Architettura di un elaboratore: insieme caratteristiche visibili al programmatore, attributi che hanno impatto diretto sulla logica di esecuzione di un programma (istruzioni, numero di bit usato per rappresentare i dati, meccanismi di I/O, tecniche di indirizzamento della memoria). **Organizzazione di un elaboratore**: aspetti hardware trasparenti al programmatore (interfacce tra calcolatore e periferiche, segnali di controllo, tecnologia delle memorie).

Calcolatore è un insieme di componenti connesse tra loro, con un insieme di sottoinsiemi tra loro correlati. Per ogni livello esiste una **struttura** (come sono correlati i componenti) e una **funzione** (cosa fa ciascun componente). **Struttura calcolatore**: CPU (unità centrale di elaborazione), memoria centrale, input e output, interconnessioni. **Funzioni calcolatore**: elaborazione dati, memorizzazione dati, trasmissione dati, controllo delle funzioni precedenti. **Struttura e funzioni CPU**: unità di controllo (controlla la sequenza delle operazioni), ALU unità aritmetico-logica (elaborazione dati), registri (memoria interna della CPU), interconnessioni (comunicazione). **Struttura unità di controllo**: logica di sequenzializzazione, registri di controllo e decodificatori, memoria di controllo.

Architettura Von Neumann: la progettazione di quasi tutti i calcolatori odierni è basata su concetti sviluppati da Von Neumann, è basata su 3 concetti**:** dati e istruzioni risiedono in un'unica memoria di lettura e scrittura, contenuti di questa memoria sono accessibili per indirizzo, l'esecuzione avviene in modo sequenziale, da un'istruzione a quella immediatamente successiva. Per eseguire un programma, possiamo costruire i componenti logici in modo che il risultato sia quello voluto. Questo è il modo di costruire un "programma cablato", cioè in forma hardware, che non può essere modificato. Esso è un sistema non flessibile, che può eseguire solo istruzioni determinate. Tramite circuiti generici accetta segnali di controllo e produce risultati. Per nuovi programmi basta quindi dare i giusti segnali di controllo. La programmazione invece è molto più facile perché invece che ridefinire ogni volta l'hardware basta fornire una nuova sequenza di codici (ossia una nuova istruzione). Questo processo viene detto "programmazione software".

IR (istruction register): contiene codice operativo dell'istruzione correntemente in esecuzione. **MBR** (memory buffer register): contiene parola che deve essere immagazzinata in memoria o è letta dalla memoria. **MAR** (memory address register): contiene indirizzo di parola in cui scrivere contenuto di MBR o che deve essere trasferita in MBR. **IBR** (istruction buffer register): contiene temporaneamente l'istruzione destra di una parola di memoria. **PC** (program counter): contiene l'indirizzo della prossima coppia d'istruzioni da caricare dalla memoria. **AC** (accumulator) e **MQ** (multiplier quotient): contengono operanti e risultati delle operazioni della ALU.

Descrivere il ciclo completo di fetch/execute delle istruzioni: (Fetch) All'inizio di ogni ciclo esecutivo il processore legge un'istruzione dalla memoria. (Calcolo indirizzo istruzione) Dopo il prelievo di tale istruzione il processore incrementa il valore del registro PC (Program Counter) in modo che la prossima istruzione eseguita sia quella posizionata all'indirizzo di memoria immediatamente successivo. L'istruzione viene prima caricata in un registro del processore noto come IR (instruction register) e poi analizzata per determinare il tipo di operazione da eseguire (Decodifica istruzione). In seguito si determina l'indirizzo dell'operando e subito dopo avviene la sua lettura (Calcolo indirizzo operando/lettura operando). Infine viene eseguita l'operazione indicata nell'istruzione e il risultato viene scritto nella memoria e viene inviato alla periferica. (Operazione sui dati/memorizzazione risultato).

Bus di sistema: mezzo di comunicazione che collega le principali componentistiche dell'elaboratore. (Processore, memoria e unità di I/O). Formato da più percorsi di comunicazione chiamate **linee**, ognuna delle quali è in grado di trasmettere segnali che rappresentano una cifra binaria (bit 0 o 1). Il numero di linee determina quanti bit si possono traportare contemporaneamente (1 bit per linea). Posso essere di 3 tipi le linee: **dati**, **indirizzi** (decidere dove instradare i dati presenti nel bus), **controllo** (per controllare l'accesso e l'uso delle altre 2 linee, trasmettono sia comandi che specificano le operazioni da compiere che segnali di temporizzazione). Il numero di linee per tipo è detto ampiezza del bus. Se un modulo desidera inviare dati a un altro deve prima di tutto ottenere l'uso del bus e poi trasferire i dati mentre se desidera richiedere i dati deve, dopo aver ottenuto l'uso del bus, prima trasferire una richiesta all'altro modulo sulle appropriate linee di indirizzo e controllo e poi attendere l'invio dei dati.

Gestione I/O tramite DMA: (direct memory access) è un modulo hardware addizionale che sostituisce la CPU per la maggior parte delle attività di I/O. Esso si occupa del trasferimento dei dati da o verso la memoria

senza passare attraverso il processore e invia a quest'ultimo un segnale di interrupt quando il trasferimento è completato. In questo modo la CPU è coinvolta solo all'inizio e alla fine del trasferimento e nel frattempo può eseguire altre attività. Prima di delegare una determinata operazione di I/O al DMA, il processore gli comunica le seguenti informazioni: lettura/scrittura, indirizzo dispositivo interessato, indirizzo iniziale in memoria del blocco dati coinvolto nell'operazione, quantità di dati da trasferire. Il DMA è connesso al bus di sistema e può accedere al canale dati in 2 modi. Una parola alla volta, sottraendo di tanto in tanto alla CPU il controllo del canale (cycle stealing) o per blocchi, prendendo in possesso il canale per una serie di trasferimenti (burst mode). Una configurazione ideale per consumare meno cicli di bus è integrare le funzioni di DMA e I/O, utilizzando il bus solo per scambiare dati con la memoria, mentre comunica in altri modi con i moduli I/O.

Gerarchia di memoria, realizzazione del mapping dei blocchi: diretto (direct mapping), associativo (n-way set) e set associativo (n-way set associative). L'indirizzamento diretto è la tecnica più semplice in quanto assegna ad ogni blocco una sola possibile linea di cache. Per accedere alla cache poi, ogni indirizzo in memoria centrale viene diviso in tre campi: tag, linea, parola. Questo metodo di traduzione si distingue per la semplicità con cui quest'ultima avviene da indirizzo ILI (memoria) ad ILS (cache) e per la veloce determinazione di hit o miss. D'altro canto necessita di contraddistinguere il blocco presente in cache tramite un'etichetta (tag) e porta inoltre ad un numero elevato di swap per accedere ai dati di blocchi adiacenti. Il secondo metodo invece supera lo svantaggio dell'indirizzamento diretto poiché ogni blocco della memoria centrale può essere caricato su qualsiasi linea della cache. Così facendo l'indirizzo di memoria presenta solamente due campi: tag e parola e garantisce nel complesso una massima efficienza di allocazione. L'unico svantaggio è dato dalla complessità circuitale richiesta per esaminare in parallelo i tag di tutte le linee di cache. L'indirizzamento set associativo invece è un compromesso che unisce i punti di forza dei precedenti riducendo i loro svantaggi. Qui la cache è suddivisa in insiemi (set) di k linee e il blocco può essere assegnato a qualunque linea dell'insieme, l'indirizzo in memoria è suddiviso nei campi tag, set e parola. Questo tipo di indirizzamento vanta una buona efficienza di allocazione a fronte di una discreta complessità di ricerca.

Modalità e granularità trasferimento delle informazioni fra i vari livelli della gerarchia di memoria: scendendo lungo la gerarchia, troviamo un decrescente costo per bit, una capacità crescente, un tempo di accesso crescente e una frequenza di accesso decrescente da parte del processore. In questo modo, memorie più piccole, più costose e più veloci sono integrate da memorie più grandi, economiche e più lente. La CPU utilizza direttamente il livello più alto della gerarchia. Ogni livello inferiore, deve contenere tutti i dati presenti ai livelli superiori (più altri). La dimensione di un blocco è la quantità minima indivisibile di dati che occorre prelevare (e copiare) dal livello inferiore. L'indirizzo di un dato diviene l'indirizzo del blocco che lo contiene, sommato alla posizione del dato all'interno del blocco.

Perché la memoria viene suddivisa in blocchi e, relativamente alle prestazioni della cache, discutere pregi e difetti dell'adozione di una dimensione di blocco elevata: per realizzare un'organizzazione gerarchica della memoria, che soddisfi i parametri di velocità, ampiezza e costo, conviene suddividere la memoria in blocchi. La dimensione di un blocco è la quantità minima indivisibile di dati che occorre prelevare (copiare) dal livello inferiore. L'indirizzo di un dato diviene l'indirizzo del blocco che lo contiene sommato alla posizione del dato all'interno del blocco. Se si adotta un blocco con dimensione elevata si guadagna in termini di costi e capacità ma si perde in termini di velocità.

Politiche di scrittura write through e write allocate: **Write through**: ogni dato modificato nella cache viene contemporaneamente modificato nella memoria centrale. In questo modo i dati sono sempre coerenti tra i vari livelli di memoria. Lo svantaggio è che per frequenti scritture sul medesimo blocco si verifica un aumento di traffico nel bus con conseguente collo di bottiglia. **Write back**: la scrittura in memoria centrale avviene solo quando il corrispettivo blocco in cache viene rimpiazzato. Ciò consente un ottimizzazione del traffico tra livelli ma causa periodi di incoerenza tra di essi, inoltre occorre sempre ricordare se sono avvenute operazioni di scrittura nel blocco tramite "dirty bit". Il problema di questa tecnica è che parti della memoria centrale non sono aggiornate, e dunque gli accessi tramite moduli I/O possono essere consentiti solo attraverso la cache.

Politica di scrittura write back: tecnica alternativa alla write through che evita il collo di bottiglia minimizzando le scritture in memoria e applicando gli aggiornamenti solo nella cache. Il problema è che parti

della memoria centrale non sono aggiornate, e dunque gli accessi tramite moduli I/O possono essere consentiti solo attraverso la cache (circuiti complicati e possibili colli di bottiglia). Oltre a ciò in un organizzazione a bus in cui più di un dispositivo è connesso e dove la memoria centrale è condivisa un'alterazione dei dati in una cache può invalidare tutti i dati nella memoria centrale e anche quelli nelle altre cache eventualmente connesse al bus. Per ovviare al problema esistono 3 approcci: monitoraggio del bus con write through (i controllori della cache osservano le linee di indirizzi ed invalidano eventuali scritture in una locazione di memoria condivisa da parte di qualche gestore del bus qualora tale locazione di memoria sia già residente nella cache), trasparenza hardware (mediante hardware aggiuntivo aggiornare costantemente la memoria centrale e tutte le cache presenti), memoria noncachable (porzione condivisa nella quale gli accessi sono dei cache miss dato che essa non viene mai copiata nella cache, può essere identificata via hardware o tramite indirizzi riservati).

Come sono organizzate le info in un CD-ROM: è un dispositivo ottico non cancellabile che sfrutta una serie di pozzetti (pit) per memorizzare dati binari su una superfice di policarbonato. Tali informazioni, scritte da un laser ad alta intensità, vengono poi recuperate da un laser a bassa potenza. Le informazioni sono organizzate su una traccia a spirale che inizia dal centro. I settori in prossimità dell'estremità del disco sono della stessa lunghezza di quelli interni. Le informazioni sono quindi impacchettate uniformemente sul disco in segmenti della stessa dimensione e questi vengono letti ruotando il disco a velocità variabile. I pit sono poi letti dal laser a velocità lineare costante.

Differenza tra interruzioni multiple e annidate: per trattare le interruzioni si possono usare due approcci, il primo è disabilitare gli interrupt durante l'elaborazione di un interrupt, questi interrupt disabilitati rimarranno pendenti fino alla riabilitazione da parte della CPU e verranno gestiti in sequenza senza quindi tener conto di eventuali priorità da parte di qualcuno di essi. Il secondo invece consiste nel definire delle priorità e consentire quindi ad un interrupt di priorità maggiore di essere trattato subito, interrompendo quindi la gestione di un interrupt a priorità inferiore.

Descrivere ciclo esecuzione con trattamento delle interruzioni: all'inizio di ogni ciclo esecutivo il processore legge un'istruzione dalla memoria. Dopo incrementa il valore del registro. L'istruzione viene caricata in un registro del processore noto come IR e poi analizzata per determinare il tipo di operazione da eseguire e l'operando da utilizzare. Si determina l'indirizzo dell'operando e dopo avviene la sua lettura. Infine viene eseguita l'operazione indicata nell'istruzione e il risultato viene scritto nella memoria e viene inviato alla periferica. Il processore controlla se è avvenuto qualche interrupt, se non vi sono interrupt pendenti, il processore procede al ciclo di prelievo e legge la successiva istruzione del programma altrimenti: sospende l'esecuzione del programma in esecuzione salvandone il contesto (memorizzando quindi l'indirizzo della prossima istruzione ed altri dati rilevanti) imposta il PC all'indirizzo di partenza di una routine per la gestione dell'interrupt. Il processore poi procede al ciclo di fetch e legge la prima istruzione del programma di gestione dell'interrupt. Quando questa routine termina, il programma riprende l'esecuzione del programma utente dal punto dell'interruzione.

Gestione interruzioni (hw e sw) nel caso di I/O input driven: per evitare che il processore debba costantemente verificare lo stato del dispositivo di I/O esso invia al modulo un comando e quest'ultimo interromperà il processore una volta pronto allo scambio di dati, dopodiché il processore esegue lo scambio e riprende poi l'elaborazione interrotta. A livello hardware, quando una periferica completa un'azione di I/O: il dispositivo invia un interrupt al processore, il processore conclude l'operazione corrente e poi controlla l'eventuale presenza di interrupt e se c'è riscontro, invia un segnale di riconoscimento al dispositivo che ha inviato tale interrupt che rimuoverà quindi il proprio segnale. Il processore poi salva il contesto del programma ponendo PSW (lo stato del processore) e la locazione della prossima istruzione da eseguire contenuta nel PC in cima alla pila di sistema. Scrive nel PC l'indirizzo della prima istruzione della routine di gestione dell'interrupt considerato. Il controllo viene trasferito al programma di gestione dell'interrupt: vengono salvate le restanti informazioni di stato del processo dai registri PSW e PC, avviene l'elaborazione dell'interrupt (un esame dell'informazioni sullo stato delle operazioni di I/O oppure invio di ulteriori comandi al dispositivo di I/O). Quando l'elaborazione dell'interrupt è completa, i valori dei registri salvati vengono recuperati dalla pila e riportati nei registri ed infine i valori di PSW e PC vengono ripristinati.

Caratteristiche e operazioni principali delle memorie a semiconduttore (sia RAM che ROM): l'elemento base delle memorie a semiconduttore è la cella di memoria che presenta specifiche proprietà: due stati stabili, che rappresentano il bit 0 e 1, la possibilità di scrivere nella cella per impostare lo stato, la possibilità di leggere lo stato, accesso casuale (si accede alla parola tramite circuito dedicato). Il più comune è la memoria **RAM** (volatile) la cui caratteristica principale è la possibilità di leggere e scrivere dati in e da memoria in modo semplice e rapido tramite segnali elettrici. Due tipologie SRAM e DRAM. **DRAM** (Dynamic RAM) è composta di celle che memorizzano i dati sotto forma di cariche nei condensatori. L'assenza o presenza di cariche determina lo stato 0 o 1. Poiché i condensatori tendono a scaricarsi naturalmente dopo un certo periodo di tempo necessitano di un refresh periodico e da ciò deriva l'aggettivo "dinamiche" per il fatto che la carica scompare dinamicamente anche in presenza di alimentazione. **SRAM** utilizzano invece gli stessi elementi base di un processore, infatti i valori binari vengono memorizzati mediante porte logiche e non necessitano di un refresh delle cariche. Le SRAM sono più veloci delle DRAM e vengono usate per la memoria cache mentre le DRAM per la memoria centrale. Un altro tipo di memoria ad accesso casuale è la **ROM** (read only memory) presenta uno schema di dati predefinito e non modificabile, non volatili e di sola lettura. Il vantaggio di questo memorie è che i dati e programmi di piccole dimensioni possono essere scritti in esse e non devono essere caricati da dispositivi esterni. I dati vengono inseriti durante il processo di fabbricazione processo costoso che richiede poi grande precisione poiché l'errata scrittura di un bit causa la perdita di tutto il lotto di ROM. Esse possono dividersi in **PROM** (programmabile): che viene utilizzata qualora sia necessario un particolare contenuto, non è volatile ed è scrivibile solo una volta. Un'altra variante di queste memorie è la memoria principalmente di lettura utile quando le operazioni di lettura sono molto più frequenti di quelle di scrittura ed è richiesta una memoria non volatile. Si dividono in 3 tipi: **EPROM**, **EEPROM**, **flash**. EPROM sono memorie cancellabili e programmabili otticamente e vengono lette e scritte elettricamente come le PROM. Prima della scrittura le celle di memoria devono essere cancellate e portate tutte allo stesso stadio mediante esposizione ai raggi UV. Le EEPROM sono memorie cancellabili e programmabili elettricamente, in esse è possibile scrivere in qualunque momento senza cancellare i dati, aggiornando solo i byte indirizzati (la scrittura richiede molto più tempo della lettura e sono più costose e meno dense delle EPROM). Le memorie flash invece sono chiamate così per la velocità con la quale possono essere riprogrammate. Così come le EEPROM adottano un sistema di cancellazione elettrica e la sua memoria può essere eliminata molto più velocemente di una EPROM ed è inoltre possibile cancellare solo alcuni blocchi piuttosto che tutta la memoria.

Descrivere le memorie DRAM: (dynamic) è composta di celle che memorizzano i dati sotto forma di cariche nei condensatori. L'assenza o presenza di cariche determina lo stato 0 o 1. Poiché i condensatori tendono a scaricarsi naturalmente dopo un certo periodo di tempo le ram necessitano di un refresh periodico e da ciò deriva l'aggettivo "dinamiche" per il fatto quindi che la carica scompare dinamicamente anche in presenza di alimentazione. Per le operazioni di scrittura si applica tensione alla linea di bit (che a seconda dell'intensità determina se bit è 1 o 0) successivamente si applica segnale alla linea degli indirizzi trasferendo la carica al condensatore. Per quelle di lettura invece prima di tutto si seleziona la linea indirizzo poi la carica viene convogliata in una linea di bit collegata ad un amplificatore che confronta la tensione con un valore di riferimento e determina se la cella contiene il bit 1 o 0 e infine si applica un refresh per ripristinare la carica nel condensatore.

Tipi di miss e strategie per diminuirli: miss di primo accesso, inevitabile e non riducibile, miss per capacità insufficiente, quando la cache non può contenere tutti i blocchi necessari all'esecuzione del programma e miss per conflitto, quando più blocchi possono essere mappati (con associazione diretta o a gruppi) su uno stesso gruppo. Risoluzione per i miss per capacità insufficiente: maggiore dimensione del blocco la quale è una buona tecnica per fruire di località spaziale che però causa un aumento di miss per conflitto (a causa del numero ridotto di blocchi disponibili). Per i miss per conflitto: maggiore associatività che causa però un incremento del tempo di localizzazione in gruppo ed è soggetta alla regola del 2:1 (cache di N blocchi stessa probabilità di miss di cache a N/2 con associazione a 2 vie). Altre tecniche di risoluzione possono essere l'adozione di una cache multilivello, la separazione di cache dati e cache istruzioni e l'ottimizzazione dei dati mediante compilatori che permettono il posizionamento accurato delle procedure ripetitive, la fusione di vettori in strutture (località spaziale) e la trasformazioni di iterazioni annidate (località spaziale).

Discutere il sistema RAID. Descrivere 0,1,2,3,4,5,6 RAID: Il sistema RAID (redundant array of indipendent disk) venne elaborato per aumentare le prestazioni di un calcolatore (usando più dischi in parallelo), e per avere ridondanza, una maggiore affidabilità poiché i dati vengono distribuiti su più dischi e sono recuperabili in caso di guasto. **Raid 0** non include la ridondanza a favore delle prestazioni, i dati sono distribuiti sui vari dischi quindi se vi sono due richieste di blocchi che si trovano in dischi diversi esse vengono servite in parallelo. I dati vengono distribuiti in strisce (strips) che possono essere blocchi fisici, settori o altro e sono distribuite a rotazione (round Robin). **Raid 1** la ridondanza viene ottenuta duplicando tutti i dati, questo implica un numero doppio di dischi fissi (mirroring). Si usa lo striping dei dati e ogni striscia si trova fisicamente su due dischi quindi una lettura può essere soddisfatta dal disco al quale si accede più velocemente e che presenta la minor latenza di rotazione, la scrittura avviene su entrambi i dischi quindi le prestazioni sono condizionate dalla più lenta delle due scritture. **Raid 2** (non commercializzata) sfrutta l'accesso parallelo ossia le testine di ciascun disco si trovano nella stessa posizione in ogni momento. Si usa lo striping dei dati con strisce molto piccole, vengono generati codici di correzione errori di Hamming memorizzati in più dischi, perciò se si riscontra un errore il controllore può riconoscere e correggere l'errore istantaneamente in modo che il tempo di accesso per la lettura non venga rallentato. **Raid 3** simile al RAID 2 ma interviene sul problema della sincronizzazione. Solo un disco ridondante, indipendentemente dal numero di dischi. Invece dei codici di correzione utilizza dei semplici bit di parità per ogni insieme corrispondente di bit. Dati presenti su un disco difettoso possono essere ricostruiti a partire dai dati sui dischi rimanenti e dalle informazioni sulla parità. Velocità di trasferimento molto alta. **Raid 4** invece ogni disco opera indipendentemente, così facendo le richieste di I/O possono essere gestite contemporaneamente. Usa lo striping dei dati con strisce grandi e utilizza una striscia di parità bit a bit sulle corrispondenti strisce di ciascun disco dati, e i bit di parità sono memorizzati nella corrispondente striscia sul disco di parità, risulta penalizzato dalle richieste di scrittura di piccola dimensione. **RAID 5** simile a RAID 4 ma l'allocazione dei blocchi di parità con round Robin è distribuita su tutti i vari dischi ed evita il collo di bottiglia. Soluzione comunemente utilizzata sui server di rete. **RAID 6** per ragioni di sicurezza si possono avere due parità calcolate in modo diverso. Memorizzata in blocchi separati su dischi differenti. Se l'utente richiede N dischi, ne occorrono N+2. Alta affidabilità sui dati.

Descrivere l'organizzazione e formattazione dei dati nei dischi rigidi: nei dischi magnetici la testina è in grado di leggere e scrivere su una porzione del disco rotante. Ciò origina la disposizione fisica dei dati in anelli concentrici, chiamati tracce (track). Tracce adiacenti sono separate da spazi (gaps) per evitare errori dovuti al disallineamento della testina o ai campi magnetici. Il trasferimento dati avviene per settori, che generalmente sono un centinaio per traccia, di lunghezza fissa o variabile, i settori adiacenti sono inoltre separati da spazi. I bit più vicini al centro del disco ruotano attorno al punto fisso, come la testina, più lentamente dei bit esterni, questa velocità viene quindi compensata per permettere alla testina di leggere tutti i bit alla stessa velocità facendo ruotare il disco a velocità angolare costante. La formattazione è l'operazione con la quale si prepara il disco per renderlo idoneo all'archiviazione: inserimento di un criterio che determini la posizione di un dato settore il suo inizio e la sua fine e un punto di partenza sulla traccia. Questi requisiti sono rispettati tramite dati di controllo memorizzati sul disco con i quali esso viene inizializzato.

Descrivere gestione da programma I/O: i dati vengono scambiati tra processore e modulo I/O. Quando il processore sta eseguendo un programma e incontra un'istruzione correlata con l'I/O, esso esegue l'istruzione inviando un comando al modulo di I/O appropriato. Il modulo eseguirà l'azione richiesta e imposterà i bit appropriati nel suo registro di stato. Il modulo non intraprende nessun'altra azione per allertare il processore. In particolare, esso non interrompe il processore, perciò il processore periodicamente controlla lo stato del modulo I/O finché non rileva il completamento dell'operazione. Ci sono 4 tipi di comandi che il processore può inviare al modulo I/O e sono Controllo (avvia una periferica e le dice cosa fare), test (testa le condizioni di stato dei moduli di I/O), lettura (ottiene i dati dalla periferica attraverso il modulo I/O) e scrittura (impone al modulo di trasmettere tramite bus i dati alla periferica).

Cosa sono gli errori soft, come ovviare a tali errori (fare un esempio): le memorie primarie sono soggette ad errori. Questi possono essere classificati in guasti hardware, che sono permanenti, e guasti software, chiamati anche "soft error", che sono casuali e non distruttivi. Essi infatti alterano i contenuti di una o più celle di memoria, senza però danneggiarla fisicamente. Gli errori "soft" possono essere rilevati ed

eventualmente corretti usando ad esempio il codice correttore di Hamming. Esempio nel caso di memorizzazione di un insieme a 4 bit. Esso funziona tramite bit di parità, si hanno 3 bit di controllo che indicano se il numero di 1 è pari... (esempio dei cerchi).

Come funziona il codice di correzione Hamming: quando i dati devono venire memorizzati si esegue un calcolo su di essi per produrre un codice che verrà memorizzato assieme ai dati. Quando si deve leggere una parola tale codice verrà impiegato per rilevare eventuali errori negli M bit di dati. Un nuovo codice è generato a partire dagli M bit dati e confrontato con i K bit precedentemente prelevati. Il confronto da 3 risultati: nessun errore rilevato, viene rilevato un errore ed è possibile correggerlo (i bit dati e i bit per la correzione vengono inviati ad un correttore che produce un insieme corretto di M bit), viene rilevato un errore ma non è possibile correggerlo, ciò viene segnalato. In un insieme di parole di 4 bit utilizziamo il diagramma di Eulero Venn per dividere il piano in 3 cerchi che determinano 7 regioni, i 4 bit dati verranno assegnati agli scomparti interni. I restanti scomparti vengono riempiti con i bit di parità. Ciascun bit di parità è scelto in modo che il numero totale di 1 nel proprio cerchio sia pari. Ora se avvenisse una modifica dei bit dati l'errore sarebbe facilmente rilevato e potrebbe essere corretto cambiando il determinato bit che lo causa.