Architettura dei Calcolatori

I principali componenti

versione: febbraio 2016

autore: prof. Giacomo Pastorino

SOMMARIO

Nota: quanto segue deve essere integrato con i propri appunti presi durante le lezioni in cui il docente ha fornito diverse spiegazioni ed esempi degli argomenti che seguono, al fine di facilitare la comprensione degli stessi. L'attiva partecipazione dello studente alle lezioni è pertanto necessaria per il raggiungimento di una preparazione soddisfacente.

Per qualsiasi dubbio, chiarimento o segnalazione, contattatemi al seguente indirizzo email: gpastorino@calvino.ge.it.

Prof. Giacomo Pastorino

INTRODUZIONE

Un calcolatore è un sistema in cui processori, memorie e dispositivi periferici sono connessi tra loro.

Il più importante contributo nell'organizzazione logica dei componenti dei calcolatori moderni si deve al matematico, fisico ed informatico ungherese **John Von Neumann**. Il primo progetto che descrisse è conosciuto come **macchina di Von Neumann** ed è ancora oggi, a mezzo secolo di distanza, alla base di quasi tutti i computer digitali. La macchina di Von Neumann era composta da cinque componenti principali: la *memoria*, *l'unità aritmetico logica* (ALU - Aritmetical Logic Unit), *l'unità di controllo* (CU - Control Unit) e i dispositivi di input e output. L'unità aritmetico-logica e l'unità di controllo formavano insieme il "cervello" del computer. Nei moderni calcolatori queste due unità sono combinate in un unico chip: la **CPU** (*Central Processing Unit*). La FIGURA 1 mostra la macchina originale di Von Neumann.

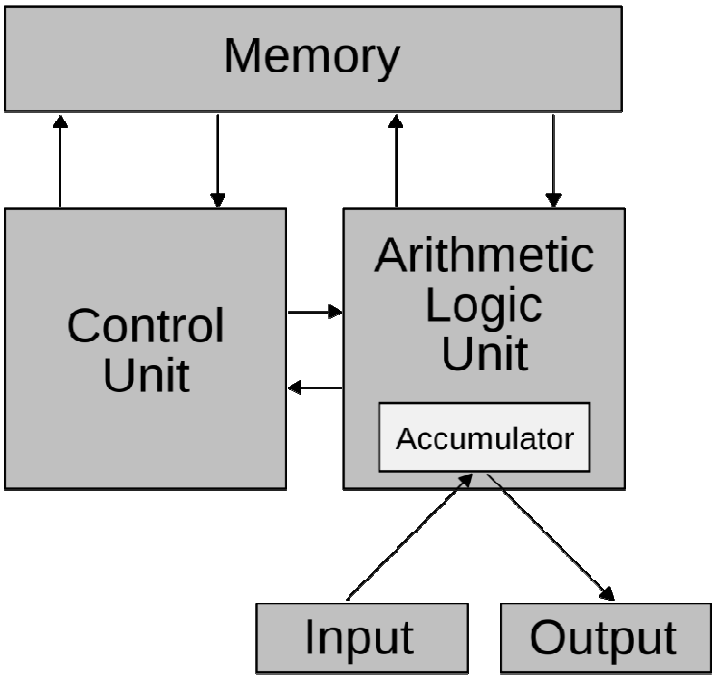

FIGURA 1 - LA MACCHINA DI VON NEUMANN

L'organizzazione dei moderni calcolatori è definita *bus-oriented*. I componenti sono connessi tra loro tramite il bus di sistema, un insieme di cavi paralleli sui quali vengono trasmessi dati, indirizzi e segnali di controllo. In FIGURA 2 è mostrata l'organizzazione di un semplice calcolatore.

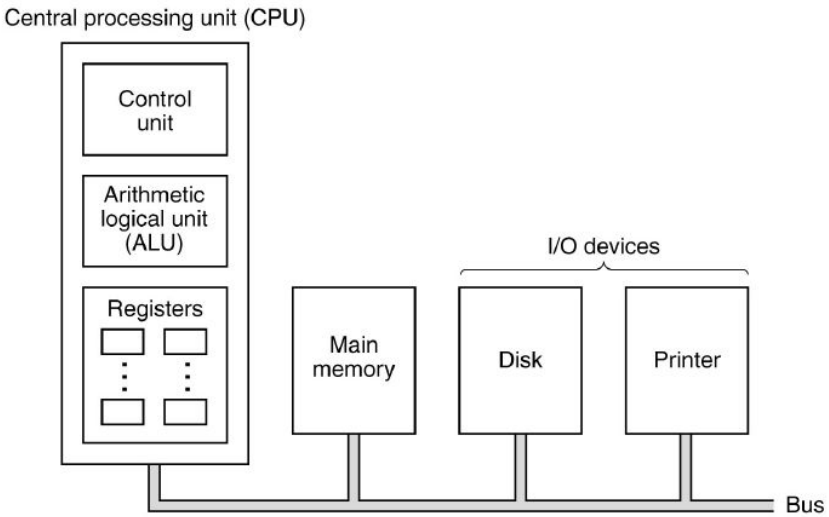

FIGURA 2 - ORGANIZZAZIONE BUS-ORIENTED

CPU - CENTRAL PROCESSING UNIT

La **CPU**, come già detto, è il "cervello" del computer e la sua funzione è quella di eseguire i programmi contenuti nella memoria principale: ne preleva le istruzioni, le esamina e le esegue.

La CPU è composta da tre parti distinte: l' **Unità Aritmetico-Logica**, l'**Unità di Controllo** ed i **registri**. L'**ALU** è la parte della CPU che esegue le operazioni, per esempio la somma e l'AND, necessarie per portare a termine l'esecuzione delle istruzioni. La **CU** si occupa invece di prelevare le istruzioni dalla memoria principale e di determinarne il tipo. La CPU contiene anche una piccola memoria ad alta velocità: i **registri**. Ogni registro ha una determinata funzione. Nei registri vengono memorizzati i risultati temporanei delle operazioni e alcune informazioni di controllo. Essendo interni alla CPU i registri possono essere letti e scritti a velocità elevate. Tre registri particolarmente importanti sono il ***Program Counter (PC)***, l'***Instruction Registrer (IR)*** ed l'***Accumulatore (ACC)***. Il Program Counter contiene l'indirizzo in memoria della prossima istruzione che dovrà essere prelevata per l'esecuzione. L'Instruction Register contiene l'istruzione che si trova in fase di esecuzione. Durante l'esecuzione di un'operazione l'Accumulatore contiene uno dei due operandi e ad operazione terminata il risultato viene memorizzato in questo registro.

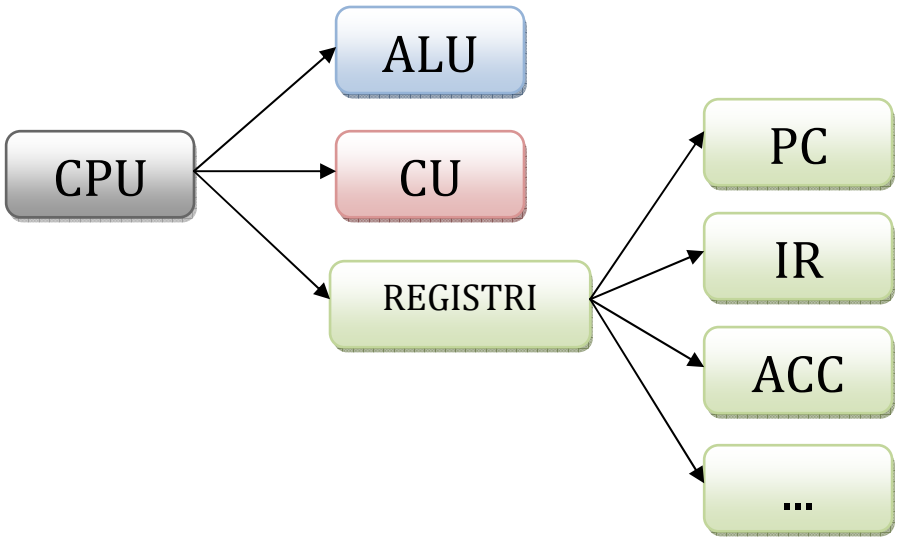

FIGURA 3 - COMPONENTI DELLA CPU

PERCORSO DATI

La FIGURA 4 mostra il **percorso dati** di una CPU. Nella figura sono visibili alcuni registri, la ALU e alcuni BUS che connettono fra loro le diverse parti.

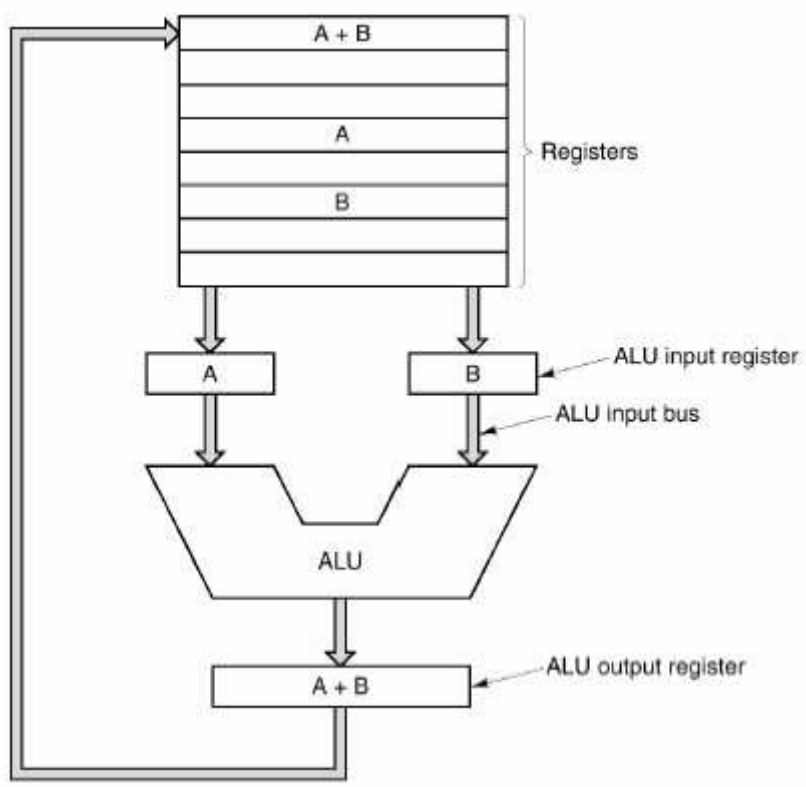

FIGURA 4 - PERCORSO DATI

I registri alimentano due ***registri di input*** della ALU che mantengono i dati in ingresso della ALU mentre questa è occupata nell'esecuzione di alcune computazioni. La ALU esegue l'operazione richiesta sui suoi input, terminata la quale memorizza il risultato nel ***registro di output***. Il risultato può essere quindi memorizzato in un registro della CPU e copiato in memoria successivamente. Non tutti i tipi di CPU prevedono la presenza dei registri di input e del registro di output. Nell'esempio in figura è illustrata l'operazione di addizione, la ALU può ovviamente compiere anche altre operazioni. Il processo che consiste nel portare gli operandi nella ALU, eseguire l'operazione richiesta e memorizzare il risultato nel registro di output è chiamato ***ciclo del percorso dei dati***.

Quasi tutte le istruzioni possono essere classificate in due categorie: le istruzioni **registro-memoria** e le istruzioni **registro-registro**. Le prime permettono di prelevare un dato o un'istruzione in memoria e copiarlo nei registri o di salvare i contenuti dei registri in memoria. Le seconde permettono invece di spostare i dati fra i registri della CPU.

Oggi i calcolatori hanno più ALU che operano in parallelo. Più veloce è il ciclo del percorso dati, maggiore risulta la velocità del calcolatore.

ESECUZIONE DELL'ISTRUZIONE

L'istruzione viene eseguita dalla CPU compiendo i seguenti passaggi:

1. prelievo dell'istruzione da eseguire dalla memoria. L'indirizzo da cui prelevare l'istruzione è memorizzato nel Program Counter. L'istruzione viene memorizzata nell'Instruction Register;
2. aggiornamento del Program Counter affinché in esso sia memorizzato l'indirizzo di quella che sarà la prossima istruzione da eseguire;
3. decodifica dell'istruzione. Si determina il tipo dell'istruzione prelevata al punto 1;
4. eventuale prelievo degli operandi dell'istruzione;
5. esecuzione dell'istruzione;
6. eventuale memorizzazione del risultato;
7. ritorno al punto 1 per eseguire istruzione successiva.

La sequenza appena descritta viene indicata come ciclo di **fetch-decode-execute** (caricamento-decodifica-esecuzione) dell'istruzione.

PARALLELISMO A LIVELLO DI ISTRUZIONE

Con il termine parallelismo si intende il compiere più azioni nello stesso tempo. Il parallelismo permette di ottenere prestazioni più elevate dai calcolatori. Esistono due forme principali di parallelismo: a *livello di istruzione* e a *livello di processore*. Nella prima forma si sfrutta il parallelismo all'interno della singola istruzione per far si che ne possano essere eseguite un maggior numero al secondo. Nel parallelismo a livello di processore sono invece presenti più CPU che lavorano contemporaneamente ad uno stesso problema.

PIPELINING

Il concetto di **Pipeline** consiste nel dividere una stessa istruzione in più parti che possono essere eseguite in parallelo. All'interno della CPU sono presenti pertanto hardware dedicati che assolvono compiti diversi e possono *lavorare in parallelo*. In figura 5 è rappresentato un esempio di Pipeline a cinque stadi.

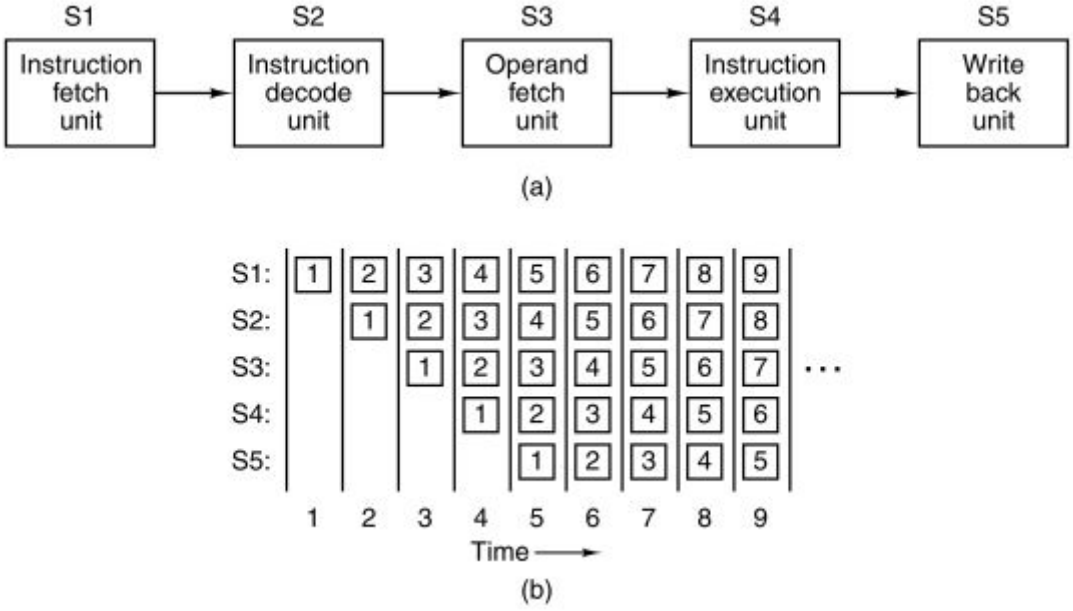

FIGURA 5 - (A) PIPELINE. (B) LO STATO DEGLI STADI IN FUNZIONE DEL TEMPO

Durante il primo periodo l'unità S1 carica l'istruzione 1. Nel secondo periodo S2 decodifica l'istruzione 1 mentre S1 carica l'istruzione 2. Nel terzo periodo S3 carica gli operandi dell'istruzione 1, S2 decodifica l'istruzione 2 e S1 carica l'istruzione 3. E così via. Nel quinto periodo tutte le unità sono operative contemporaneamente. La prima istruzione viene terminata dopo il quinto periodo. In seguito, ad ogni periodo viene terminata un'istruzione. Nel caso in cui un'istruzione debba operare sul risultato di un'istruzione precedente, deve necessariamente attendere che questa abbia memorizzato il proprio risultato prima di effettuare il caricamento degli operandi.

ARCHITETTURE SUPERSCALARI

La presenza della Pipeline contribuisce ad aumentare le prestazioni dei calcolatori. E' logico pensare che avere due Pipeline porti vantaggi ancora maggiori in termini di prestazioni. In FIGURA 6 è mostrato lo schema di una CPU con doppia pipeline in cui un'unica unità di caricamento (S1) preleva due istruzioni alla volta e le inserisce nelle pipeline, ognuna delle quali è dotata di una ALU.

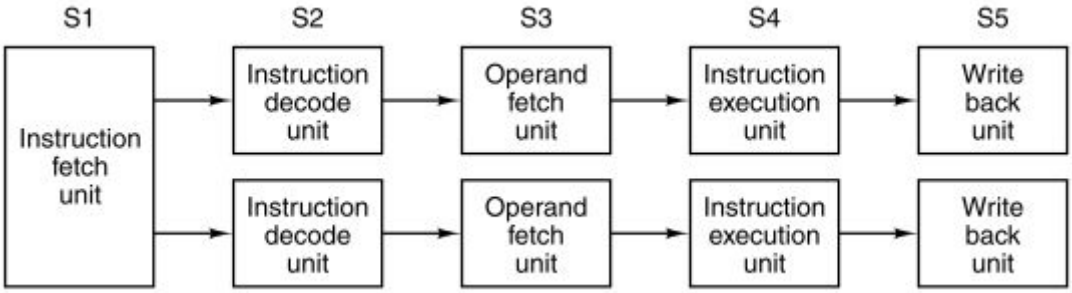

FIGURA 6 - DOPPIA PIPELINE

Affinché due istruzioni possano essere eseguite in parallelo non devono esserci conflitti nell'uso delle risorse del sistema e nessuna delle due deve dipendere dal risultato dell'altra. E' pertanto necessario, al fine di garantire il corretto funzionamento del sistema, che vengano individuati ed eliminati tali conflitti. Alcune istruzioni non potranno semplicemente essere eseguite in parallelo, pertanto il sistema provvederà a non accoppiarle durante l'esecuzione. Ne verrà quindi eseguita prima una e poi l'altra (in un ordine ben preciso se una delle due dipende dal risultato dell'altra). In questo caso una delle due istruzioni viene eseguita, mentre l'altra verrà accoppiata ed eseguita contemporaneamente ad una terza istruzione.

Si potrebbe pensare ad architetture con quattro pipeline, ma questo porterebbe a duplicare troppi elementi hardware. Si utilizza alternativamente un approccio che prevede di avere una singola pipeline con più unità di esecuzione dell'istruzione (S4). Questo tipo di architettura è chiamata **superscalare**. La FIGURA 7 ne mostra un esempio. Nel processore superscalare è implicito il fatto che lo stadio S3 sia significativamente più veloce dei singoli componenti dello stadio S4. Se così non fosse crollerebbe l'idea di base di questa configurazione e non sarebbe necessario avere più unità di esecuzione dell'istruzione.

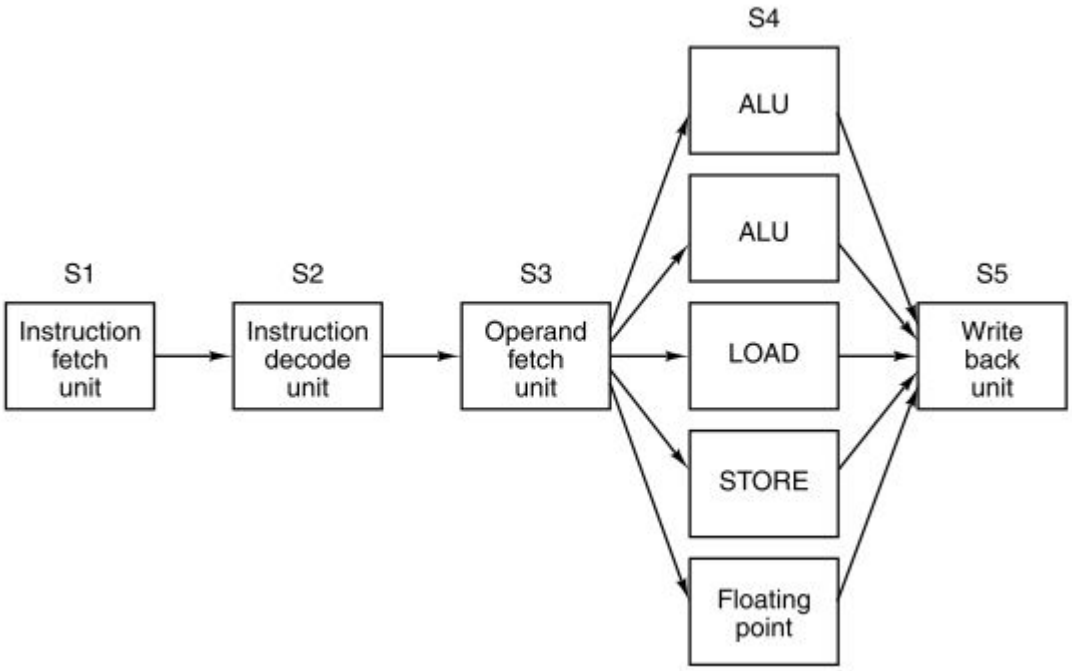

FIGURA 7 - ARCHITETTURA SUPERSCALARE CON CINQUE UNITÀ DI ESECUZIONE DELL'ISTRUZIONE

PARALLELISMO A LIVELLO DI PROCESSORE

Per ottenere prestazioni ancora più elevate l'unica soluzione è quella di progettare calcolatori con più di una CPU.

MULTIPROCESSORI

Il **multiprocessore** è un sistema composto da più CPU con una *memoria in comune*. Dal momento che tutte le CPU possono leggere e scrivere nella memoria comune, è necessario che si coordinino al fine di evitare di ostacolarsi a vicenda. Quando due o più CPU hanno interagiscono in modo così profondo si dice che sono *tighly coupled* (legate strettamente). Per ridurre il numero di conflitti e migliorare le prestazioni si può far ricorso ad architetture che prevedono la presenza di *memorie locali*, una per processore, oltre alla memoria condivisa. Per accedere alla propria memoria locale i processori utilizzano un bus dedicato. Pertanto il traffico nel bus diminuisce considerevolmente. Un vantaggio dei multiprocessori è rappresentato dal fatto che risulta facile programmare per un sistema a memoria condivisa.

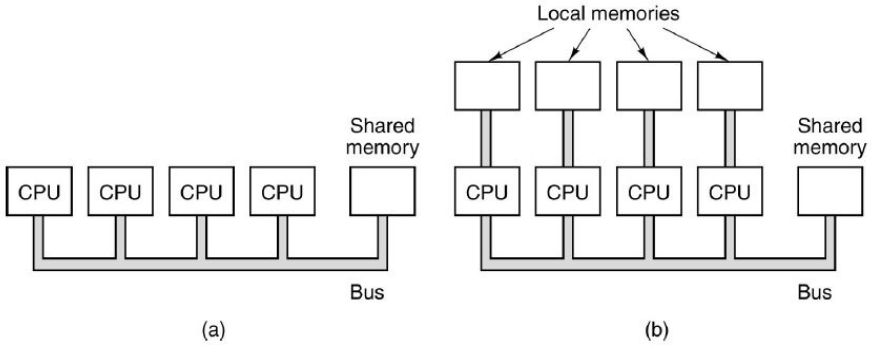

FIGURA 8 - MULTIPROCESSORE (A) E MULTIPROCESSORE CON MEMORIE LOCALI (B)

MULTICOMPUTER

Costruire multiprocessori fino ad un massimo di 256 processori è relativamente semplice. Realizzarne di più grandi diventa invece molto complicato. La difficoltà maggiore si ha nel connettere tutti i processori alla memoria condivisa. Per aggirare questo problema si possono costruire sistemi di calcolatori interconnessi, ciascuno dotato della propria memoria. Questi sistemi sono i **multicomputer**. Le cpu in questo caso si dicono *loosely coupled* (con legame lasco). Le CPU de multicomputer comunicano tra loro inviandosi messaggi molto veloci. Nei grandi sistemi un messaggio attraverserà più macchine intermedie nel percorso tra il mittente ed il destinatario, in quanto i calcolatori non saranno connessi mutualmente l'uno all'altro.

Come detto i multiprocessori sono facili da programmare mentre i multicomputer sono facili da costruire. Per unire le qualità di entrambi sono state effettuate parecchie ricerche indirizzate alla realizzazione di sistemi ibridi. Tali calcolatori danno l'illusione che esista una memoria condivisa (per ottenere la facilità nella programmazione) senza averne realmente una, in quanto troppo costosa.

MEMORIA PRINCIPALE

La **memoria** è quella parte del calcolatore in cui sono depositati i programmi ed i dati. L'unità di base della memoria è la cifra binaria: il **bit**. La numerazione binaria è il metodo più *affidabile* per memorizzare l'informazione digitale in quanto necessita la distinzione tra due soli valori. L'informazione digitale può essere memorizzata utilizzando valori di una cera quantità fisica continua, come la tensione o la corrente. Se occorresse distinguere fra più valori, come per esempio accade nella numerazione decimale, ci sarebbe una minor separazione tra i valori adiacenti e la memoria risulterebbe meno affidabile.

Le memorie sono costituite da un certo numero di **celle**, dette anche **locazioni**. I dati e le istruzioni sono memorizzati nelle celle della memoria. Ad ogni cella corrisponde un **indirizzo**, attraverso cui il programma può riferirsi alla cella stessa. Se una memoria ha *n* celle, gli indirizzi corrispondenti andranno da *0* a *n-1*. Tutte le celle della memoria contengono lo stesso numero di bit.

Le memorie principali degli odierni calcolatori sono memorie **RAM** - Random Access Memory (memoria ad accesso casuale). Il nome deriva dal fatto che, a differenza di quanto avviene per i dischi magnetici (vedi oltre), il tempo di accesso ai dati non dipende dalla posizione in cui questi sono memorizzati. Esistono due tipi di memorie RAM: ***statiche*** e ***dinamiche***. I due tipi di memorie hanno diverse caratteristiche. Le Ram statiche sono molto più veloci e più costose rispetto alle ram dinamiche. Per contro queste ultime necessitano di effettuare un *refresh* di ciascun bit a prefissati intervalli di tempo. Le ram statiche sono utilizzate principalmente per la memoria cache del calcolatore. La memoria principale è, di norma, costruita utilizzando ram dinamiche.

Le memorie RAM sono memorie di tipo **volatile**: quando non vengono alimentate (vale a dire quando il calcolatore viene spento) perdono i dati memorizzati al proprio interno.

MEMORIA CACHE

Le CPU sono più veloci delle memorie. Da un certo punto di vista, si può affermare che il progresso abbia contribuito a preservare questo divario. Dato che diventa possibile collocare sempre più circuiti in un chip, i produttori di CPU hanno tratto vantaggio da ciò per sviluppare le architetture pipeline e superscalare, migliorando pertanto le prestazioni dei processori. I produttori di memoria si sono invece storicamente concentrati sul produrre, grazie a queste nuove tecnologie, memorie sempre più capienti. Più lenta è la memoria, più tempo la CPU dovrà attende quando deve effettuare delle operazioni di lettura o di scrittura. Una tecnica che permette di ovviare a questi limiti consiste nel

combinare una piccola quantità di memoria veloce con una grande quantità di memoria lenta, al fine di ottenere buoni risultati mantenendo i costi entro limiti accettabili, sfruttando la velocità di una memoria e la capacità dell'altra. La piccola memoria veloce è la memoria **cache**.

La logica di base su cui si base l'utilizzo delle due memorie è semplice: i dati utilizzati più di frequente sono mantenuti nella memoria cache. Quando la cpu necessita di un dato, controlla prima se è presente in cache. Soltanto in caso negativo andrà a prelevare il dato dalla memoria principale. Il successo o il fallimento di questo approccio dipendono quindi da quali dati sono presenti nella memoria cache. I riferimenti alla memoria centrale tendono a seguire un principio, detto ***principio di località***, secondo cui in un breve intervallo di tempo, si tende a fare riferimento solo ad una piccola frazione di memoria. In altre parole, se la CPU ha necessità di reperire un dato memorizzato in una determinata locazione di memoria, con ogni probabilità, entro un breve intervallo di tempo, avrà la necessità di caricare un dato memorizzato in una posizione di memoria vicina alla posizione del dato precedente. L'idea che ne scaturisce è che, quando la CPU deve caricare un dato non presente in cache, esegue l'accesso alla memoria centrale e provvede a caricare in cache il dato stesso ed i dati vicini, i modo che quando serviranno alla CPU,verranno caricati velocemente in quanto presenti nella memoria cache.

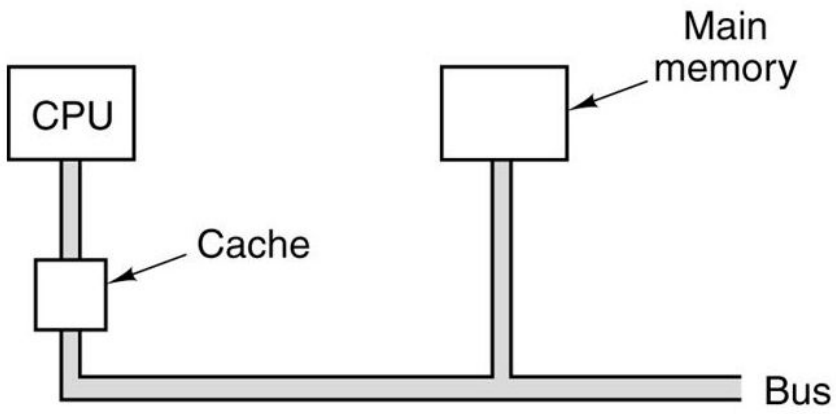

FIGURA 9 - POSIZIONAMENTO LOGICO DELLE MEMORIE

Riassumendo:

1. la CPU necessita di un dato;
2. la CPU ricerca il dato nella cache;
3. se lo trova, lo legge e lo utilizza;
4. se non lo trova, lo reperisce nella RAM e lo trasferisce nella cache, insieme ai dati ad esso attigui, nella locazione da più tempo non utilizzata;

Da un punto di vista logico, come illustrato in FIGURA 9, la memoria cache si trova tra la CPU e la memoria principale. Solitamente nei sistemi sono presenti più memorie cache: cache di ***livello 1 (L1)***, cache di ***livello 2 (L2)*** e cache di ***livello 3 (L3)***. La cache L1 è sempre integrata nel processore. Gli altri livelli di cache, in base all'architettura, possono essere integrati anch'essi nel processore o nella scheda madre.

MEMORIA SECONDARIA

La memoria principale non è mai abbastanza capiente. Inoltre essendo costruita utilizzando memorie RAM, se da un lato ha il vantaggio di essere una memoria relativamente veloce, dall'altro ha lo

svantaggio di essere volatile. Il calcolatore ha logicamente necessità di avere una memoria non volatile in cui programmi e dati vengono conservati anche a calcolatore spento. Questo compito è assolto dalla memoria secondaria.

GERARCHIA DI MEMORIA

La soluzione che si adotta nei calcolatori per memorizzare i dati è quella di organizzare gerarchicamente la memoria. In questo modo vengono sfruttate le caratteristiche positive di tutti i tipi di memoria. In FIGURA 10 è rappresentata una **gerarchia di memoria** a cinque livelli.

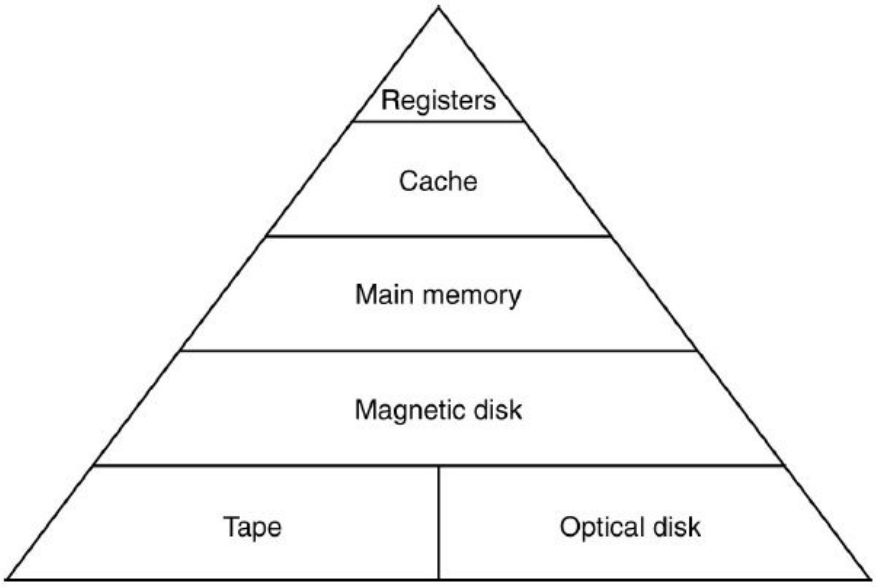

FIGURA 10 - GERARCHIA DI MEMORIA

Nella parte più alta della piramide si trovano i registri, ossia la memoria posizionata all'interno della CPU stessa. Più sotto vediamo la cache, di cui abbiamo parlato nel paragrafo precedente, la cui dimensione può variare dai 32 KB fino ad alcuni megabyte. Scendendo troviamo la memoria centrale, con dimensioni comprese tra 1 gigabyte per i sistemi più economici e centinaia di gigabyte per quelli professionali. Scendendo ancora si trovano i dischi magnetici, di cui parleremo nel paragrafo seguente, che rappresentano la memorizzazione permanente del sistema. Infine troviamo le memorie utilizzate per l'archiviazione, come i dischi magnetici (*tape*) e i dischi ottici. Da notare come muovendosi verso il basso della gerarchia variano tre parametri chiave:

- ✓ aumenta il tempo di accesso: via via che si scende nella gerarchia troviamo memorie sempre più lente.
- ✓ Aumenta la capacità di memorizzazione.
- ✓ Diminuiscono i costi.

DISCHI MAGNETICI

Un **disco magnetico** consiste in uno o più *piatti di alluminio rivestiti di materiale magnetico*. Il diametro dei piatti varia dai 3 ai 9 cm; i dischi più piccoli vengono utilizzati nei calcolatori portatili, in cui si trovano di dimensioni anche inferiori ai 3 cm. La **testina** del disco sfiora la superficie

(rimanendo sospesa su di un cuscinetto d'aria). Durante la scrittura del disco una corrente passa attraverso la testina e magnetizza la superficie che si trova al di sotto, orientando le particelle magnetiche in direzione opposta a seconda del verso della corrente. Al contrario, durante la lettura, quando la testina passa al di sopra di un'area magnetizzata, viene indotta nella testina una corrente positiva o negativa, rendendo così possibile la lettura dei bit memorizzati. In questo modo, mentre il piatto ruota intorno alla testina, è possibile leggere o scrivere un flusso di bit.

La sequenza circolare di bit scritti mentre il disco compie una rotazione completa è chiamata **traccia**; le tracce sono una serie di centri concentrici attorno all'asse di rotazione del disco. Le tracce sono divise in un certo numero di **settori** contenenti solitamente 512 byte di dati preceduti da un ***preambolo*** che permette alla testina di sincronizzarsi prima della lettura e della scrittura. I dati sono seguiti da un codice per la correzione dell'errore (***ECC***). Tra due settori consecutivi c'è una piccola area chiamata *spazio tra i settori*. Preambolo ed ECC vengono creati durante la formattazione del disco e rappresentano in sostanza zone di memoria che non possono essere utilizzate dai dati o dai programmi. Solitamente la capacità del disco formattato è inferiore del 15% circa rispetto a quella lorda.

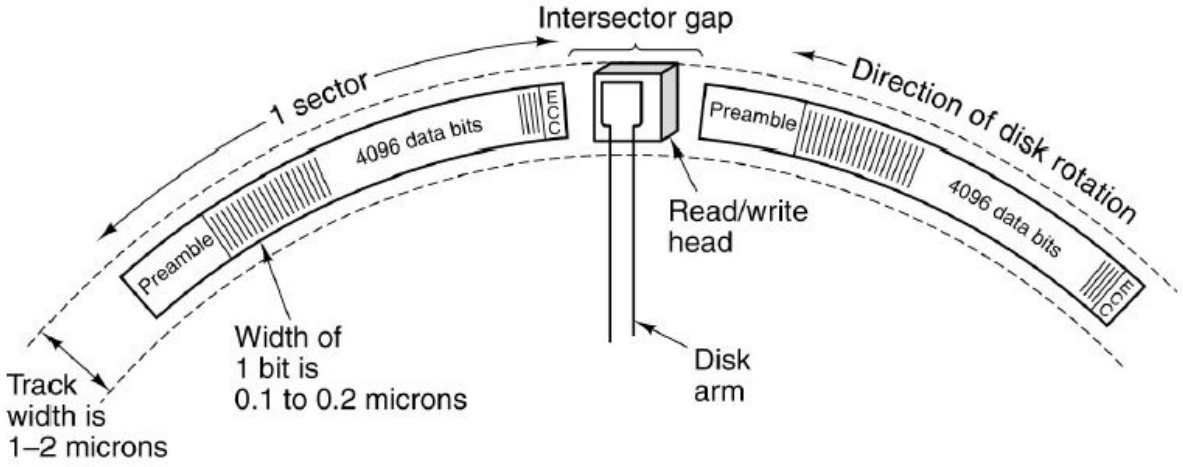

FIGURA 11 - PORZIONE DI UNA TRACCIA DEL DISCO

Tutti i dischi hanno ***bracci mobili*** in grado di spostarsi radialmente per posizionarsi su diverse distanze rispetto al centro di rotazione del piatto. La maggior parte dei dischi consiste in più piatti impilati verticalmente. Ciascuna superficie ha il proprio braccio con la propria testina. L'insieme di tracce alla stessa distanza dal centro è detto **cilindro**.

Le prestazioni del disco dipendono da diversi fattori. Oltre al *tempo di trasferimento*, sono fattori importanti altri due tempi: il tempo di **seek** ed la **latenza rotazionale**. Il *seek* è lo spostamento radiale che il braccio compie per posizionarsi nella traccia in cui sono presenti i dati che la testina dovrà leggere o in cui dovrà scrivere nuovi dati. La *latenza rotazionale* è invece il ritardo rappresentato dall'attesa che il settore desiderato ruoti sotto la testina. Il ritardo medio corrisponde a metà rotazione del piatto. Il tempo di *seek* e la *latenza rotazionale* sono i fattori con incidenza maggiore sulla velocità di accesso al disco e dominano sulla *velocità di trasferimento*. Pertanto leggere casualmente settori sparsi sul disco è un modo molto inefficiente di operare rispetto alla lettura di dati memorizzati in zone contigue.

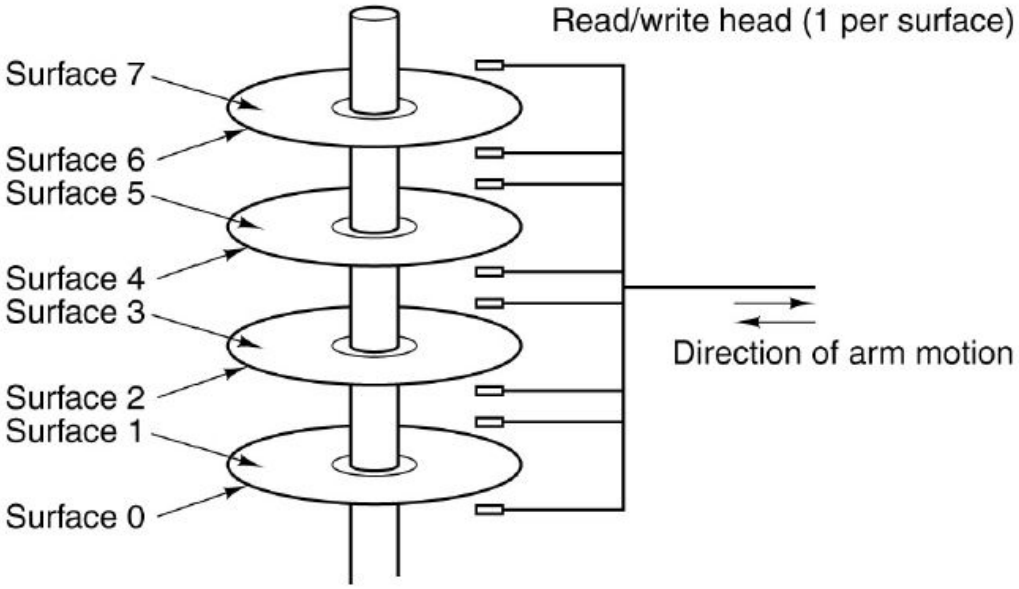

FIGURA 12 - DISCO CON QUATRO PIATTI

BUS

Un **bus** è un collegamento elettrico che unisce diversi dispositivi. Alcuni bus sono *interni* alla CPU e permettono di trasferire i dati da e verso la ALU, mentre altri sono *esterni* alla CPU e servono a connetterla con la memoria e con i dispositivi di input/output. Di seguito ci concentreremo sui bus esterni alla CPU.

I primi PC avevano un unico BUS esterno chiamato ***Bus su Sistema***. I computer odierni hanno invece un bus specifico per connettere CPU e memorie e almeno un altro bus per le periferiche.

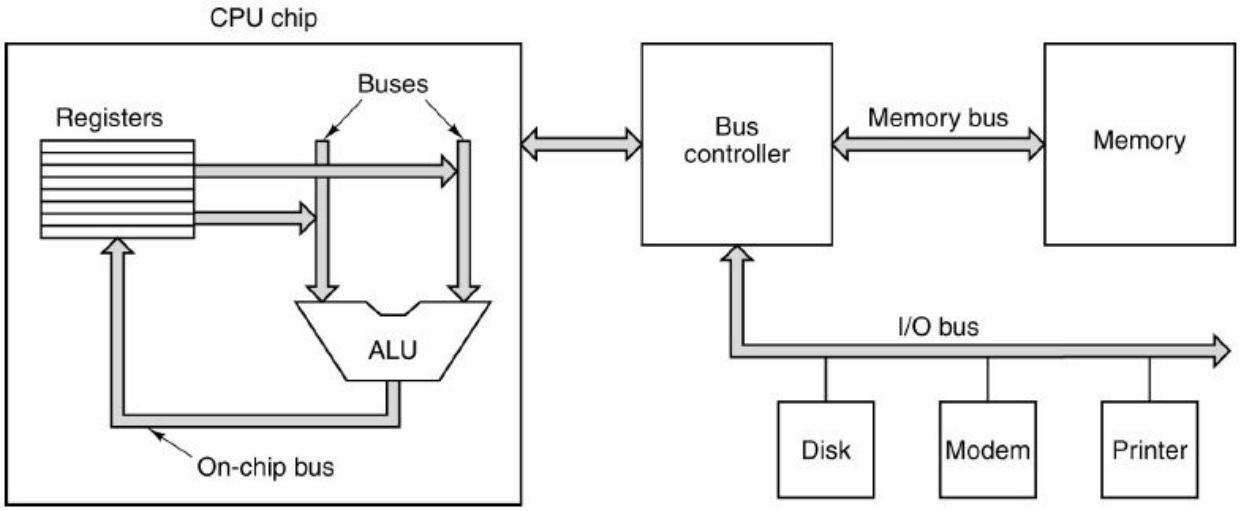

FIGURA 13 - UN CALCOLATORE CON PIÙ BUS

Il **bus di sistema** si suddivide in tre sottoinsiemi funzionali:

- **bus di controllo**: permette alla CPU di coordinare le attività del sistema; trasporta segnali e comandi provenienti dall'*unità di controllo* e diretti ai componenti del calcolatore. E' bidirezionale in quanto i componenti hardware inviano alla CPU i propri segnali di risposta.
- **Bus dati**: è la porzione di bus utilizzata per il trasferimento dei dati; è bidirezionale.
- **Bus degli indirizzi**: viene utilizzato per trasportare gli indirizzi di memoria in cui la CPU vuole scrivere o dai cui vuole leggere i dati. E' un bus unidirezionale.