Secondo capitolo

I processori

La **CPU** non è altro che il **cervello di un calcolatore,** la quale preleva le istruzioni contenuti nella memoria e le esegue.

È composto da due parti fondamentali:

- **Unità di controllo CU → preleva le istruzioni dalla memoria e le distingue per tipo**
- **Unità aritmetica-logica ALU → dati due dati prelevati dai registri, esegue operazioni di tipo aritmetico logico**

Inoltre, è composto da una **piccola memoria ad alta velocità,** necessaria per la memorizzazione temporanea dei dati e alcune informazioni di controllo. La memoria è composta da alcuni **registri,** ognuna delle quali ha una propria funzionalità e una dimensione predefinita. Tra i registri più importanti, ricordiamo:

- **Program counter(PC):** Punta alla successiva istruzione che dovrà essere prelevata per l'esecuzione.
- **instruction Register (IR):** Contenente l'istruzione che si trova in fase di esecuzione.

Generalmente, tutte le componenti di un computer sono collegate tramite **bus,** che non sono altro che un'insieme di cavi paralleli necessari per il trasferimento dei dati, indirizzi e segnali di controllo.

In una CPU di Von Neumann è presente il **datapath (percorso dati),** composto da

- **ALU;**
- **Shifter;**
- **Registri;**
- **Bus** per la comunicazione tra componenti;

Il percorso dati opera come segue:

- Vengono prelevati da due registri i dati e trasferiti nei registri di input;
- Dai registri di input, vengono trasferiti nell'ALU, nella quale verranno eseguite operazioni aritmetico-logiche (oppure nulla);
- L'output dell'ALU viene successivamente inviato allo shifter, un circuito digitale che esegue operazioni di spostamento dei bit;
- L'output dello shifter viene poi trasferito in un registro, che può essere anche lo stesso dal quale abbiamo prelevato.

La maggior parte delle istruzioni possono essere:

- **registro-registro →** descritto in precedenza, ovvero i dati necessari sono già memorizzati nei registri;
- **registro-memoria →**i dati necessari sono presenti in memoria, pertanto necessitiamo prima di prelevarli e memorizzarli in un registro; si vedrà successivamente che questi dati verranno memorizzati in registri specifici.

Esecuzione di un'istruzione - fasi

1. Viene prelevata l'istruzione indicata dal PC e inserita all'interno dell'IR;
2. Il PC punta all'istruzione successiva;
3. Determinare la tipologia di istruzione che dovrà essere eseguita;
4. Se l'istruzione usa una parola in memoria, determinare dove si trova ed, eventualmente, prelevarla e inserirla in un registro;
5. Esecuzione dell'istruzione;

6. Ricominciare dal punto 1.

Questo ciclo viene detto **fetch-decode-execute.**

Strategie di progettazione delle CPU

Negli anni '70, con la diffusione dei linguaggi di programmazione ad alto livello, emerse la necessità di ridurre il *gap semantico* tra il software e l'hardware. I programmatori scrivevano codice sempre più astratto, mentre l'hardware delle CPU era progettato per operare con istruzioni binarie molto basilari. Per colmare questa distanza, si svilupparono due filosofie di progettazione delle CPU: **CISC** e **RISC**.

- **RISC - Reduced Instruction Set Computer →** La CPU è in grado di eseguire poche istruzioni semplici in un solo ciclo di percorsi dati; rispetto ad un'architettura CISC, anche se impiegava più istruzioni per fare ciò che una macchina CISC potesse fare con una sola istruzione, le sue istruzioni erano 10 volte più veloci, in quanto non interpretate
- **CISC - Complex Instruction Set Computer →** La CPU è in grado di comprendere istruzioni complesse nativamente; nonostante a livello di prestazione è minore rispetto ad un'architettura RISC, si hanno programmi più compatti, riducendo il consumo di memoria, conseguenza di un minor numero di istruzioni per eseguire ciò che un'architettura RISC ne esegue con più istruzioni.

Successivamente, si cercò di incorporare i vantaggi di entrambe le architetture, in un **approccio ibrido:**

- Le CPU Intel contengono un sottoinsieme di istruzioni RISC più comuni, mentre le più complesse sono interpretate secondo la modalità CISC.

Principi di progettazione dei calcolatori moderni

Per la progettazione delle CPU, si tenta di seguire dei **principi di progettazione RISC:**

- **Tutte le istruzioni comuni sono eseguite direttamente dall'HW**, per garantire una maggiore velocità di esecuzione;
- **Massimizzare la frequenza di emissione delle istruzioni,** per una massimizzazione delle prestazioni gioca un ruolo fondamentale il **parallelismo;**
- **Le istruzioni devono essere semplici da decodificare:** per far ciò, le istruzioni devono essere regolari, di lunghezza predefinita e con un numero ridotto di campi/variabili;
- **Soltanto le istruzioni di LOAD e STORE possono far riferimento alla memoria,** mentre tutte le altre istruzioni devono operare solo sui registri;
- **Necessario un gran numero di registri:** poiché l'accesso in memoria è lento, è necessario memorizzare i valori letti dai registri, fin quanto non sono più utilizzati.

Parallelismo

Per ottenere prestazioni più elevate con una data velocità di clock, molti progettisti puntano sul parallelismo, che può essere di due tipologie:

- **a livello d'istruzione →** il parallelismo è sfruttato all'interno delle istruzioni per ottenere un maggior numero di istruzioni al secondo;
- **a livello di processore →** utilizzo di più CPU che lavorano contemporaneamente sullo stesso problema.

Parallelismo a livello d'istruzione: pipelining

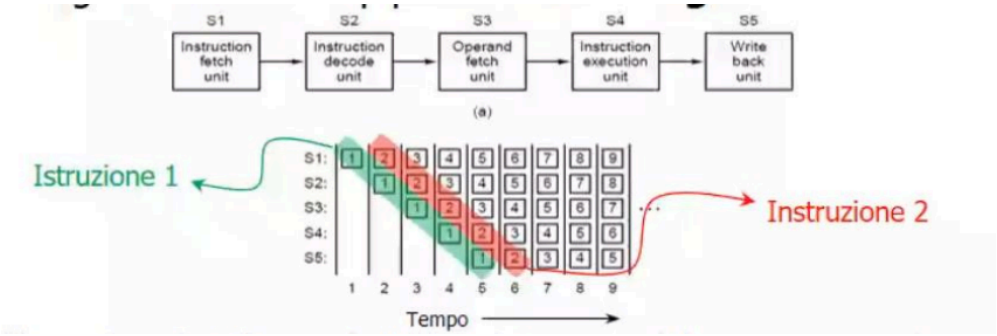
Ormai diamo per certo che uno dei problemi principali di una diminuzione della velocità di esecuzione è dovuto dal prelievo delle istruzioni dalla memoria. Per alleviare la problematica, i calcolatori sono stati dotati della capacità di prelevare in anticipo le istruzioni dalla memoria, in modo tale da averle a disposizione appena ne era necessario. Tali istruzioni venivano memorizzati all'interno di registri chiamati **buffer di prefetch.**

La **tecnica di prefetching** divide l'esecuzione dell'istruzione in due parti: il prelievo dell'istruzione e la sua esecuzione effettiva. Il concetto di **pipeline** spinge questa strategia oltre:

Invece di suddividere l'esecuzione in due parti, viene **diviso in più parti, in modo tale da essere eseguite in parallelo.**

Può essere considerata come una catena di montaggio: ogni fase esegue un'operazione di una differente istruzione, per poi, quando terminata, passarla alla fase successiva.

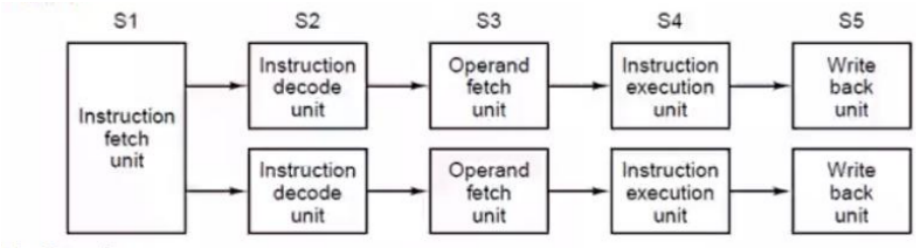
Nonostante migliori a livello di **latenza** (tempo che un'istruzione impiega per essere elaborata) e la **larghezza di banda del processore,** sorge una problematica: la pipeline non può essere impiegata nel caso in cui un'istruzione dipende dall'output di un'istruzione che ancora non è stata eseguita.



Processori con più pipeline: architetture superscalari

L'utilizzo di più pipeline non fa altro che migliorare ulteriormente le prestazioni: lo schema indicato in figura indica una singola unità di fetch che preleva due istruzioni alla volta e le inserisce nelle pipeline, ognuna delle quali è dotata di una ALU.

Affinché funzioni, non devono esserci conflitti nell'uso dei registri e nessuna delle due istruzioni deve dipendere dal risultato dell'altro.



Parallelismo a livello di processore

A differenza del parallelismo a livello d'istruzione, che comporta un miglioramento di prestazione **dai 5 a 10,** un parallelismo a livello di processore (quindi l'utilizzo di più CPU parallelamente), comporta un incremento **di 50, 100 e anche più.**

Esistono tre differenti approcci:

- **Computer con parallelismo sui dati;**
- **Multiprocessori;**
- **Multicomputer.**

Parallelismo sui dati

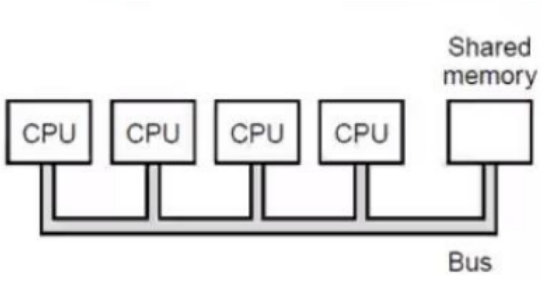
Il **parallelismo sui dati** è una tecnica di elaborazione in cui la stessa operazione viene eseguita contemporaneamente su più dati.

Secondo la classificazione di **Flynn**, i sistemi con parallelismo sui dati si basano su due concetti: **il flusso di istruzioni ed il flusso dei dati.** Tali sistemi si suddividono in quattro categorie:

- **SIMD (Single Instruction, Multiple Data)→** Esegue una **singola istruzione** simultaneamente su **molti dati;**
- **MIMD (Multiple Instruction, Multiple Data) →** Permette a più processori di eseguire **istruzioni diverse** su **dati diversi** in parallelo.
- **SISD (Single Instruction, Single Data) →** Un solo processore esegue **una singola istruzione** alla volta su un **unico flusso di dati**.
- **MIMD (Multiple Instruction, Multiple Data) →** Più processori lavorano in parallelo, eseguendo **istruzioni diverse** su **dati diversi.**

Multiprocessori

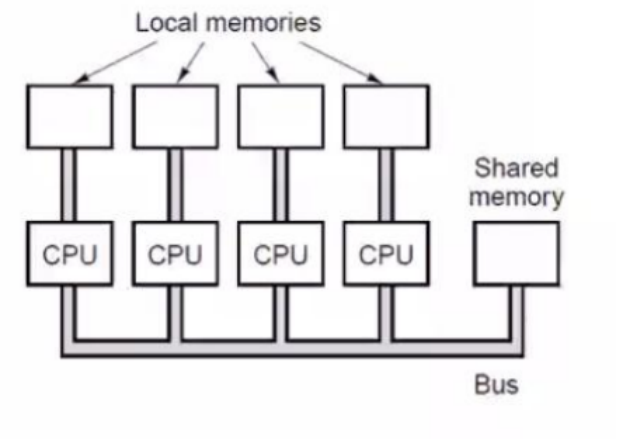
Un **multiprocessore** è un sistema composto da più CPU con **una memoria in comune**. Poiché ogni CPU può leggere e scrivere qualunque zona della memoria comune, le CPU devono sincronizzarsi per evitare di ostacolarsi a vicenda. Quando due o più CPU hanno la possibilità di interagire in modo così profondo si diche che sono *legate strettamente*.



Multicomputer

I multiprocessori hanno problematiche non indifferenti, tra cui il collo di bottiglia (poiché utilizzano lo stesso bus per comunicare con la memoria comune), oltre che conflitti che si verificano se i processori veloci tentano di accedere alla memoria.

Per questo motivo, si è pensato di abbandonare l'idea di memoria comune e accoppiare a ciascun CPU una memoria privata. In questo caso, le CPU di un multicomputer sono accoppiate in modo lasco e comunicano tra di loro tramite messaggi.



La memoria principale

La **memoria** è quella parte del calcolatore in cui sono depositati programmi e dati.

L'unità base della memoria è la cifra binaria, detta **bit,** che può avere valore 1 o 0.

Indirizzi di memoria

Le memorie sono costituite da un insieme di **celle o locazioni,** ciascuna delle quali può memorizzare informazioni. Ogni cella ha un numero, detto **indirizzo della cella,** tramite il quale il programma può riferirsi ad essa. Tutte le celle di una memoria contengono lo stesso numero di bit; se una cella è costituita da k bit, essa può contenere una delle 2^k combinazioni di bit.

La cella rappresenta la più piccola unità indirizzabile; ormai hanno standardizzato la dimensione di una cella a 8 bit (= 1 byte). I byte sono raggruppate in **parole**.

I bytes in una parola possono essere scritti:

- da sinistra a destra - **big endian**
- da destra a sinistra - **little endian**

Ad esempio: il numero $0A1F_h$ può essere scritto:

- in big endian = F1A0
- in little endian = 0A1F

Codici correttori - distanza di Hamming

Occasionalmente le memorie dei calcolatori possono commettere degli errori per via di picchi di tensione sulle linee di alimentazione o per altre cause. Per proteggersi da tali errori, alcune memorie utilizzano dei condici di correzione degli errori.

Supponiamo che una parola di memoria consista di **m bit di dati** ai quali aggiungiamo **r bit di controllo**; quindi abbiamo **n = m + r** la lunghezza totale. Tale unità è detta **parola di codice a n bit.**

Date due parole di codice, è possibile determinare il numero di bit corrispondente rispetto al quale differiscono. Per determinare il numero di bit diversi, calcoliamo l'XOR tra i bit delle due parole e contare quanti bit valgono 1 nel risultato.

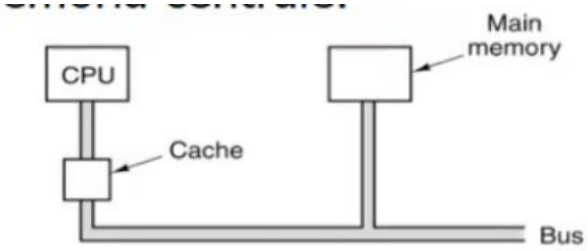
DA METTERE L'IMMAGINE DI ESERCIZIO!!!!

Memoria cache

Storicamente le CPU sono sempre state più veloci delle memorie. Tuttora, i progettisti delle CPU migliorano le prestazioni dei processori tramite architetture superscalari e pipeline; al contrario, i progettisti delle memoria incrementano le capacità di memorizzazione, svantaggiando la velocità. La differenza di prestazioni comporta quanto segue: quando la CPU lancia una richiesta alla memoria, essa non otterrà la parola desiderata se non dopo molti cicli di CPU. **Più lenta è la memoria, più cicli dovrà attendere la CPU.**

Attualmente il problema non è la tecnologia, ma a livello economico: è possibile costruire memorie veloci quanto le CPU ma, per poter andare alla stessa velocità, devono essere inserite sul chip della CPU. Inserire una memoria grande renderebbe il chip più grande, il chè significa che incrementerebbe la dimensione e il costo.

Nonostante ciò, esistono delle tecniche che permettono di combinare una piccola quantità di memoria veloce con una grande quantità di memoria lenta, al fine di velocizzare le prestazioni. Tale memoria piccola è detta **cache**. La memoria cache memorizza le parole più usate. Quando la CPU necessita di una parola, verifica inizialmente se è presente nella cache; se così fosse, la preleva da quest'ultima, altrimenti ne fa richiesta alla memoria centrale, la quale inserirà la parola nella cache eliminando una delle parole già presenti in base all'algoritmo di sostituzione delle pagine.

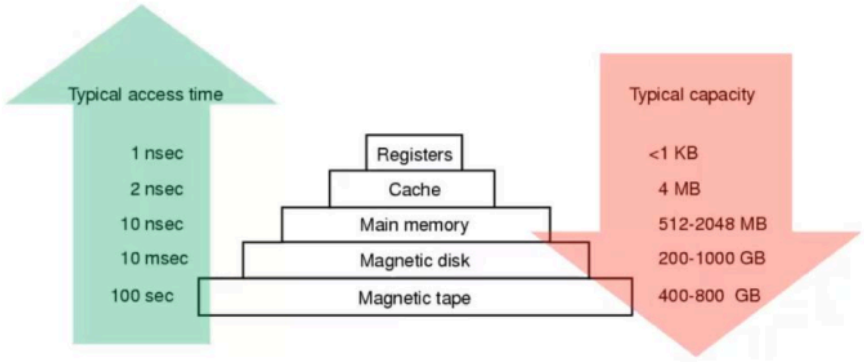


Assemblaggio e tipi di memoria

Un **gruppo di integrati,** tipicamente 8 o 16, è **montato** su un piccolo circuito stampato e venduto come **unità.** Questa unità viene chiamata **SIMM (Single Inline Memory Module)** quando ha una riga di connettori su un solo lato della scheda, oppure **DIMM (Double inline memory module)** quando ha due righe di connettori su entrambi i lati della scheda. Le SIMM riescono a trasferire solo 32 bits per ciclo di clock mentre le DIMM il doppio della banda.

Memorie secondarie

Gerarchia di memorie



Dischi magnetici

Un **disco magnetico** è composto come segue:

- **piatti:** uno o più piatti di alluminio rivestiti di materiale magnetico, originariamente dal diametro di 50cm, oggi tra 3 e 12 cm
- **testina:** contiene un **solenoide,** sfiora la superficie rimanendo sospesa su un cuscinetto d'aria. Una **corrente** che passa **attraverso la testina magnetizza** la superficie sottostante **orientando le particelle** in direzione opposta a seconda del verso della corrente. Quando la **testina passa sopra** una di queste aree magnetizzate, viene **indotta** nella stessa **una corrente positiva o negativa** che consente di **leggere i bit memorizzati.**
- **traccia:** sequenza circolare di bit scritti mentre il disco compie una rotazione completa. Ogni traccia è divisa in **settori** di lunghezza fissa (contenenti in genere 512 byte di dati), intervallati da **preamboli** (ciò consente alla testina di sincronizzarsi prima e dopo la scrittura o lettura); può capitare che due settori siano vicini tra loro, per ovviare a questo problema c'è uno **spazio tra settori** . Dopo i dati segue un **ECC** (Error Correction Code)**,** un **codice di Hamming**, oppure un **codice Reed-Solomon** (capace di correggere errori multipli).
- **Controllore del disco:** processore dedicato, che oltre a pilotare il dispositivo, accetta i comandi software (come READ, WRITE, FORMAT), di controllare il movimento del braccio, di rilevare e correggere errori, convertire gli 8 bit dei byte letti dalla memoria in uno stream seriale e viceversa. Alcuni controller gestiscono bufferizzazione di più settori, memorizzando in una cache i settori letti, oltre a gestire i settori contenenti errori.

Hard disk magnetico

Successivi rispetto ai floppy disk, gli **hard disk magnetici** consistono in **una serie di dischi impilati tra loro verticalmente.** Ciascun disco ha **entrambe le superfici trascrivibili** e quindi ciascun disco ha un totale di **2 testine.** L'**insieme di tracce poste alla stessa distanza dal centro** si chiama **cilindro.** Le prestazioni dei singoli dischi dipendono da un insieme di fattori:

- **Seek** (ricerca)**: spostamento radiale sulla posizione corretta** per poter leggere o scrivere
- **Latenza:** ritardo dovuto all'attesa che il settore desiderato ruoti sotto la testina
- **Tempo di trasferimento:** dipende da velocità rotazionale e dalla densità lineare
- **Burst rate:** velocità alla quale la testina può leggere i dati dopo essersi posizionata sul primo bit, velocità mantenuta per un singolo settore
- **Sustained rate:** velocità di lettura media nell'arco di alcuni secondi (compresi anche ricerca e latenze rotazionali).

RAID

Nel corso dell'ultimo decennio le prestazioni delle CPU sono aumentate in maniera esponenziale. Non possiamo dire la stessa cosa riguardo i dischi: con il tempo, la differenza di prestazioni tra CPU e dischi è aumentata in modo considerevole.

Come già visto, spesso si utilizza l'elaborazione parallela per velocizzare le prestazioni delle CPU. Si è pensato di applicare la stessa ideologia anche con l'I/O. L'idea, suggerita da Patterson negli anni '80, è di un'organizzazione a sei dischi per migliorare le prestazioni e l'affidabilità. Così nasce i RAID (Redundant Array of Indipendent Disks - "insieme di dischi econonici ridondanti "). Ovviamente esiste anche il "nemico" dei RAID, ovvero i SLED ("singolo disco capiente e costoso").

Tutti i RAID hanno la proprietà di distribuire i dati sulle diverse unità per permetterne la gestione parallela.

Esistono diverse tipologie di RAID, che vanno dal livello 0 al livello 6:

RAID livello 0

In questa organizzazione, il disco virtuale simulato dal RAID è suddiviso in strip ("strisce") di k settori ciascuna. L'organizzazione RAID scrive sulle strip in modo ciclico. Questo modo di distribuire dati su più unità è chiamato striping.

Il RAID di livello 0 lavora meglio quando le richieste sono di grandi dimensioni. Se la dimensione di una richiesta supera il numero di unità moltiplicato per la dimensione di una strip, allora alcune unità riceveranno più richieste, che verranno soddisfatte una dopo l'altra.

Tra i vantaggi abbiamo prestazioni eccellenti e di facile implementazione, con lo svantaggio di una diminuzione di affidabilità.

RAID livello 1

Diversamente dal RAID di livello 0, in questo caso si duplicano tutti i dischi: nell'esempio abbiamo 4 dischi primari e quattro di backup.

In caso di scrittura, ogni strip viene scritta due volte, mentre, nel caso di lettura, possiamo distribuire il carico di lavoro su più unità. Nonostante le prestazioni nella scrittura diminuiscono, non possiamo dire la stessa cosa in caso di lettura. Inoltre, la tolleranza ai guasti è eccellente: nel caso in cui si rompe un disco, possiamo utilizzare la sua copia.

RAID livello 2

Il RAID di livello 2 funziona sulla base di una parola o di un byte. Questo significa che ogni bit di dati viene distribuito su più dischi, e un disco dedicato memorizza il bit di parità per garantire la ridondanza. Tuttavia, a causa del gran numero di dischi necessari per implementare questa configurazione e delle sue limitazioni in termini di prestazioni, il RAID 2 non è comunemente utilizzato nelle configurazioni moderne.

RAID livello 3

È una versione semplificata del RAID di livello 2 e utilizza la parità a livello di byte per la ridondanza dei dati.

In questa configurazione, i dati vengono distribuiti su più dischi e un disco dedicato memorizza le informazioni di parità. Il RAID 3 è noto per le sue alte prestazioni di lettura sequenziale, ma ha limitazioni nelle prestazioni di scrittura parallela a causa della necessità di accedere al disco dedicato per la parità.

RAID livello 4

Il RAID di livello 4 è come il RAID di livello 0, con una parità strip-a-strip scritta su un disco aggiuntivo. Se ogni strip è lunga k byte, si calcola l'XOR fra tutte, ottenendo una strip di parità lunga k byte.

Se un disco si guasta è possibile ricalcolare i byte persi grazie al disco di parità. Questo schema protegge dalla perdita di un disco, ma ha prestazioni basse quando si aggiornano piccole quantità di dati: se si modifica una settore, è necessario leggere tutti i dischi, in quanto occorre ricalcolare e riscrivere la parità e, a partire da questi, calcolare quelli nuovi.

Il disco di parità può diventare un collo di bottiglia a causa del grande carico di lavoro che pesa su di esso.

RAID livello 5

Il RAID di livello 5 elimina il collo di bottiglia distribuendo i bit di parità su tutti i dischi in maniera uniforme, in modalità round robin. Nonostante ciò, la ricostruzione del suo contenuto dopo un guasto a un disco, è un'operazione abbastanza delicata.

RAID livello 6

Il RAID 6 è un livello che si concentra maggiormente sulla ridondanza dei dati offrendo più protezione rispetto al RAID 5. Questa configurazione utilizza una tecnica di parità a doppia distribuzione, che garantisce la tolleranza di due guasti simultanei. In parole povere, il RAID 6 può sopportare la perdita di due dischi rigidi senza perdita di dati.

Nel RAID 6, i dati vengono divisi e distribuiti tra gli N+2 dischi rigidi del sistema, dove N rappresenta il numero minimo di dischi richiesti per la configurazione RAID. Vengono calcolate due funzioni di parità, che consentono di ricostruire i dati persi in caso di guasto dei dischi. Ciò significa che il RAID 6 offre una protezione dei dati più robusta rispetto al RAID 5, ma richiede più spazio di archiviazione per le funzioni di parità.

Nonostante ciò, presenta degli svantaggi:

1. La scrittura dei dati può risultare più lenta rispetto ad altre configurazioni RAID a causa del calcolo delle funzioni di parità.
2. Requisiti di spazio di archiviazione più elevati rispetto al RAID 5 a causa della presenza di due funzioni di parità.

Dischi a stato solido (SSD)

Basati su memoria flash non volatile, gli **SSD** (Solid State Disk), alternativa alle tecnologie a disco magnetico.

- **pro**: Non hanno problemi meccanici, usando componenti elettroniche per la memorizzazione, più veloce
- **contro**: I **transistor si deteriorano nel corso del tempo**. Una delle cause può essere l'iniezione a caldo (hot carrier injection); una carica di elettroni viene rinchiusa all'interno del transistor, lasciandolo in uno stato permanente di blocco, acceso o spento.

Input/Output

Il bus

Il bus è un sistema di comunicazione utilizzato per collegare la CPU, la memoria e i dispositivi di I/O.

Ogni dispositivo di I/O ha un controller che gestisce il loro accesso al bus. Quando un controllore legge e scrive dati in memoria senza intervento della CPU, si dice che effettua un DMA (Direct Memory Access). Solitamente, una volta che il trasferimento è completato, il controllore produce un interrupt che obbliga la CPU a sospendere immediatamente l'esecuzione del programma corrente e a far iniziare una procedura detta **gestione dell'interrupt.**

Quando due dispositivi necessitano di utilizzare lo stesso bus nello stesso tempo, un chip detto **arbitro del bus** stabilisce i turni. Solitamente, le periferiche di I/O hanno precedenza sulla CPU, in quanto, nel caso in cui si fermassero, perferebbero dati. Quindi, quando una periferica chiede l'accesso al bus, questo gli viene subito concesso. Tale processo è detto **furto di cicli** e rallenta le prestazioni del calcolatore.

Generalmente, un bus interno è suddiviso in

- **bus dati →** trasmette informazioni tra componenti interne del computer;
- **bus di controllo →** utilizzati per stabilire chi può trasmettere dati sul bus dati, indicare la tipologia di operazione, la dimensione dei dati trasmessi, etc.
- **bus indirizzi →** specifica la posizione fisica dove i dati devono essere letti o scritti.

Dispositivi di I/O (indico solo quelle che possono esserci alle domande d'esonero)

Stampanti

Una stampante è un dispositivo di uscita che permette di trasferire su un supporto (tipicamente carta) i documenti elettronici.

Ne esistono di varie tecnologie:

- **a matrice** → utilizzano una testina che contiene una matrice di punzoni elettromagnetici che scorre longitudinalmente, mentre il foglio le scorre sotto.

  La qualità di stampa può essere incrementata utilizzando un maggior numero di aghi sulla testina oppure ripassando più volte le linee di stampa (overlapping). Normalmente, queste stampanti sono monocromatiche, economiche, lente, rumorose e a bassa qualità bassa; nonostante ciò, hanno il vantaggio che il nastro si usura meno rispetto alle cartucce di una stampante a getto d'inchiostro o laser.

- **a getto d'inchiostro →** Hanno un principio di funzionamento simile a quelle ad impatto: la testina scorre longitudinalmente mentre il foglio scorre. La testina mobile di stampa contiene le cartucce d'inchiostro e l'inchiostro è spruzzato da piccoli ugelli.

  Esistono due differenti tecnologie:

  - le piezoelettriche → hanno uno speciale tipo di cristallo che si deforma con la tensione elettrica e spinge l'inchiostro fuori la testina (> tensione → > quantità d'inchiostro;
  - le termiche → contengono una piccola resistenza dentro ogni ugello, la tensione è utilizzata per scaldare la resistenza che incrementa la pressione dell'inchiostro e lo spinge fuori l'ugello.

  Hanno il vantaggio di essere economiche, silenziose, hanno una buona qualità, ma sono lente e hanno un costo di manutenzione elevato

- **laser →** Combinano alta qualità di stampa, velocità, silenziosità e costi moderati. Queste stampanti utilizzano una tecnologia simile a quella delle macchine fotocopiatrici. Il cuore della stampante è un tamburo di precisione rotante.

  All'inizio di ciascun ciclo di pagina, il tamburo è caricato fino a 1000 V e rivestito con materiale fotosensibile. La luce del laser colpisce longitudinalmente il tamburo e fa perdere carica ad alcune regioni. Il tamburo ruota ed attrae sulla linea del toner elettrostaticamente sensibile.