Architettura degli Elaboratori
2 - Organizzazione dei Sistemi di Calcolo

Ugo Dal Lago

Dipartimento di Scienze dell'Informazione
Università degli Studi di Bologna

Anno Accademico 2007/2008

Sommario

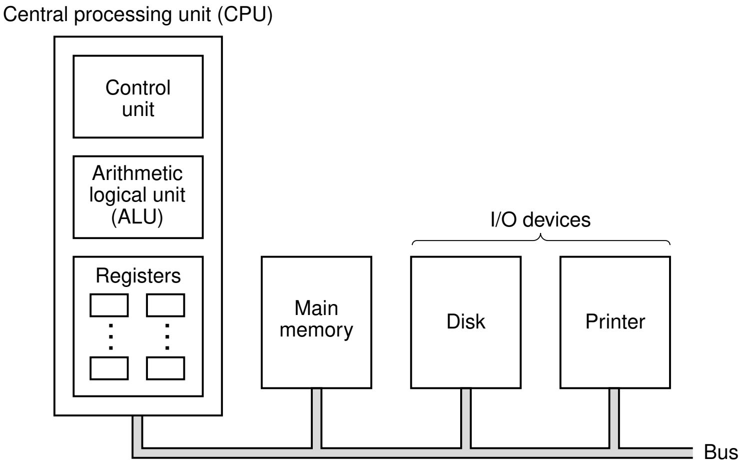
Processori

Memoria Principale

Memoria Secondaria

Input-Output

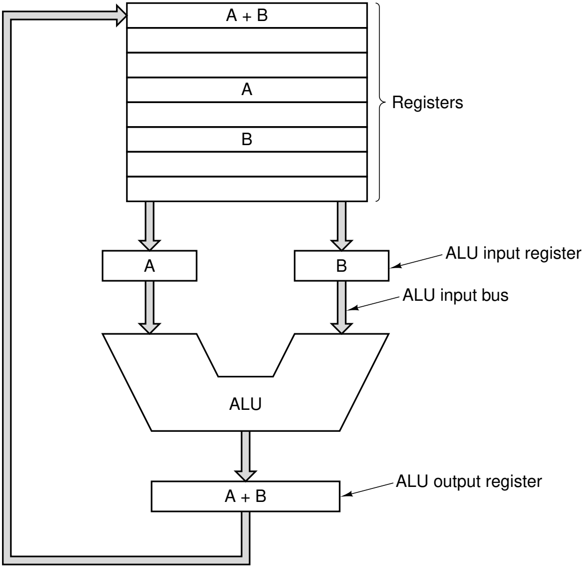
Organizzazione del Calcolatore



Il Processore

- Anche detto **CPU**, è il cervello del calcolatore.
- La sua funzione è quella di prelevare i programmi dalla memoria principale e di eseguire le relative istruzioni, una alla volta.
- La CPU è a sua volta composta da tre parti distinte:
  - L'**Unità di Controllo**, **UC**, che ha il compito di prelevare le istruzioni dalla memoria, determinandone il tipo.
  - L'**Unità Aritmetico Logica**, **ALU**, che esegue le operazioni necessarie per portare a termine le istruzioni.
  - I **registri**, memorie ad altà velocità che tengono traccia di risultati temporanei e informazioni di controllo.
- Due registri svolgono ruoli particolarmente importanti:
  - Il registro **PC** (o **program counter**), che contiene l'indirizzo della prossima istruzione da eseguire.
  - Il registro **IR**,(o **instruction register**), che contiene l'istruzione che si sta eseguendo.

Percorso Dati



Il Ciclo PDE

- La CPU esegue ogni istruzione compiendo una serie di piccoli passi, ripetuti iterativamente:
  1. Preleva l'istruzione cui punta PC e copiala in IR.
  2. Modifica PC per farlo puntare all'istruzione seguente.
  3. Determina il tipo dell'istruzione in IR.
  4. Se l'istruzione usa una parola di memoria, determina dove si trova.
  5. Se necessario, preleva la parola per portarla in un registro della CPU.
  6. Esegue l'istruzione.
- Questo ciclo viene detto **prelievo-decodifica-esecuzione** (**PDE**) oppure **fetch-decode-execute** (**FDE**).
- Ma **chi** esegue questo ciclo? La risposta la conosciamo già: l'unità di controllo.

Interpretazione e Unità di Controllo

- Il modo più semplice per realizzare l'unità di controllo consiste nel costruirla come un circuito hardware compatibile con un insieme fissato di istruzioni.
- Ciò diventa però complicato se l'insieme delle istruzioni è troppo grande. In tal caso, l'unità di controllo risulterebbe troppo costosa.
- Una possibile risposta a questo problema risiede nell'**interpretazione**: l'unità di controllo diventa **microprogrammata** e quindi può eseguire insiemi di istruzioni diversi e potenzialmente molto grandi.
- Cambiando il microprogramma che governa l'unità di controllo, possiamo modificare l'insieme delle istruzioni che possono essere eseguite dal calcolatore
- Ciò ha una serie di vantaggi:
  - Eventuali **errori** possono essere facilmente corretti.
  - **Nuove istruzioni** possono essere aggiunte con un costo minimo.

RISC contro CISC

- Alla fine degli anni Settanta, l'interpretazione permise di sperimentare istruzioni molto complesse (**CISC**). Non si pensava di realizzare macchine con istruzioni più semplici di quelle già presenti.
  - Tipico esempio era il calcolatore VAX, prodotto da Digital.
- All'inizio degli anni Ottanta, però, ricercatori dell'IBM e di alcune università californiane progettarono architetture con un numero ridotto di istruzioni (**RISC**).
  - Tipico esempio è il calcolatore MIPS.
- Le istruzioni delle macchine RISC sono in numero minore di quelle delle macchine CISC, ma sono eseguibili **più velocemente**.
- Tra RISC e CISC si scatenò presto una sorta di "guerra", che si combatte in parte ancora oggi...
- Intel Pentium è considerato CISC, mentre UltraSPARC è considerato RISC.

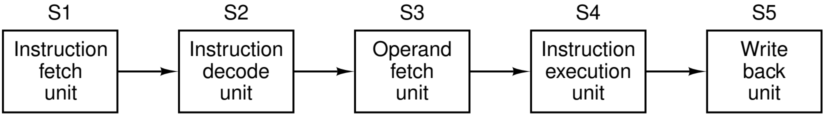
Principi di Progettazione dei Calcolatori Moderni

- Le istruzioni sono eseguite direttamente dall'unità di controllo, che **non è più microprogrammata**.
- Un parametro cruciale per le prestazioni è **la frequenza di emissione delle istruzioni** piuttosto che il tempo di esecuzione delle istruzioni.
  - In particolare in presenza di parallelismo.
- Il processo di **decodifica** dovrebbe essere più semplice possibile.
- Meno istruzioni fanno riferimento alla memoria, meglio è. Bastano un'istruzione per leggere dalla memoria (**LOAD**) e un'istruzione per scrivere in memoria (**STORE**).
- Visto che il tempo di accesso alla memoria è più lungo del tempo di accesso ai registri, è oppoprtuno avere a disposizione **un numero molto grande** di registri.

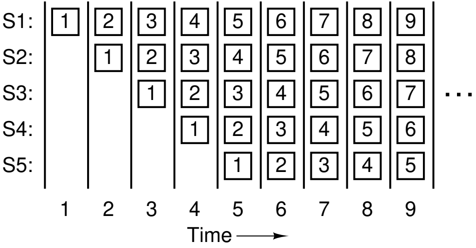
Parallelismo a Livello d'Istruzione

- Nel parallelismo a livello d'istruzione, cicli PDE relativi ad istruzioni distinte vengono sovrapposti tra loro.
- Lo scopo è quello di migliorare le prestazioni globali del calcolatore.
- Forma primitiva di parallelismo: **buffer di prefetch**.
  - Mentre si svolgono le ultime fasi del ciclo PDE relativo ad una certa istruzione, si carica l'istruzione successiva.
- Generalizzando l'idea a tutte le fasi del ciclo PDE, si ottiene la cosiddetta **pipeline**.
  - Tutte le fasi sono del ciclo PDE sono sovrapposte.
  - In questo modo la **latenza** e la **larghezza di banda del processore** risultano bilanciate.

Pipeline


S1
Instruction fetch unit
S2
Instruction decode unit
S3
Operand fetch unit
S4
Instruction execution unit
S5
Write back unit

(a)

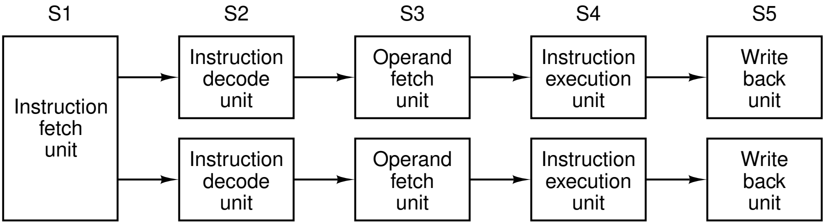

S1:
S2:
S3:
S4:
S5:
1 2 3 4 5 6 7 8 9
Time

(b)

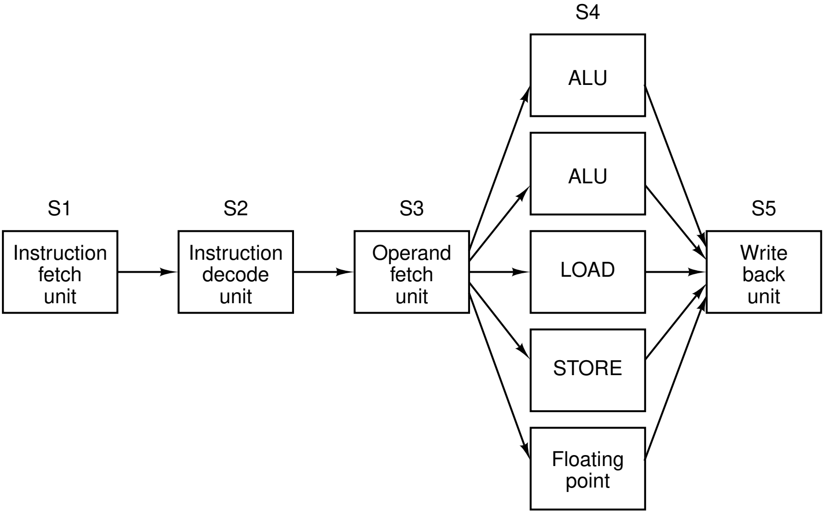
Parallelismo a Livello di Istruzione

- Un'ulteriore generalizzazione del concetto di pipeline è la **pipeline doppia**:
  - Un'unica unità di fetch carica due istruzioni alla volta.
  - Le fasi successive sono sovrapposte.
  - Si possono però verificare conflitti.
  - Utilizzata nel processore **Pentium**.
- Un'ennesima generalizzazione a più di due pipeline richiederebbe la duplicazione di troppi componenti hardware.
- L'idea è quindi quella di avere un'unica pipeline con più unità funzionali specializzate
- Si parla quindi di **architettura superscalare**
  - Utilizzata nel processore **Pentium II**.

Doppia Pipeline



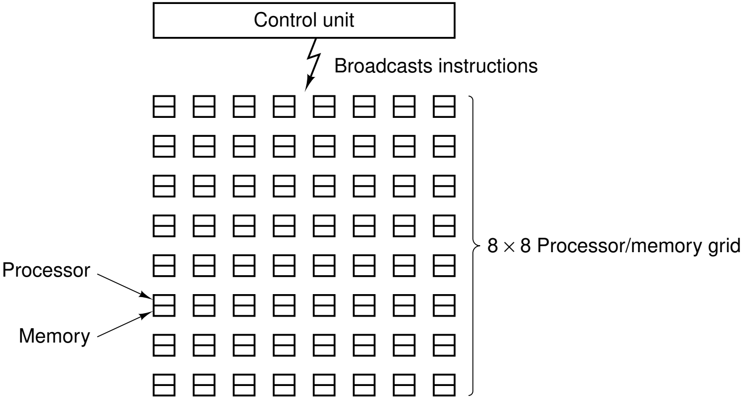
Processore Superscalare



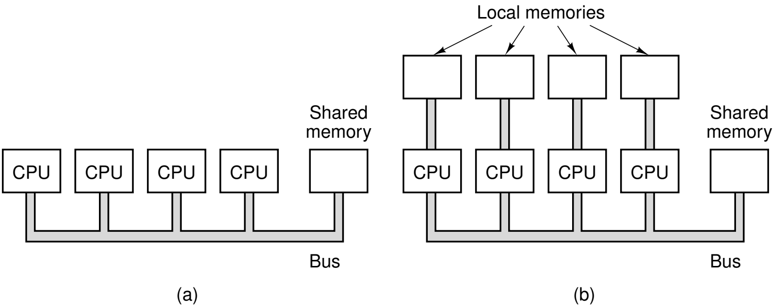
Parallelismo a Livello di Processore

- Una forma di parallelismo più forte consiste nell'avere molte CPU che lavorino in modo coordinato.
- Un primo esempio sono gli **array computer**.
  - Molti processori identici che eseguono la stessa sequenza di istruzioni su insiemi diversi di dati.
  - Si parla di **SIMD** (single istruction-stream, multiple data-stream).
- Un altro esempio importante è quello dei **multiprocessori**.
  - Molti processori anche diversi che condividono una memoria, senza eseguire necessariamente la stessa istruzione.
  - In questo caso si parla di **MIMD** (multiple istruction-stream, multiple data-stream).
- Infine, si possono avere i **multicomputer**, la forma più estrema di parallelismo.
  - Molti processori che non condividono una memoria e che comunicano scambiandosi dei messaggi.
  - Si può fare in modo che moltissime CPU cooperino.

Array di Processori



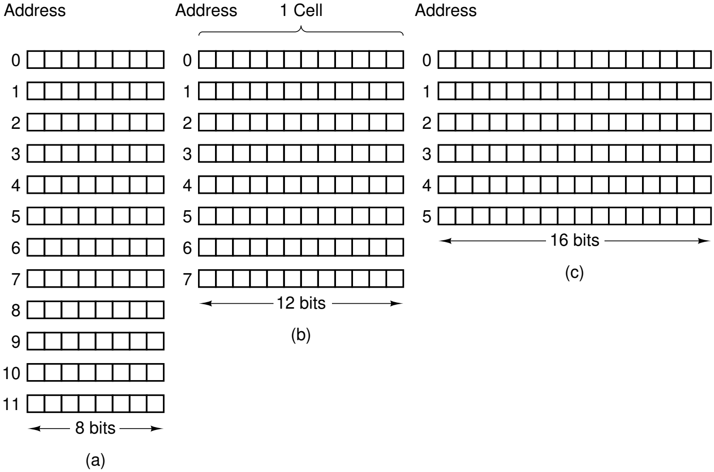
Multiprocessori



Bit, Byte, etc.

- La **memoria** è la parte del calcolatore in cui sono depositati programmi e dati.
  - **Volatile** o **principale**: il suo contenuto è perso quando si spegne il calcolatore.
  - **Non volatile** o **secondaria**: il suo contenuto non viene perso quando si spegne il calcolatore.
- L'unità base di memorizzazione è la **cifra binaria**, detta **bit**.
- La rappresentazione binaria dell'informazione è la più efficiente.
- La **cella** (o **locazione**) è ciascuna delle parti in cui è divisa la memoria.
- Ogni cella ha un **indirizzo**.
- Il **byte** consiste in 8 bit.
- La **parola** (o **word**) consiste in un certo numero di byte (o di bit).
- Calcolatori diversi lavorano con parole di dimensione diversa.

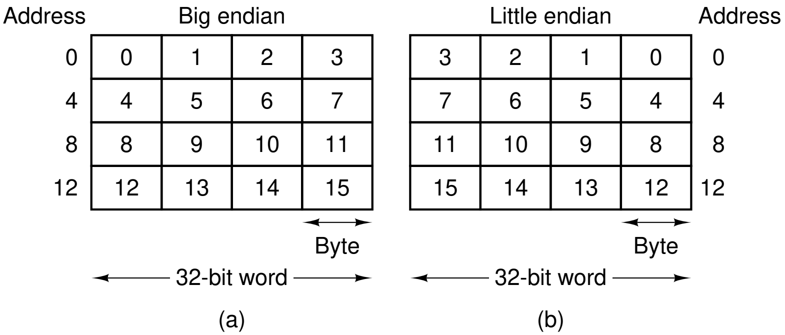
Memoria a 96 bit



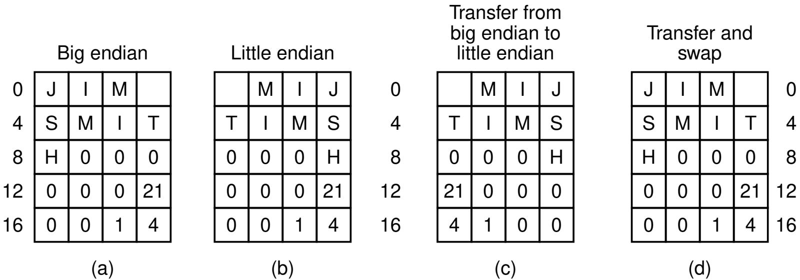
Ordinamento dei Byte

- All'interno di una parola i byte possono essere numerati da sinistra a destra o da destra a sinistra.
  - Nel primo caso si parla di sistema **big-endian**.
  - Nel secondo si parla di sistema **little-endian**.
- Se i calcolatori manipolassero la memoria solo scrivendo e leggendo parole, non ci sarebbero problemi.
- Il punto è che spesso esistono istruzioni che manipolano il singolo byte.
- In fase di trasmissione dati possono crearsi problemi se uno dei due sistemi utilizza il sistema big-endian mentre l'altro utilizza il sistema little-endian.

Big-Endian e Little-Endian



Big-Endian e Little-Endian



Codici Correttori

- Occasionalmente le memorie dei calcolatori sono soggette ad **errori**, sia durante le operazioni di lettura che durante le operazioni di scrittura.
- Per proteggersi da questi errori, alcune memorie utilizzano dei **codici di rilevazione e/o correzione di errori**.
- Se una parola consiste di m bit, si aggiungono r bit di controllo ottenendo una **parola di codice** a $n = m + r$ bit.
- La **distanza di Hamming** tra due parole di codice a n bit è il numero di bit rispetto ai quali le due parole differiscono.
- Un **codice** è un meccanismo atto a determinare gli r bit di controllo relativi ad ogni parola di m bit.
- La **distanza di Hamming** di un codice è la minima distanza di Hamming tra parole di codice ottenute tramite il codice stesso.
  - Per **rilevare** d errori singoli è necessario un codice con distanza di Hamming almeno pari a $d + 1$.
  - Per **correggere** d errori singoli è necessario un codice con distanza di Hamming almeno pari a $2d + 1$.

Esempi di Codici

- Uno dei codici più semplici è il cosiddetto **bit di parità**.
  - C'è un unico bit di controllo ($r = 1$), che è scelto in modo che il numero di bit 1 nella parola di codice sia **pari**.
  - Il codice ha distanza di Hamming pari a 2.
  - Il codice può quindi essere usato per rilevare errori singoli.
- Supponiamo di lavorare con parole di codice lunghe 10 e di supporre che esistano solo 4 parole legali tra le 2^{10} possibili:

  0000000000
  0000011111
  1111100000
  1111111111

  Tutte le altre parole sono illegali.
  - Questo codice ha distanza di Hamming pari a 5.
  - Può quindi correggere errori doppi, ed è facile rendersi conto perché.

Errori Singoli: Un Limite Strutturale

- Supponiamo che un codice con m bit di dati e r bit di controllo sia in grado di correggere tutti gli errori singoli.
- Ciò significa, in particolare, che per ognuna delle 2^m parole di codice legali esistono n parole di codice errate distinte a distanza 1 da essa.
- Ciascuna delle 2^m parole di codice legali "richiede" $n+1$ parole di codice ad essa dedicate:
  - 1 per la parola di codice corretta.
  - n per le parole di codice errate.
- Dato che il numero totale di combinazioni di bit è 2^n, deve valere

  $$(n+1)2^m \leq 2^n$$

  che può essere riscritta come

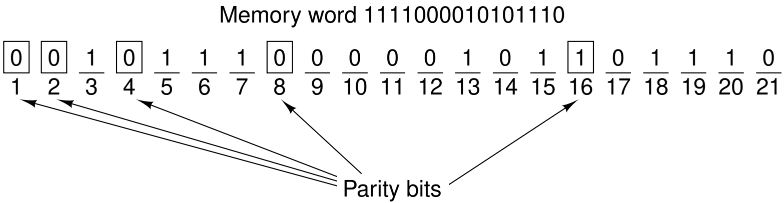
  $$m+r+1 \leq 2^r$$

- È possibile ottenere questo limite teorico utilizzando un metodo ideato da Richard Hamming.

Codice di Hamming di Lunghezza Arbitraria

- Ad m bit di dati vengono aggiunti r bit di controllo.
- Per identificare la posizione di un bit si usano gli interi tra 1 e $m + r$.
- I bit di controllo sono quelli con posizione pari ad una potenza di due.
- Gli r bit di controllo sono bit di parità, ma calcolati non a partire da tutti i bit di dati bensì a partire da alcuni tra essi.
- Del generico bit in posizione b si tiene conto nel calcolo dei bit di parità $b_1, b_2, \ldots, b_j$ se e solo se $b_1 + b_2 + \ldots + b_j = b$.
- Per correggere l'errore (che si suppone singolo) si procede controllando il valore di tutti i bit di parità, cercando di restringere progressivamente l'insieme delle posizioni scorrette.

Esempio



Memory word 1111000010101110
0 0 1 0 1 1 1 0 0 0 0 0 1 0 1 1 0 1 1 1 0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
Parity bits

CPU e Memorie

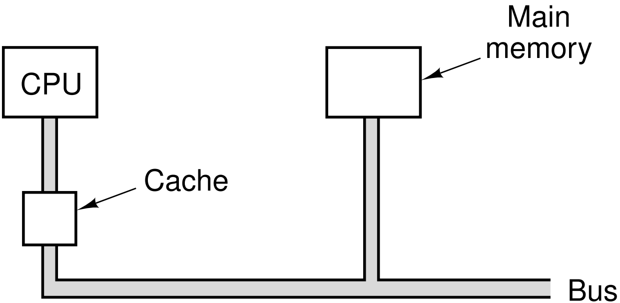
- Le CPU sono sempre state, storicamente, **più veloci** delle memorie.
  - Questo divario prestazionale è se possibile aumentato negli ultimi anni.
- Di conseguenza, ogni richiesta di lettura dalla memoria richiederà un certo tempo, durante il quale è possibile che la CPU debba attendere, degradando le prestazioni del sistema.
- Esistono un certo numero di soluzioni al problema:
  - Prima di tutto, la CPU può iniziare a processare qualunque istruzione riguardante la memoria **appena la incontra**.
  - Si può chiedere a chi genera il codice (il programmatore oppure un compilatore) di produrre programmi che non soffrano di questo problema, per esempio facendo in modo che ogni richiesta di lettura sia **seguita** da un certo numero di istruzioni "innocue".

  Nessuna di queste due strategie risolve però il problema.
- Vi sono anche considerazioni di carattere **economico** oltre che **tecnologico**.

Memoria Cache

- Una tecnica interessante consiste nell'utilizzare una memoria non troppo grande ma molto veloce, chiamata **cache**.
- Le parole di memoria **usate più di frequente** sono mantenute all'interno della cache.
- Quando la CPU necessita di una parola, la cerca nella cache e, solo nel caso in cui essa non sia presente, la richiede alla memoria centrale. In tal caso il contenuto della memoria centrale viene trasferito nella cache.
- Le prestazioni globali del sistema dipendono da **quale frazione** delle richieste di accesso alla memoria può essere soddisfatta direttamente dalla cache.
- La maggior parte dei programmi esibiscono il cosiddetto **principio di località**:
  - Riferimenti alla memoria temporalmente vicini sono spesso anche spazialmente vicini.
  - Lo stesso indirizzo viene referenziato in più istanti vicini tra loro.

Posizione Logica della Cache

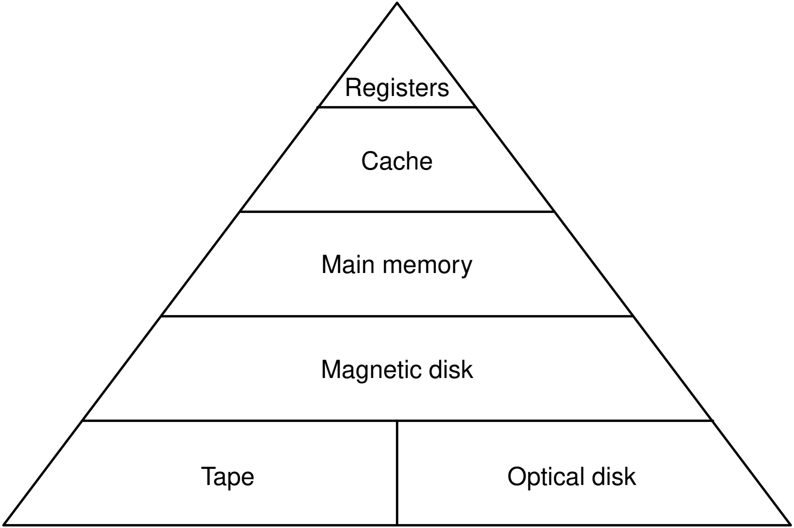


Ancora sulla Cache

- Supponiamo che
  - c sia il tempo di accesso alla memoria cache.
  - m sia il tempo di accesso alla memoria centrale.
  - h sia l'**hit-ratio**, ossia la frazione di riferimenti che può essere soddisfatta dalla cache.

  allora il tempo medio di accesso t è $c + (1 - h)m$.
- Quando h si avvicina a 1, il tempo medio t si avvicina a c, mentre quando h si avvicina a 0, t si avvicina a $c + m$.
- Le memorie centrali e le cache sono suddivise in blocchi di lunghezza fissata. Nel caso delle cache, si parla di **linee di cache**.
  - Quando si verifica un fallimento, è l'intera linea ad essere portata in cache dalla memoria centrale.
  - Il principio di località può essere ulteriormente sfruttato.
- Possiamo avere:
  - Una **cache unificata**, quando in una sola cache vengono memorizzati dati e istruzioni.
  - Una **cache specializzata**, quando le istruzioni vanno a finire in una cache e i dati in un'altra.

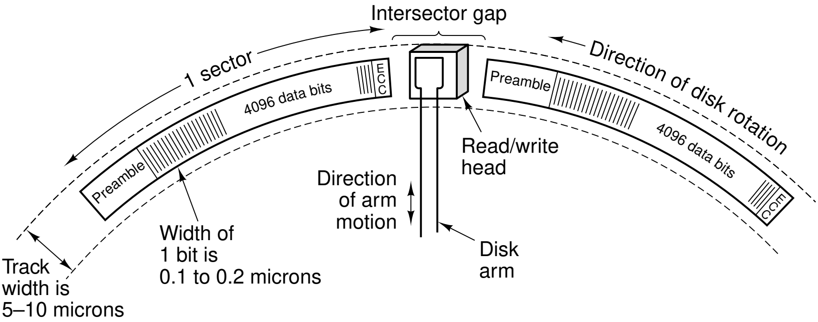
Gerarchie di Memoria



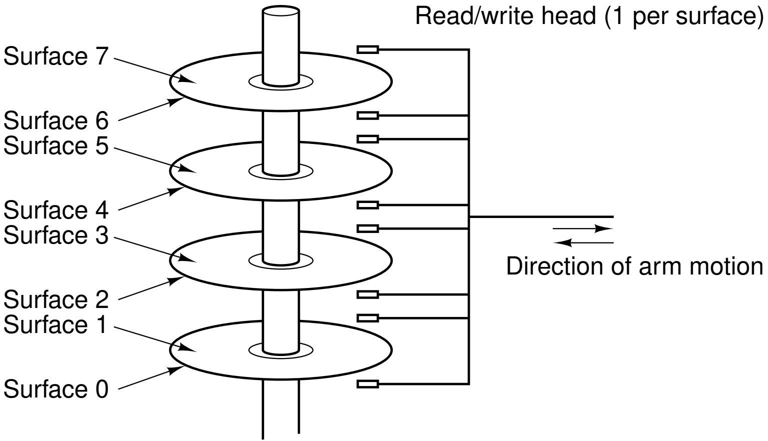
Dischi Magnetici

- Un disco magnetico consiste in uno o più piatti di alluminio rivestiti di **materiale magnetico**.
  - Una corrente elettrica che passa attraverso la testina magnetizza la superficie che si trova sotto.
  - Quando la testina passa sopra un'area magnetizzata, viene indotta nella testina una corrente positiva o negativa.
- La **testina** del disco sfiora la superficie o, nel caso dei floppy disk, tocca realmente la superficie.
- La **traccia** è una sequenza circolare di bit scritti mentre il disco compie una rotazione completa.
- Le tracce sono suddivise in un certo numero di **settori**, contenenti un numero fissato di byte (512 byte, di solito).
- Quando un disco consiste in più piatti, l'insieme delle tracce ad una stessa distanza dal centro è chiamato **cilindro**.
- Con la **tecnologia attuale** si riescono ad ottenere:
  - Tra 5000 e 10000 tracce per centimetro.
  - Tra 50000 e 100000 bit per centimetro lungo una stessa traccia.

Porzione di Traccia



Disco con Quattro Piatti



Dischi Magnetici

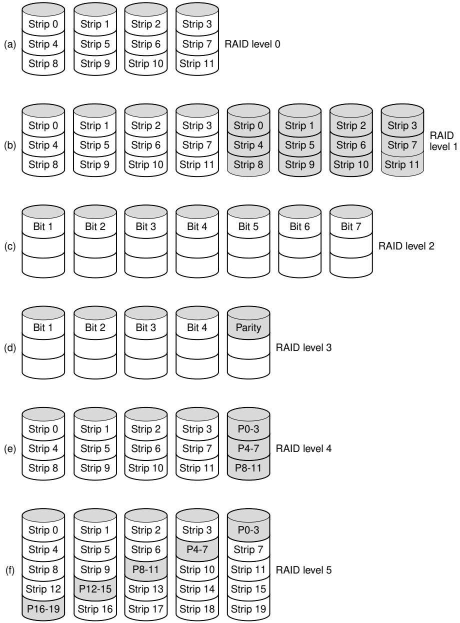
- Le prestazioni dei dischi dipendono da:
  - Tempo di **seek** (spostamento radiale sulla posizione corretta), circa $5 - 10$ ms.
  - Tempo di **latenza** (in attesa che il settore ruoti sotto la testina), circa $3 - 6$ ms.
  - Tempo di **trasferimento**, circa $13 - 16$ μs per 512 byte.

  I tempi di seek e latenza dominano la velocità di trasferimento.
- Il **burst rate** è meno significativo del **sustained rate**.
- Le tracce più esterne hanno una distanza lineare maggiore, e ciò mette in evidenza un problema, dato che i dischi ruotano generalmente a velocità angolare costante.
  - Oggi si usa una strategia diversa: i cilindri sono suddivisi in **zone**.
  - Il numero di settori per traccia aumenta man mano che ci si sposta dalla traccia più interna verso l'esterno.
- Ad ogni disco, chiamato **controllore del disco**.

Dischi Magnetici

- **Floppy Disk**
  - Sia i supporti che le testine si consumano velocemente.
- **Dischi IDE**
  - Per mantenere la retrocompatibilità, si sacrifica la semplicità.
  - Si passò poi a **EIDE**, **ATA-3**, **ATAPI-4**, etc.
- **Dischi SCSI**
  - Hanno velocità di trasferimento più elevata dei dischi IDE.
  - SCSI non è solo un'interfaccia per hard disk, ma è anche un bus al quale possono essere collegati un controllore e/o altri dispositivi.
  - Sono apparsi poi **Fast SCSI**, **Ultra SCSI**, etc.
- **RAID**
  - L'idea è quella di **distribuire** e/o **replicare** l'informazione su più dischi.
  - Il gruppo di più dischi è però visto come un unico disco virtuale.
  - Esistono livelli RAID dallo 0 al 5: ognuno di essi gestisce la distribuzione e la ridondanza in modo diverso.

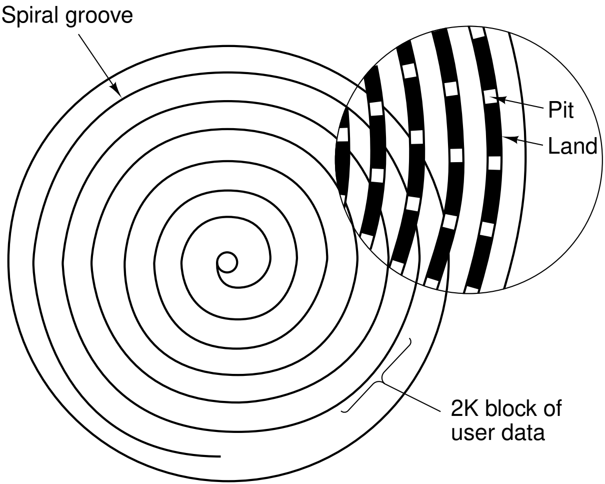
RAID dal livello 0 al livello 5



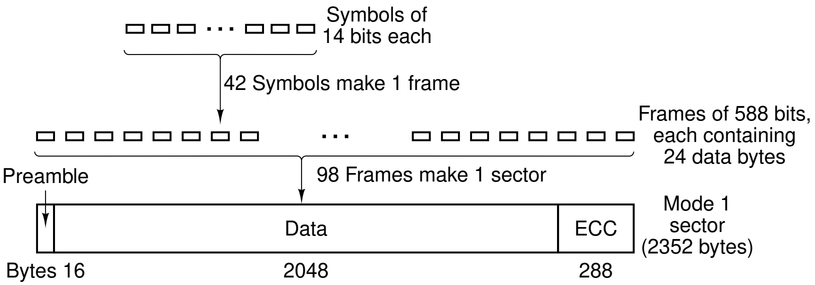
Dischi Ottici

- Nascono per registrare programmi televisivi, ma oggi si prestano ad utilizzi più interessanti.
- I **CD-ROM** Hanno dimensione standard: diametro di 120 mm, uno spessore di $1,2$ mm e un buco di 15 mm.
- I CD-ROM vengono prodotti con policarbonato liquido, alluminio riflettente e vernice protettiva.
- I dati sono rappresentati dalla presenza di piccoli buchi, detti **pit**, mentre le aree non "bucate" si chiamano **land**.
- La lettura dei dati avviene tramite l'impiego di un laser con lunghezza d'onda di $0,78$ micron.
- La scrittura dei dati avviene una sola volta, utilizzando uno stampo, detto **master**.
- I lettori CD-ROM a singola velocità raggiungono una velocità di trasferimento dati di circa 175.000 byte al secondo.
  - Neanche un lettore a 32x raggiunge la velocità di un disco SCSI!

Struttura di Registrazione di CD-ROM



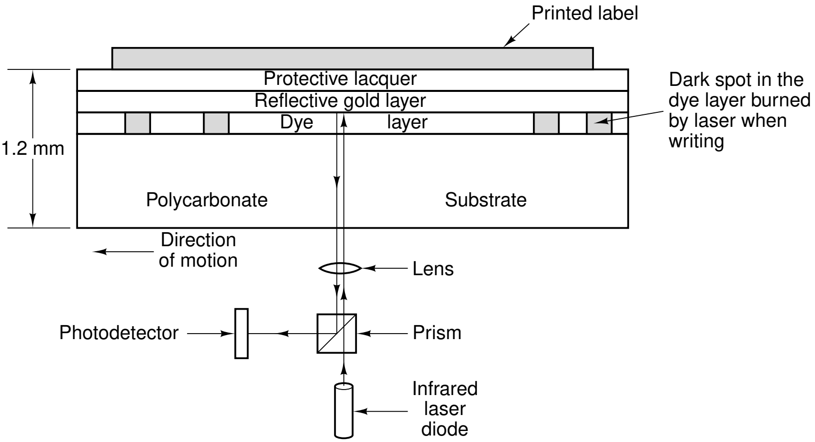
Struttura logica dei dati di un CD-ROM



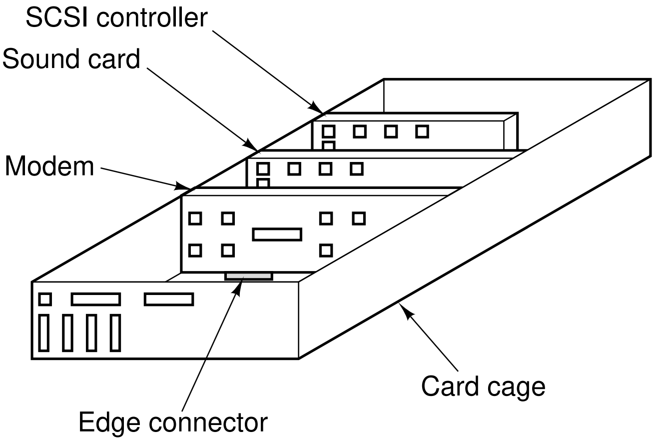
Dischi Ottici

- I **CD-R**, o **CD registrabili**, supportano la scrittura (una sola volta), ma anche la lettura (più volte).
- A partire dalla metà degli anni '90, i masterizzatori di CD divennero una periferica di uso comune.
- La tecnologia è diversa:
  - I pit e i land devono essere **simulati**. A tale scopo si aggiunge uno strato di pigmento tra il policarbonato e il livello riflettente.
  - Lo strato di pigmento nello stato iniziale è trasparente.
  - Per scrivere il CD-R si utilizza un laser, con potenza molto maggiore di quello usato in lettura. In questo modo si crea una regione scura nel pigmento.
- Un'altra peculiarità dei CD-R sta nel fatto di essere scrivibili in modo incrementale.
- Un ulteriore passo avanti è rappresentato dai **CD-RW**, che possono essere scritti più volte.
  - Lo strato di pigmento è sostituito da una lega di argento, indio, antimonio e tellurio.
  - Nei lettori CD-RW, il laser funziona con tre potenze distinte.

Spaccato di un disco CD-R



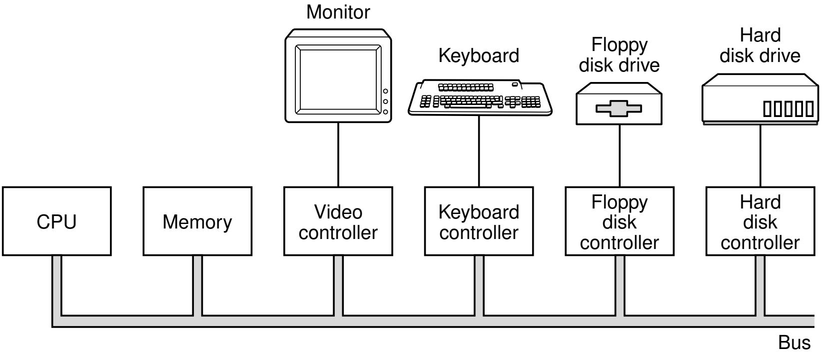
Struttura Fisica di un Personal Computer



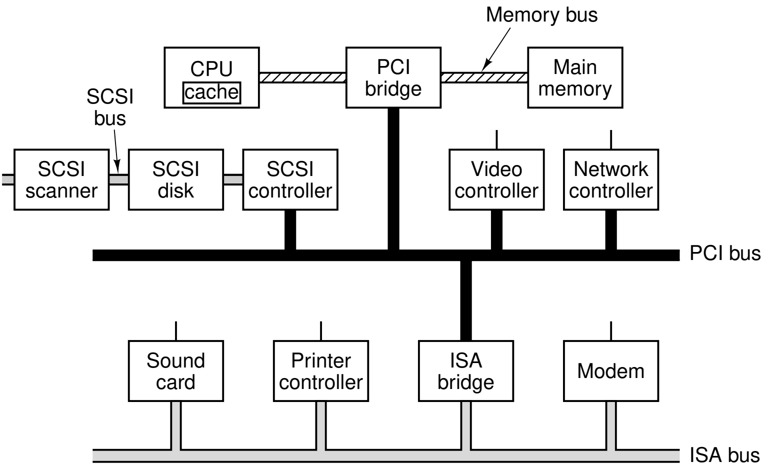
Controller, Bus, etc.

- Ad ogni dispositivo di I/O è associato un **controllore**.
  - Compito del controllore è governare il dispositivo e gestire il suo accesso al bus.
  - In alcuni casi il controllore può scrivere direttamente in memoria senza l'intervento della CPU. Si parla in tal caso di **DMA** (o **direct memory access**).
  - Nel caso in cui si utilizzi il DMA, la CPU può eseguire delle normali istruzioni mentre il controllore accede alla memoria. Quando il trasferimento è completato, il controllore produce un **interrupt**.
- Che succede se la CPU e uno o più controller accedono contemporaneamente al bus?
  - Esiste uno speciale chip, chiamato **arbitro del bus**, che stabilisce i turni, dando di solito la precedenza ai dispositivi di I/O.

Struttura Logica di un Semplice Personal Computer



Tipico PC Odierno



Alcune Periferiche di Uso Comune

- **Terminali**
  - Sono composti da tastiera e monitor.
- **Mouse**
  - Sono dispositivi di puntamento.
  - Possono essere meccanici oppure ottici.
- **Stampanti**
  - Possono essere ad aghi, a getto d'inchiostro, laser, etc.
  - Possono essere monocromatiche o a colori.
- **Apparecchiature per le Telecomunicazioni**, per esempio:
  - **Modulatori-Demodulatori**, altrimenti detti **Modem**.
  - **Schede di Rete**.
- **Macchine Fotografiche Digitali**.