , andando ancora di più a centralizzare l’importanza del parallelismo.

Questo tipo di architettura prende il nome di multi processore.

Questa architettura sarà di grande importanza per la realizzazione del prossimo componente la GPU.

3.5 GPU

Le GPU(Graphical Process Unit)

La Gpu , come precedentemente annunciato, è un componete formato da più cores che prendono il nome di CUDA core, e all’incirca il numero si aggira intorno ai 5000

In confronto ad un Cpu che ha dai 8 ai 32 core.

La differenza principale trai core CPU e GPU è la mole di lavore che essi possono compiere.

Notando che la GPU a molti più core rispetto alla CPU, ma essi possono essere usati solo per operazioni molto semplici che richiedono un grande numero di calcoli paralleli.

A differenza dei core della CPU che svolgono calcoli molto più complessi.

Il classico esempio per spiegare la differenza tra GPU e CPU è la stampa su schermo di un frame .

Ogni pixel/oggetto di un frame viene gestito da un core della GPU, questi dati vngono passati alla CPU che li processerà per andare a creare il frame.

Infatti nel mondo del gaming una GPU con prestazioni maggiori darà la possibilità al giocatore di poter aumentare la grafica con cui si gioco ad un determinato gioco.

Mentre una CPU più prestante darà la possibilità al giocatore di poter giocare ad un frame rate maggiore(ovvero immagini in un singolo secondo).

Le GPU si suddividono in intergate ed esterne .

Le GPU intergrate condividono memoria e potenza di calcolo con la CPU(avendo prestazioni nettemente inferiori rispetto alla controparte).

D’altro canto, le GPU esterne , hanno una propria memoria da cui fetchano le istruzioni, ed i propri core per eseguire i calcoli(andando ad aumentare esponenzialmente la velocità ma aumentando anche del’energia elettrica necessario per utilizzare il pc).

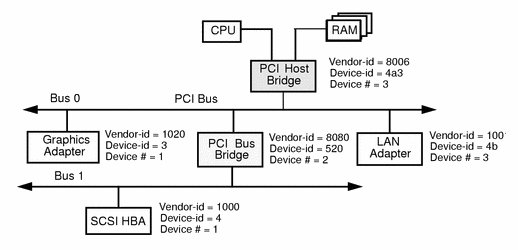
3.6 BUS

Il Bus è una struttura di comunicazione che mette in comunicazione i dispositivi di

I/O e memoria con la CPU.

Esso è integrato nella scheda madre .

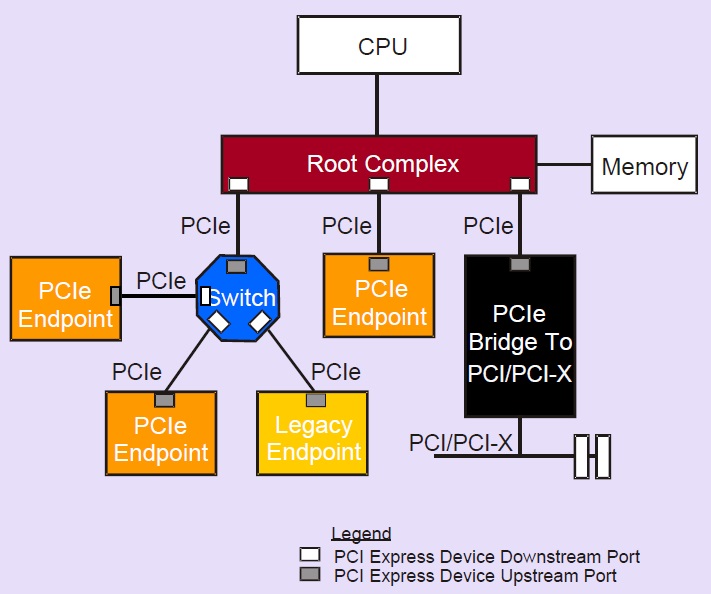
Il Bus è dotato di un chip ,chiamato arbitro del bus, che stabilisce i turni , ovvero scedula l’uso del bus per CPU e per i dispositivi di I/O.Generalmente le periferiche hanno la precedenza sulla CPU, dato che i i dispositivi di I/O non possono aspettare in quanto significherebbe una perdita di dati.(Approfondimento Capitolo 3.7 Periferiche).



Il BUS mostrato in figura , rappresenta un BUS PCI(Peripheral Component Interconnect) .La caratteristica che differenzia questo modello di BUS rispetto ai precedenti sta nell’implementazione di un Bridge tra memoria e CPU, che evita di occupare il Bus per comunicazione tra i due.

Nel mondo dei computer non importa quanto sia veloce una cosa: un sacco di persone penseranno che sia troppo lenta.Questo destino è toccato anche ai bus PCI, sostituiti con i bus PCIe.

Il bus PCIe non è nemmeno un bus, ma una connessione punto punto che utilizza commutazione di pacchetto ,più simile ad internet che ad un classico bus.



3.7 Periferiche

Gli attuali computer, come già detto in precedenza , sono basati sull’archietettura di Von Neauman , arricchiti con periferiche di ogni genere.

Esempi classici di periferiche (o sistemi I/O input output) possono essere:

-Lettori CD/DVD/Blu-Ray

-Tastiere

-Mouse

-Tavolette Grafiche

-Webcam

-HardDisk/SSD

Di periferiche c’è ne sono davvero tanto, ma adesso concentriamoci sulla solo struttura comune.

Ogni periferica è composta da componenti una parte hardware(la struttura del componente) e da un software chiamato driver.

I driver sono molto importanti per un dispositivo di I/O in quanto senza di essi , non sarebbe possibile la comunicazione con la CPU.

Nel capitolo precedente abbiamo parlato di bus , e di come i componenti di I/O abbiano la possibilità e la precedenza nella comunicazione sul Bus.

Ogniqualvolta la CPU necessita di una periferica, le invia un segnale sul Bus e nel tempo necessario alla periferica di svolgere il lavoro affidatole , la CPU esegue altri calcoli.

Essendo che le periferiche non possono aspettare, in quanto perderebbero dati, in loro aiuto entrano in gioco gli interuppt.

Gli interrupt non sono altro che avvisi che la periferica manda alla CPU non appena ha finito di svolgere il compito affidatole dalla CPU.

Una volta che la CPU riceve l’interrupt, per prima cosa decodifica da quale periferica è arrivato l’interrupt, successivamente interrompe tutto cioè che stava facendo per ricevere i dati attraverso il bus .

Una volta che la CPU riceve i dati dalla periferica, li elabora e continua , successivamente, con ciò che stava facendo prima.

Da non confondere interrupt con trap.

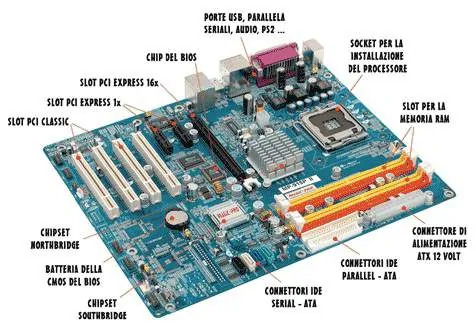
In quanto il primo è una richiesta che una periferica manda alla CPU per poter inviare il risultato dei dati da essa processati richiesti dalla CPU, la seconda invece è una successione che avviene in base ad un’azione non consentita, esempio divisione per zero.

In questo caso viene fecciata una sequenza di azioni dalla memoria che consente di recuperare dall’eccezione avvenuta.

3.8 Scheda Madre

Tutti i componenti di un computer vengono montati su una scheda madre, che ha integrata in esso il bus, consente ai componenti di comunicare tra loro.

I principali componenti di una scheda madre usata nei computer attuali :

* **Chipset**, l'insieme di [chip](https://it.wikipedia.org/wiki/Circuito_integrato) che si occupano di smistare e dirigere il traffico di informazioni passante attraverso il Bus di sistema, fra CPU, RAM e periferiche di input/output .
* **Socket**(CPU), una parte fondamentale del computer che accoglie la CPU. Nelle schede embedded (o in quelle vecchie e molto economiche) è assente, e il processore è saldato direttamente sulla scheda.
* **ROM**(*Read Only Memory)*, la piccola memoria presente su tutti i personal computer, che in alcuni casi può essere riprogrammata, (può essere PROM, EEPROM, flash o altro) .
* **RAM**(*Random Access Memory*), può essere di diversi tipi, esistono vari tipi di RAM diffusi dalle industrie sin dai primi anni ottanta
* **Il bus di PCI**. Si tratta di un collegamento dati generico punto-multipunto, progettato per permettere di collegare alla scheda madre delle altre schede *di espansione* alloggiate su connettori (*slot*), che ne estendono le capacità. Attualmente il tipo di bus più diffuso è il bus PCI Express, arrivata alla versione PCIe 4.0, con la prossima commercializzazione della versione 5.0, molto più veloce del PCI originario. In linea di principio a un bus può essere collegato hardware di ogni tipo: schede video aggiuntive, schede audio professionali, schede acquisizione dati, unità di calcolo specializzate, coprocessori.
* 

3.9 ROM

ROM(Read only Memory) appartiene alla famiglie delle memorie PROM(Programmable Read Only Memory), in quanto le case produttrici lasciano il ruolo di programmarle a chi poi andrà effettivamente ad utilizzarle.Esse sono programmabili una sola volta.

L’evoluzione delle PROM sono le EPROM(*Erasable Programmable Read Only Memory*) che non sono altro che delle PROM a cui è stata data la possibilità di essere riprogrammate un numero definito di volte attraverso l’uso di radiazioni ultraviolette.

Ultimo figlio della famiglia delle PROM sono le EEPROM(*Electrically Erasable Programmable Read Only Memory*)

Esse , a differenza delle precedenti EPROM , possono essere riprogrammate un numero di volte maggiore, e senza l’ausilio di radiazioni ultraviolette.

Le EEPROM sono il più diffuso componente elettronico ad utilizzare principi della meccanica quantistica; ad esempio, l'effetto tunnel.

L’effetto tunnel in breme definisce che , attraverso le leggi della meccanica classica sia impossibile che una particella superi una barriera con energia superiore rispetto all’energia posseduta dalla particella.

Questo non è vero per la meccanica quantistica , la quale afferma che la percentuale che avvenga questo evento è diversa da zero.

Un esempio è un elettrone ad energia cinetica fissata che, nel raggiungere una zona in cui esso dovrebbe essere respinto per via della energia insufficiente, riesce a superarla in una piccola frazione di casi.

Le ROM sono utilizzare per contenere firmware, e BIOS del computer.

**Capitolo 4**

Funzionamento complessivo componenti computer classico

Una volta compreso i vari componenti di un pc , andremo a vedere come essi interagiscono tra loro.

Spieghiamo tutto attraverso un’esempio che chiunque abbia mai avuto un computer avrà fatto almento una volta, aprire Word.

Per aprire Word, il numero di azioni che noi compiamo sono banali, e soprattutto impiegano una frazione di secondo:

-Muovere il mouse

-cliccare sull’icona di Word

In realtà, dietro queste semplici azioni, c’è una mare, che andremo a scoprire insieme.

Iniziamo dal dispositivo di I/O il mouse.

Una volta mosso il mouse,il driver del mouse registrerà i movimenti effettuati con esso, ed invierà ,attraverso il bus , queste informazioni alla CPU.

Questo come già detto è definito interrupt, in quanto la CPU smette di fare ciò che sta facendo, per eseguire il movimento del mouse sullo schermo.

Una volta mosso il mouse, e cliccato sul file eseguibile di Word , avverrà il processo di caricamento del file eseguibile tramite il loader nella memoria centrale.

Ma come avviene nello specifico il caricamento?Viene caricato tutto il programma nella RAM?Come funziona il caricamento se il file da eseguire ha un numero maggiore di byte di quanti la RAM possa ricevere?

Tutte queste domande hanno una risposta in comune: La Memoria Virtutale.

La **memoria virtuale** è un'architettura di sistema capace di simulare uno spazio di memoria centrale (memoria primaria) maggiore di quello fisicamente presente o disponibile, dando l'illusione all'utente di un enorme quantitativo di memoria.

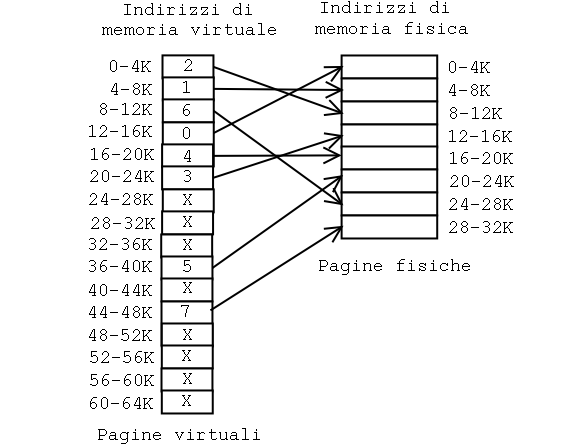
Essendo che molti programmi sono davvero molto pesanti, non è possibile caricarli tutti nella RAM , che dispone di capacità limitata , cosi entrano in gioco due approcci:

-la paginazione

-la segmentazione

Per la paginazione è necessario che si abbia una memoria in cui è contenuto l’intero programma .

Si pensi al programma nella memoria secondaria come al programma originale, ed ai i pezzi inviati alla RAM, come una parte del programma originale.

In caso di modifica al programma, verrà modificato anche l’originale presente in memoria secondaria.Lo spazio degli indirizzi virtuali è suddiviso in un certo numero di pagine della stessa dimensione.La dimensione di una pagina è sempre una potenza di 2.Allo stesso modo , anche lo spazio degli indirizzi fisici è suddiviso in porzioni che hanno la stessa dimensione delle pagine.

Questi pezzi di memoria principale che servono per contenere le pagine della memoria virtuale prendono il nome di blocchi di memoria(page frame).

Ogni computer provvisto di una memoria virtuale è provvisto di un dispositivo atto alla corrispondenza tra i blocchi di memoria e le pagine della memoria virtuale che prende il nome di MMU(Memory Management Unit).

Esso può risiedere nel chip della CPU o in un chip separato che lavora a stretto contatto con la CPU.

Per prima cosa la MMU controlla nella tabella delle pagine(memoria virtuale) se la pagina richiesta dal processo sia situata nella memoria principale.

In caso essa sia situata nella memoria, il valore verrà prelevato direttamente dalla memoria primaria(come gia detto in precedenza, quest’operazione è molto conveniente , in quanto la memoria centrale è molto più veloce nella scrittura e lettura dei dati rispetto alla memoria secondaria).

In caso non si trovasse nella memoria centrale, si solleva un’errore di pagina (page fault) ovvero una trap.

Dopo l’occorrenza di un’errore di pagina, il OS(sistema operativo) leggerà dalla memoria secondaria la pagina richiesta , la inserirà nella memoria primaria ripetendo, infine, l’istruzione che ha causato l’errore.

Il caricamento delle pagine nella tabella delle pagine viene effettuato attraverso l’algoritmo della paginazione a richiesta.

In parole povere, le pagine vengono caricate in memoria principale solo quando viene effettuata una richiesta, non prima.

L’insieme della pagine usate di frequente da un programma prende il nome di working set.

Per fare spazio nella memoria principale a pagine non presente, e di cui c’è richiesta vengono usato prevalentemente due algoritmo:

-LRU(Least Recent Used)

-FIFO(First in First out)

L’algoritmo LRU cancella dalla memoria centrale la pagina usata meno recentemente

L’algoritmo FIFO ,invece, rimuove dalla memoria centrale la pagina caricata meno recentemente.

In molte situazioni, è necessario disporre di più spazi degli indirizzi virtuali separati , per esempio , un compilatore potrebbe avere molte tabelle che si riempiono man mano che un programma viene compilato:

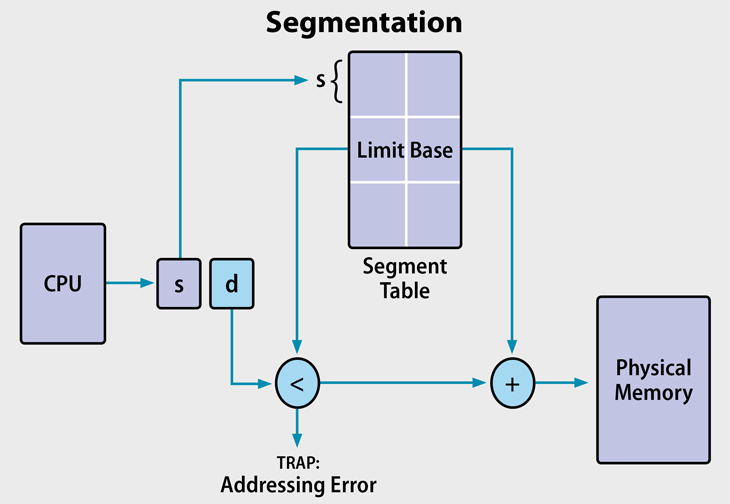
-tabella dei simboli che contiene nomi e attributi variabili.

-il codice sorgente.

-una tabella per contenere tutte le variabili in virgola mobile e intere.

-l’albero sintattico del programma.

-lo stack, per le chiamate di procedura del compilatore.

In una memoria unidimensionale , queste cinque tabelle sarebbero state allocate in unità contigue dello spazio degli indirizzi virtuali.In caso una di queste tabelle dovesse riempirsi,potrebbero verificarsi due situazioni.

In un caso, verrebbe fatto notare con un messaggio di errore l’impossibilità di proseguire con la compilazione.

In un’altro caso, il compilatore sotragga spazio ad altre tabelle che hanno spazio libero.

La soluzione è questo problema è la segmentazione, che prevede la suddivisione della memoria in spazi indipendenti , detti segmenti.(Esempio stack, heap)

Ogni segmento consiste in una sequenza lineare di indirizzi con lunghezza differente.

Poichè i diversi segmenti sono indipendenti, possono crescere e decrescere in maniera indipendente , senza andare ad influenzarsi a vicenda.

Differentemente dalla paginazione , in cui per definire una pagina era sufficiente un’identificativo(numero della pagina),nella segmentazione sono necessari due identificativi, il segmento e l’indirizzo all’interno del segmento.

La segmentazione può essere implementata attraverso lo swapping e la paginazione stessa.

Per quanto riguarda lo swapping o frammentazione esterna, segue lo stesso modus operandi della paginazione, in caso sia richiesto un segmento non presente nella memoria, esso viene creato (se non gia presente nella memoria secondaria) e caricato nella memoria primaria, anche per i segmenti vale la regola che vengono caricati solo quando richiesti.

Segmentazione e paginazione differiscono in quanto i primo non hanno dimensione fissa a dispetto delle pagine.

Lo swapping prende il nome anche di frammentazione esterna , in quanto, non avendo dimensione uguale i segmenti, una volta caricati in memoria , non si incastreranno alla perfezione andando a creare spazi di memoria inutilizzati, andando a diminuire prestazioni.

In questa situazione entrano in gioco i garbage collector che puliscono e riordinano la memoria in modo tale da ottimizzare li spazi.

Due algoritmo utilizzati per la segmentazione a frammentazione esterna sono :

-best fit

-first fit

Il best fist sceglie la porzione di memoria più piccola che può contenere il segmento,

Andando ad incidere sulle prestazioni, ma tenendo più ordinata la memoria.

Al contrario, first fit, sceglie la prima porzione di memoria disponibile per allocare il segmento, andandò a migliorare le prestazioni complessive.

L’altro modo per implementare la segmentazione è di suddividere ogni segmento in pagine e svolgere la loro paginazione su richiesta.

Ogni segmento, dunque, richiederà una tabella delle pagine .

In questo caso avremo la frammentazione interna.

Classico esempio della segmentazione con paginazione e frammentazione interna è lo stack, nel quale ogni pagina ha la stessa dimensione, ma ciò che vi è caricato dentro no, andando a lasciare spazio libero nella pagina .

Una volta capito come vengono caricati in memoria programmi molto più grandi di essa, possiamo continuare nella spiegazione.

I dati dalla memoria secondaria alla memoria centrale verrano fatti passare attraverso il bus, e i dati che utilizzarà la CPU per eseguire le operazioni verrano prelevate dalla RAM attraverso il bridge PCI.

Una volta arrivate nella CPU le istruzioni e i dati svolgeranno una serie di passaggi, e cio avviene in ogni ciclo di clock.

Nella fase di fetch la CPU preleva le istruzioni o i dati , dallo stack, che deve eseguire, nel ciclo di clock ci troviamo nel fronte di discesa.

Ogni istruzione fetchata avrà nella parte finale dell’ultima parola che compone l’istruzione, l’indirizzamento ai dati che necessita.

Possiamo dividere l’indirizzamento in :

-Immediato:nel caso in cui l’istruzione contenga direttamente il dato

- Diretto:nel caso in cui è presente l’indirizzo in cui è situato il dato

- Indiretto:nel caso in cui viene fornito un puntatore che restituirà l’indirizzo del dato.

La sequenza delle istruzioni da eseguire sarà dettata dalla CU.

I dati vengono salvati momentaneamente nei registri(primo posto velocita lettura scrittura).Una volta acquisiti tutti i componenti necessari per eseguire un’operazione, vengono passati all’ ALU che eseguirà l’operazione, e salverà se ancora necessari per istruzioni future i dati nei registri, oppure li salverà direttamente in memoria secondaria.

Prima del fronte di salito, ovvero della fine del ciclo di clock , e dopo che l’ALU ha processato ed eventualmente salvato i dati nei registri, c’è la fase di richiesta di lettura o scrittura dei dati in memoria .

Una volta che verrà stabilito il working set , il programma partirà sullo schermo , consentendoci di svolgere tutte le funzioni che esso è stato progettato per eseguire.

**Capitolo 5**

Introduzione meccanica quantistica

La **meccanica quantistica** è la teoria fisica che descrive il comportamento della materia, della radiazione e le reciproche interazioni, con particolare riguardo ai fenomeni caratteristici della scala di lunghezza o di energia atomica e subatomica, dove le precedenti teorie classiche risultano inadeguate.

La fisica quantistica inizia ufficialmente nel 1900, con il lavoro di **Max Planck** sullo spettro del **corpo nero**. Un corpo molto caldo diventa luminoso.

Tutto ciò avviene per *Bremsstrahlung* (“radiazione di frenamento”).

Ovvero, essendo che alte temperature significano aumento dell’energia cinetica, ed aumentando l’energia cinetica, gli elettroni si staccheranno dai nuclei , andando a produrre ioni negativi(elettroni) e ioni positivi(nuclei non più stabili). Gli elettroni inizieranno a muoversi in maniera indipendente. In caso dovessero scontrarsi con altri nuclei si avrà la **radiazione di frenamento** che è una radiazione elettromagnetica , ovvero fotoni di alta energia.

Sorprendentemente, la fisica di fine Ottocento non era in grado di spiegare questo spettro, anzi prevedeva un risultato privo di senso. Un corpo caldo avrebbe dovuto emettere una quantità infinita di energia a frequenze elevate, un comportamento vietato dal principio di **conservazione dell’energia**.

Cosi Plank creò la sua costante , la costante di Plank , per far tornare i conti

*h = 6,626 070 15 × 10 − 34 J ⋅ s*

Einstein , ebbe un’idea, ovvero poter risolvere il problema riscontrato da Plank attraverso l’effetto fotoelettrico, e viceversa,ovvero svelare i mistri dell effetto fotoelettrico attraverso il corpo nero.

L’effetto fotoelettrico consiste in un esperimento nel quale un elettrodo di metallo emette **elettroni** quando viene investito da una **radiazione ultravioletta**. Gli elettroni escono dal metallo .(probabile effetto tunnel spiegato prima ?????)

Da ciò si scoprirà che la materia e la luce hanno forma corpuscolare.Andando a definire il fotone come quanto.

Secondo la teoria di Plank ipotizza che la radiazione emessa dal corpo nero non fosse continua ma fosse invece “quantizzata”, ovvero emessa in quantità discrete **di** energia dette “quanti **di** energia”.

Ma ,c’è da dire che la luce si comporti in maniera bivalente, infatti essa è sia corpuscolare che ondulatoria, in quando nessun oggetto che presenta massa può muoversi alla velocità della luce.ma a sua volta è corpuscolare per il precedente accostamento tra effetto fotoelettrico e corpo nero.

Einstein afferma anche che la **teoria di Planck della radiazione** non si limitati solo alla luce. toccando il comportamento di ogni sistema in grado di oscillare e di assorbire e scambiare energia.

Infatti esso afferma , tramite la **teoria quantistica dei solidi,** che come per la luce, anche i solidi scambino energia soltanto per pacchetti discreti, multipli interi del quanto di energia ipotizzato da Planck.

La **teoria quantistica dei solidi** è una delle branche più fertili della meccanica quantistica, da cui è nata anche la **tecnologia dei semiconduttori** e gran parte dei dispositivi elettronici che sono ormai parte della nostra vita quotidiana.

**Capitolo 6**

Computer quantistico

Un **calcolatore quantistico** è un computer che utilizza le proprietà quantistiche della materia.

Il modello prevalente di computazione quantistica descrive la computazione in termini di una rete di porte logiche quantistiche.

Un sistema hardware quantistico ha all'incirca le dimensioni di un'automobile ed è costituito principalmente da sistemi di raffreddamento per mantenere il processore superconduttore alla sua temperatura operativa ultra-fredda.

Un processore classico utilizza bit classici per eseguire le sue operazioni. Un computer quantistico utilizza qubit per eseguire algoritmi quantistici multidimensionali.

I bit , come spiegato in precedenza, possono rappresentare uno solo dei due stati da loro disponibili in un’istante di tempo.

I qbit , sorvolano questo problema , rendendo possibile che esso assuma tutti i possibili stati nello stesso istante di tempo.

I processori quantistici devono essere molto freddi, circa un centesimo di grado sopra lo zero assoluto, per evitare la "decoerenza" o mantenere i loro stati quantistici. Per raggiungere questo obiettivo, si utilizzano superfluidi ultra-raffreddati. A queste temperature bassissime, alcuni materiali presentano un importante effetto meccanico quantistico: gli elettroni si muovono attraverso di essi senza resistenza. Ciò li rende "superconduttori".

Quando gli elettroni,nello stato fondamentale, passano attraverso i superconduttori, si accoppiano, formando "coppie di Cooper"(non si respingono perchè c’è una stasi per la criogenesi). Queste coppie possono trasportare una carica attraverso barriere, o isolanti, tramite un processo noto come tunneling quantistico.

Caratteristiche quantistiche principali qbit:

**Sovrapposizione**

Il qbit entra in uno stato di sovrapposizione, che rappresenta una combinazione di tutte le possibili configurazioni del qubit.

**Entanglement**

L'entanglement quantistico è un effetto che mette in relazione il comportamento di due cose separate. I fisici hanno scoperto che quando due qubit sono entangled, le modifiche a un qubit hanno un impatto diretto sull'altro.

**Interferenza**

In un ambiente di qubit entanged posti in uno stato di sovrapposizione, ci sono onde di probabilità. Queste sono le probabilità dei risultati di una misurazione del sistema. Queste onde possono accumularsi l'una sull'altra quando molte di esse raggiungono il picco in corrispondenza di un particolare risultato(interferenza costruttiva), oppure annullarsi a vicenda (interferenza distruttiva) quando i picchi e le depressioni interagiscono.