Base:
Componenti: cpu, memoria, input output, connessioni

Von Neumann:
dati e istruzioni in memoria (accessibile per indirizzo) che vengono eseguiti in ordine sequenziale

Programma cablato:
componenti logici hardware che non possono venire modificati, se sono circuiti generici accetta segnali di controllo che dicono che programma eseguire

Programma: Sequenza di operazioni logiche e aritmetiche

Programmazione software: l'hardware è generico, preleva il codice di una istruzione e genera i segnali di controllo per la CPU che interpreta le operazioni aritmeticologiche

CPU e memoria: si scambiano dati usando due registri della CPU, il MAR (memory address register) che contiene l'indirizzo in memoria da cui leggere o scrivere e il MBR (memory buffer register) che contiene i dati da scrivere o che riceve i dati letti. [c'è un AR e un BR anche per l'I/O]

Ciclo della CPU:
La CPU ha un registro PC (program counter) che contiene l'indirizzo della prossima istruzione e che verrà incrementato dopo aver preso l'istruzione. L'istruzione che viene presa (fetch) sarà caricata nell'IR (istruction register) e interpretata dalla CPU.
Ci sono 4 tipi di istruzioni:
- Processore-memoria: trasferimento CPU-Memoria
- Processore-I/O: trasferimento CPU-I/O
- Elaborazione dati: Operazioni aritmetico-logiche
- Controllo: alterazione della sequenza di istruzioni
Per trasferire dati viene utilizzato un AC (accumulatore) dove vengono salvati i dati trasferiti o da trasferire.

Esempio:
Registri PC, IR, AC
Parole della M di 16bit
Dati e istruzioni a 16bit
Codici operativi (OpCodes) a 4bit [0001: carica AC in M, 0010: scrive in M l'AC, etc]
Possono esserci 16 codici (4 cifre, codice binario => 4^2)
Restano altri 12bit indirizzabili nell'istruzione, quindi 2^12 byte = 4096 = 4Kb

Le interruzioni: sono meccanismi attraverso i quali gli altri moduli possono interrompere la sequenza di esecuzione (per esempio a causa di un errore, in seguito a un timer, per segnalare la fine di un'operazione di I/O o a causa di un guasto hardware)
Vengono utilizzate per migliorare l'efficienza dell'elaborazione (così che la CPU non debba sprecare tempo aspettando un segnale da un dispositivo esterno o da parte dell'utente)
Dopo aver eseguito un'istruzione la CPU controlla se le interruzioni sono abilitate e controlla se c'è una interruzione pendente; se c'è
sospende l'esecuzione del programma corrente,
salva il contesto (l'indirizzo della prossima istruzione),
imposta il PC (program counter) all'inizio del programma di gestione dell'interruzione,
una volta eseguito il programma rimette il contesto nel PC e continua il programma.
Le interruzioni possono essere sequenziali (una dopo l'altra) o annidate (una dentro l'altra).
Le interruzioni multiple possono essere gestite in due modi:
- stabilendo una priorità così che le interruzioni con la priorità più alta possano interrompere le interruzioni con priorità più bassa
- disabilitando le interruzioni quando viene eseguita un'interruzione

Dopo aver eseguito un'istruzione la CPU controlla se le interruzioni sono abilitate e controlla se c'è una interruzione pendente; se c'è
sospende l'esecuzione del programma corrente,
salva il contesto (l'indirizzo della prossima istruzione),
imposta il PC (program counter) all'inizio del programma di gestione dell'interruzione,
una volta eseguito il programma rimette il contesto nel PC e continua il programma.
Le interruzioni possono essere sequenziali (una dopo l'altra) o annidate (una dentro l'altra).
Le interruzioni multiple possono essere gestite in due modi:

- stabilendo una priorità così che le interruzioni con la priorità più alta possano interrompere le interruzioni con priorità più bassa
- disabilitando le interruzioni quando viene eseguita un'interruzione

Connessioni:
Memoria-processore
I/O-processore
I/O-memoria (attraverso DMA direct memory access)
Bus: collega due o più dispositivi, è condiviso tra tutti i dispositivi, può essere utilizzato solo da un dispositivo, è diviso in più linea (ogni linea, un bit)
Ci sono 3 gruppi di linee:
Bus dati: trasporta i dati, ampiezza importante per l'efficienza
Bus indirizzi: indica la sorgente o la destinazione dei dati, ampiezza indica la quantita di memoria indirizzabile
Bus di controllo: controllare il tipo di trasmissione che può essere:
- scrittura/lettura memoria
- richiesta bus (un modulo deve usare il bus)
- bus grant (concesso il controllo del bus a un modulo)
- interrupt request (interruzione pendente)
- clock (sincronizza le operazioni)
- reset (inizializza tutti i moduli)
Per usare il bus un modulo deve fare richiesta del bus, ricevere il bus grant e trasferire i dati o la richiesta al modulo da cui leggere
A volte vengono usati più bus (utilizzando delle Expansion bus interface che comunicano con l'interfaccia principale del bus) per evitare problemi di congestione e di ritardo
I bus possono usare un timing sincrono: gli eventi sono determinati da un clock, ovvero una linea su cui viene spedita una sequenza alternata di 0 e 1 di uguale durata (una sequenza 1-0 è un ciclo di clock); gli eventi partono da un ciclo di clock
Se il timing è asincrono invece, gli eventi dipendono dai precedenti eventi

---
Memoria:
Locazione: Può essere nel processore, interna o esterna

Capacità: La memoria è espressa in bytes o in parole (parola=lunghezza in bits necessaria per rappresentare un numero). Per indirizzare la memoria in bits: 2^lunghezzaindirizzo = memoria

Unità di trasferimento: parole o blocchi

Metodo di accesso:
- sequenziale: memoria organizzata in records, meccanismo di lettura/scrittura condiviso (bisogna bisogna muovere il meccanismo di lettura scrittura dalla posizione corrente alla nuova posizione), tempo di accesso variabile dipendente dalla posizione
- diretto: meccanismo di lettura/scrittura condiviso ma i blocchi o i records hanno un indirizzo basato sulla posizione fisica, tempo di accesso variabile
- accesso casuale: ogni locazione ha un sistema di accesso fisico, tempo di accesso costante
- associativo: ad accesso casuale e consente di prendere una parola in base al suo contenuto

Prestazioni:
- tempo di accesso: nelle memorie ad accesso casuale, tempo necessario per leggere e scrivere i dati; nelle altre, tempo necessario per posizionare il meccanismo di lettura/scrittura alla locazione desiderata
- tempo di ciclo: nelle memorie ad accesso casuale, il tempo prima che un secondo accesso possa essere iniziato
- velocità di trasferimento: velocità a cui i dati vengono trasferiti dentro e fuori dall'unità di memoria

Modello fisico: semiconduttore, magnetico, ottico, magnetico ottico

Caratteristiche fisiche: volatile/non volatile, riscrivibile/non riscrivibile

Organizzazione:
La memoria viene organizzata su più livelli: nel primo (o primi), la cache, è una memoria molto veloce e costosa dalle dimensioni ridotte
nel secondo (o gli ultimi), la memoria centrale, è una memoria molto ampia e lenta dal costo contenuto.
La CPU usa il livello più alto e ogni livello inferiore dovrà avere i dati contenuti ai livelli superiori.
Questo è un compromesso per avere sia la velocità della cache che l'ampiezza della memoria centrale a un costo accessibile e si basa sulle proprietà dei programmi (spesso gli indirizzi sono consecutivi e gli accessi a indirizzi contigui o a quelli acceduti più di recente sono più probabili) e sulla congettura 90/10 (nel 90% del tempo di esecuzione un programma vengono usate il 10% del numero di istruzioni dello stesso)

La memoria può venire suddivisa in blocchi (la sua dimensione è la dimensione minima di dati che occorre prelevare dal livello inferiore); di conseguenza l'indirizzo di un dato è la posizione del blocco sommato alla posizione del dato all'interno del blocco.

Hit e miss: un dato richiesto dalla CPU può essere in cache (hit) oppure può non esserci (miss)
Se non c'è comincia lo swapping (scambio dati con il livello inferiore = cerca in M il blocco contenente il dato, alloca una linea in C, carica il blocco da M in C)
TmedioAccesso = TaccessoInCache * ProbabilitàHIT + TaccessoMemoria * (1-ProbabilitàHIT)

Tecniche per organizzare la cache
Associazione diretta: ogni blocco del livello inferiore può essere collocato in una specifica posizione del livello superiore
conversione semplice tra indirizzi di memoria e della cache
si usa un tag per contraddistinguere il blocco nella cache
ogni indirizzo si divide in tag-line-word

Associazione completa: ogni blocco può essere posto in qualunque posizione del livello superiore
viene associata una tabella di posizioni contenenti il numero di blocco (tag)
è pesante determinare la corrispondenza la corrispondenza tra cache e memoria e verificare gli hit e i miss

Associazione a N gruppi: ogni blocco può essere allocato in un gruppo di blocchi nel livello superiore
ogni indirizzo si divide in tag-set-word
vengono associate N tabelle di elemenenti contenenti le tag dei blocchi

Politiche di rimpiazzo dei blocchi in cache:
- casuale: occupazione omogenea dello spazio
- FIFO (first in first out): il primo inserito
- LFU (least frequently used): il blocco con meno accessi
- LRU (least recently used): il blocco che ha accessi più vecchi

Problema della scrittura:
Incoerenza tra cache e memoria:
write through: scrive in contemporanea sulla cache e sulla memoria (+ traffico + coerenza)
write back: scrivo in differita al rimpiazzo del blocco in cache (necessario controllo sulla scrittura sul blocco, ottimizzazione traffico, periodi di incoerenza)

se ci sono più cpu e più cache si può:
- monitorare il bus con dei contrllori che intercettano modifiche su locazioni condivise e una politica di write through
- via hardware, quando una modifica viene effettuata in memoria, modifica tutte le cache
- una parte della memoria è condivisa e non cachabile

Problema dei miss:
- miss di primo accesso: necessario
- miss per capacità insufficiente: la cache non può contenere tutti i blocchi
- miss per conflitto: più blocchi possono essere mappati sullo stesso gruppo
Se aumento la dimensione del blocco aumentano i miss per conflitto, se aumento l'associatività incremento il tempo di localizzazione nel gruppo
Posso risolvere con cache multilivello, separando la cache dati dalla cache istruzioni, ottimizzando gli accessi tramite compilatori
---
Semiconduttori:
RAM: accesso casuale, read/write, volatile, memorizzazione temporanea, statica o dinamica

DRAM:
bit memorizzati come cariche in condensatori
con il tempo decadono le cariche -> necessario refresh delle cariche
costruzione più semplice e meno costosa
più lente
analogico
memoria principale
funzionano facendo fluire la corrente (ad alta e bassa tensione per scrivere, confrontando il valore di carica con un valore per leggere)

SRAM:
bit memorizzati tramite porte logiche
nessuna perdita di carica -> nessun refresh

costruzione complessa (più componenti per un bit) e costosa
più veloci
digitale
cache
funzionano controllando due dei quattro transistor

ROM: non volatili, usate per microprogrammi, subroutine di libreria, BIOS, funzioni tabulate
PROM:programmabili una sola volta
EPROM: si cancellano con raggi ultravioletti
EEPROM: electrically erasable, lente
Memorie flash: cancellazione elettrica di blocchi di memoria

I guasti:
Guasti hardware (permamenti)
Errori software (random non distruttivi, non permanenti)

Codici correttori di Hamming: viene generato un codice che viene usato come checksum per validare i dati. Nel caso non sia corretto corregge i bit

---

Dischi magnetici:
Disco di vetro (+rigido, +resistente, +uniformità della superficie, -difetti, -distanza testina disco, al posto dell'alluminio) rivestito di materiale magnetico
Read/Write tramite la testa che è una bobina conduttiva che produce un campo magnetico in grado di modificare i campi del disco; i campi sul disco, muovendosi, inducono corrente sulla bobina (adesso però c'è un sensore magneto-resistivo separato)
I dati vengono salvati in anelli in tracce concentriche divise in settori (un blocco ha come dimensione minima il settore)
Si registrano i dati a più zone in modo che con la rotazione la velocità angolare costante non provochi una minore densità di memorizzazione nelle tracce più esterne

Ci possono essere piatti multipli ma serve una testina per faccia disco e i dati sono distribuiti sul cilindro

Prestazioni:
T = Ts + Tl + Tt
Ts = tempo di seek, di posizionamento della testina
Tl = (1000/(RPM/60))/2
Tt = byteDaTrasferire/(RPM/60)/bytePerTraccia

RAID: Redundant Array of Indipendent/Inexpensive Disks
Non gerarchici, insieme di dischi visti dal sistema come un singolo, dati distribuiti
0: Nessuna ridondanza, dati distribuiti a strisce. + velocità, - consumo dello stesso disco
1: Contenuto replicato su più dischi, 2 copie di dati su dischi separati. + sicurezza, + costoso
2: Dischi sincronizzati, codice di correzione degli errori (Hamming). ++sicurezza, ++ costoso
3: Un disco ridondante, bit di parità per ogni insieme corrispondente di bit, i dati su disco difettoso possono essere ricostruiti con gli altri dati e le informazioni di parità
4: ogni disco opera indipendentemente, unità di informazione ampia, parità bit a bit calcolata tra unità di informazione per ogni disco, informazione di parita sul parity disk
5: come 4, parità distribuita su tutti i disci, allocazione roundrobin per la parità. - collo di bottiglia del parity disk
6: Calcolo di parità tramite due metodi distinti, servono N+2 dischi, alta affidabilità dei dati (devono rompersi 3 dischi), + lento

Ottica
CD-ROM: all'inizio per dati audio, 650mb -> 70minuti
dischi di policarbonato (etichetta, strato protettivo, strato riflettente, plastica) con materiale riflettente
dai memorizzati come pozzi
lettura laser
densità di memorizzazione costante
velocità lineare costante, 1.2ms^-1
traccia a spirale lunga 5.27km
formato:
modo 0= campo dati vuoto
modo 1= 2048 byte dati + correzione errori
modo 2= 2336 byte dati

l'accesso casuale è difficile a causa della velocità lineare costante e della necessità di spostare la testina in posizione approssimata, configurare la giusta velocità di rotazione, leggere l'indirizzo, spostarsi sul settore richiesto

facile da produrre, rimovibile, robusto ma lento, sola lettura

ci sono i cd rw che funzionano a cambiamento di fase

dvd: digital video disk, digital versatile disk
multistrato (carbonato, semriflettente, carbonato, riflettente x2), alta capacità

nastro magnetico: lento, seriale, economico, usato per i backup

Input/Output:
Moduli più lenti della cpu e della ram che si interfacciano con l'uomo, con la macchina (monitoraggio, controllo), con il mondo (rete)

Funzionamento: la CPU interroga il modulo sullo stato del dispositivo, il modulo ritorna lo stato del dispositivo, se pronto la CPU richiede i dati e comincia il trasferimento
Il modulo I/O può controllare le funzioni del dispositivo o lasciare il controllo alla CPU, supportare dispositivi singoli o multipli, nascondere o rivelare proprietà alla CPU
CPU e I/O possono comunicare in 3 modi:
Programmed I/O : polling continuo
Interrupt Driven I/O : richiesta -> trasferimento dati
Direct memory Acccess DMA: richiesta alla DMA -> interruzione quando finisce

La CPU invia un indirizzo che identifica il modulo e il dispositivo (ogni dispositivo ha un identificatore unico) oppure un comando di controllo (comando), di test (controllo di uno stato), di read/write (trasferimento dati)

I/O Mapping:
Memory-mapped: indirizzamento uguale a quello della memoria, sembra come una lettura/scrittura di memoria, no comandi speciali, si può nascondere meglio alcune aree dell'I/O, con la tecnica della memoria segmentata più aree di I/O mappano lo stesso spazio di indirizzamento.
MA il dato è nella memoria del dispositivo quindi bisogna disabilitare la cache, non compatibile con i bus multipli
Separato: spazi di indirizzamento separati, comandi speciali per l'I/O

DMA:
Modulo hardware addizionale che riduce l'intervento della CPU sostituendola nelle attività di I/O
La CPU fa una richiesta alla DMA che ritorna un'interruzione una volta finito.
Può accedere al canale una parola alla volta (sottraendo poco alla volta il canale alla CPU =cycle stealing) o per blocchi (prendendo possesso del canale =burst mode)

Aritmetica dei calcolatori:
ALU: Esegue le operazioni aritmetico-logiche
Gestisce interi e reali

Complemento 2:
il primo bit è di segno, 0 positivo 1 negativo
per ottenere il numero negativo di uno positivo basta fare il complemento del numero e aggiungere uno
0010 => 2, 1101+1=1110 => -2
per convertire da una lunghezza all'altra basta aggiungere zeri o uno a seconda del segno

Hardware:
somma: somma binaria
sottrazione: circuiti per la somma e per il complemento
moltiplicazione: moltiplicazione normale con numeri positivi, poi tengo conto dei segni

binario della parte intera (2=>10), moltiplico la parte decimale per 2 e segno 1 se è sopra 1 e 0 se è sotto (0.625*2 = 1.25 =>1,
0.25*2=0.5 => 0, 0.5*2=1 =>1), normalizzo (10.101 => 1.0101), esponente shift+exp (1+3 => 100) => 0 100 0101

1 110 0111
intero dell'esponente (110=>6), esponente (6-3=3), denormalizzo (1.0111*2^3=1011.1), converto (1*2^3+0*2^2+1*2^1+1*2^0+1*2^(-1) 0 11.5), segno (-11.5)

numeri reali: virgola fissa (limitativi) o mobile

notazione scientifica (numero * base^esponente) => floating point

aumentando l'esp aumentano i numeri e diminuisce la precisione: precisione singola a 32bit o doppia a 64
standard IEEE754 con esp a 8 o 11 bit.
arrotondamento più vicino, per eccesso o a zero

LINGUAGGIO MACCHINA
E' l'insieme delle istruzioni che la CPU può eseguire. Ogni istruzione è composta da un codice operativo, un riferimento all'operando sorgente, un riferimento all'operando risultato e un riferimento alla successiva istruzione. Gli operandi possono essere in RAM, in un registro della CPU o in un dispositivo di I/O.
Le istruzioni sono ovviamente sequenze di bit, ma vengono simbolicamente rappresentate da funzioni come ADD, SUB e LOAD (es. ADD A,B).
Le istruzioni possono essere di 4 tipi: elaborazione dati immagazzinamento dati, trasferimento dati, contro di flusso del programma.
La differenza tra RISC e CISC sta appunto nel numero di indirizzi per ogni istruzione, in quanto possono essere omessi e resi impliciti al fine di semplificare la CPU.
Gli operandi possono essere indirizzi, numeri, caratteri rappresentati in ASCII e dati logici.
Il linguaggio macchina è difficile da scrivere, capire e cambiare ed il programmatore deve occuparsi anche della gestione della RAM. Il primo passo di astratizzazione è il linguaggio Assembler. Qui ogni istruzione ha un codice mnemonico e anche gli indirizzi e i dati sonofacilmente individuabili.

MODI DI INDIRIZZAMENTO
Esistono diversi metodi per specificare l'indirizzo degli operandi:
Immediato: l'operando è parte dell'istruzione, non c'è alcun accesso in memoria ma il valore passato è limitato dalle dimensioni del campo indirizzo
Diretto: viene passato l'indirizzo di memoria, ma c'è comunque uno spazio di indirizzamento limitato
Indiretto: il campo indirizzo contiene l'indirizzo di una cella di memoria che a sua volta contiene l'indirizzo dell'operando. Vengono effettuati 2 accessi in memoria ma c'è ampio spazio di indirizzamento.
Registro: l'indirizzo punta ad un registro. E' ottima per la velocità, in quanto le istruzioni sono più corte, ma ci sono un numero limitato di registri.
Registro indiretto: stesso principio dell'Indiretto, ma con un registro. Risolve il problema dello spazio di indirizzamento.
Con spiazzamento: il campo indirizzo è diviso in 2, un campo è il valore di base e l'altro punta ad un registro contenente l'indirizzo del valore secondario.
Relativo: il campo indirizzo è diviso in 2, un campo è il valore di base e l'altro punta al PC dal quale, di fatto, otteniamo l'indirizzo secondario (indirizzo A + PC).
Registro-base: sempre 2 parti, A contiene lo spiazzamento e R contiene il puntatore all'indirizzo base (implicito o esplicito).
A pila:è un indirizzamento a registro indiretto con sequenze lineari di locazioni. L'operando è in cima alla pila.

STRUTTURA E FUNZIONI DEL PROCESSORE
La CPU preleva le istruzioni, le interpreta, preleva i dati e li elabora per poi finire scrivendo i dati in memoria. E la parte più importante è proprio quest'ultima, la memoria, in quanto i registri della CPU rappresentano il livello più alto della gerarchia di memoria e costituiscono, di fatto, una scelta progettuale importante.
Ci sono diversi tipi di registri. La prima distinzione è quella tra registri utente, ovvero quelli destinati a contenere dati di elaborazione, e quelli di controllo e di stato usati da CU e SO per controllare l'esecuzione dei programmi e il lavoro della CPU.
I registri utente possono essere ad uso generale o specializzato (memorizzazione dati, indirizzi o codici di condizione). I primi hanno il vantaggio di aumentare la flessibilità per il programmatore, ma portano ad un aumento delle dimensioni delle istruzioni. I secondi... il contrario.
Per le architetture CISC solitamente ci sono tra 8 e 32 registri generali, con meno si avrebbero troppi miss e con una abbondanza non si migliorerebbe la situazione. I RISC invece solitamente ne hanno di più