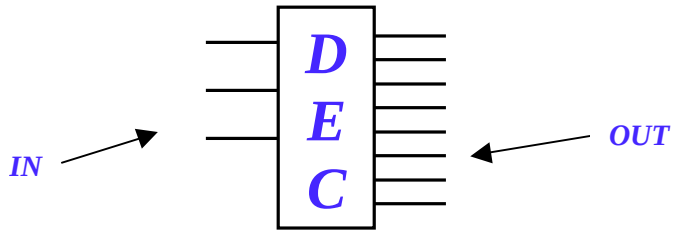
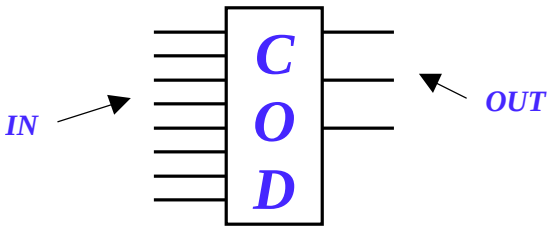
Prontuario di Architettura Degli Elaboratori 2
(Prof. G. A. De Biase)

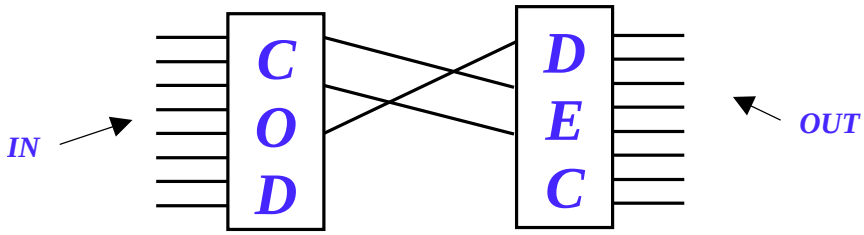
- **decodificatore**
  è un circuito combinatorio che associa alla stringa di input ($\log_2$N bit) una delle N uscite
  Esempio: decodificatore a 8 output e $\log_2 8=3$ input. Le otto combinazioni di input possibili (otto numeri binari) determinano la linea di output da attivare (otto linee)



- **codificatore**
  associa ad ognuna delle N **singole** linee di input una diversa stringa di output di $\log_2$N bit.
  Ha la funzione opposta al decodificatore.

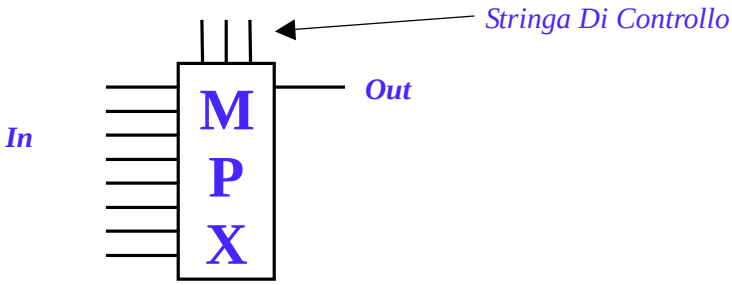


- **transcodificatore**
  può essere composto da un **COD + DEC** (per trasformare un numero in un altro numero tramite inversione dei segnali di uscita)



  oppure di un **DEC + COD** per trasformare una stringa binaria in un'altra, con lo stesso meccanismo.

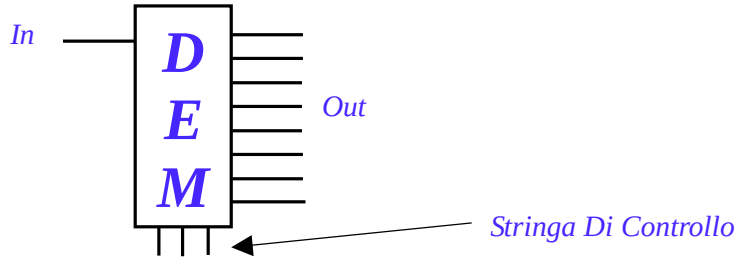
- **multiplexer**
  collega **una** delle N linee di input, scelta dalla stringa binaria di controllo ($\log_2$N linee) all'uscita



ad esempio se la stringa di controllo è “011” verrà selezionala la linea di input numero 3, e l’output verrà collegato a quella linea.

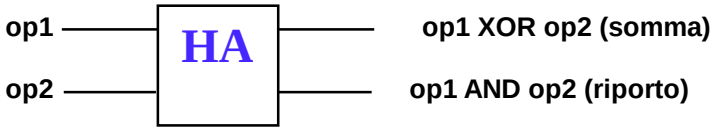
- **demultiplexer**

  - collega l'entrata ad **una** delle N uscite, scelta dalla stringa binaria di controllo ($\log_2 N$ bit)

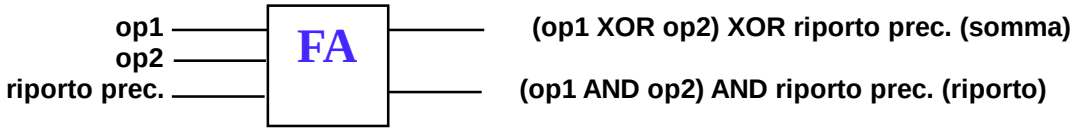


- **addizionatore**

Half Adder: restituisce la somma aritmetica e il riporto dei due bit in input



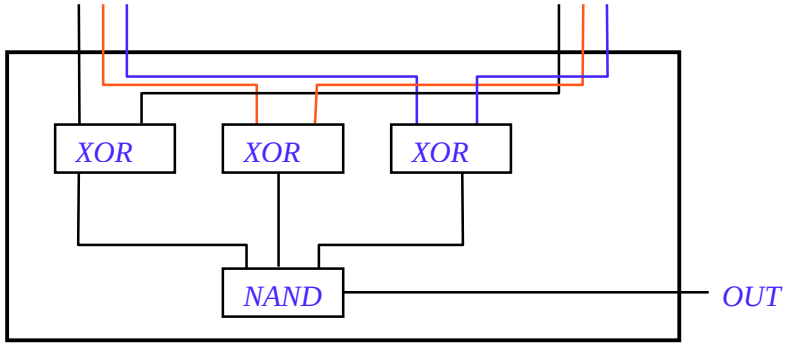
Full Adder: restituisce la somma aritmetica e il riporto dei due bit in input + il riporto della somma precedente



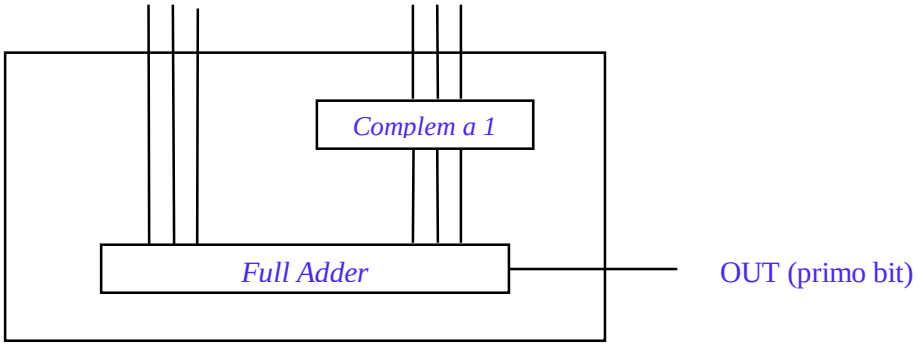
Adder: N Full Adder collegati in serie. Il riporto prodotto da ognuno dei full-adder va nel full-adder successivo. Occorre attendere un tempo di propagazione del riporto, proporzionale a N, per ottenere il risultato della somma

- **comparatore logico**

input = 2 stringhe di N bit
output = 1 se le stringhe sono uguali bit per bit, 0 altrimenti

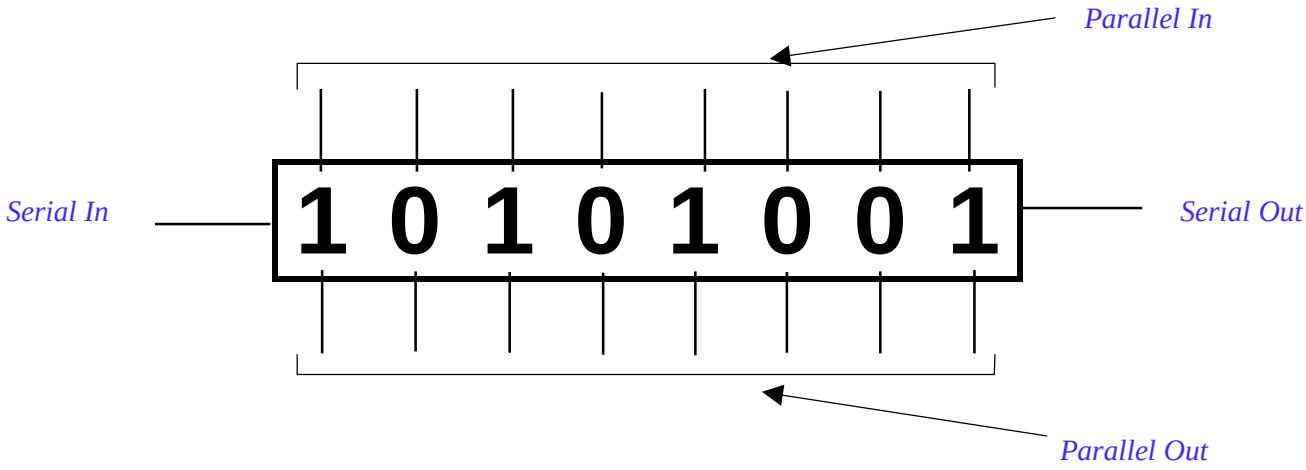


- **comparatore aritmetico**
  - sottrae la seconda stringa alla prima. Se il risultato è negativo (ovvero, se il bit più significativo e' alto) allora la prima stringa è minore della seconda, se il risultato è 0 sono uguali, se è positivo la prima stringa è maggiore della seconda

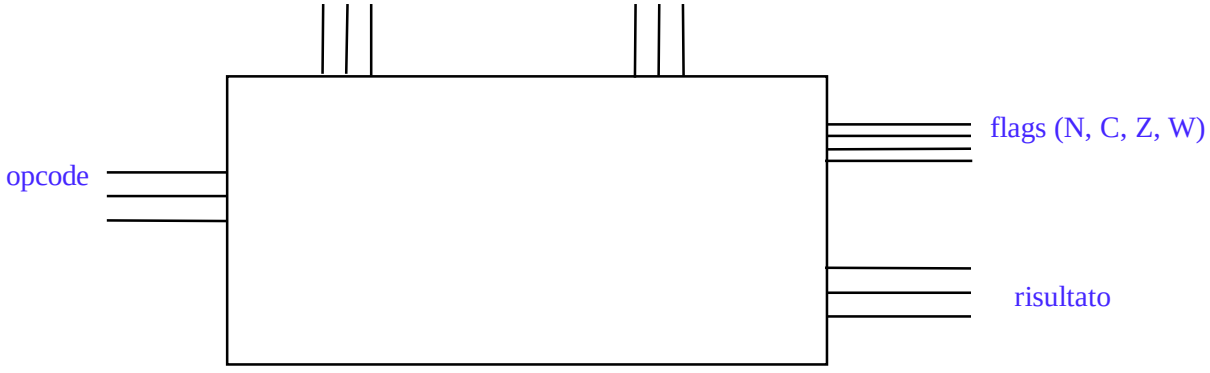


- **contatore**
  - **prefissabile:** può essere caricato con una stringa da cui iniziare il conteggio
  - **mod N:** conta fino a N e poi ricomincia

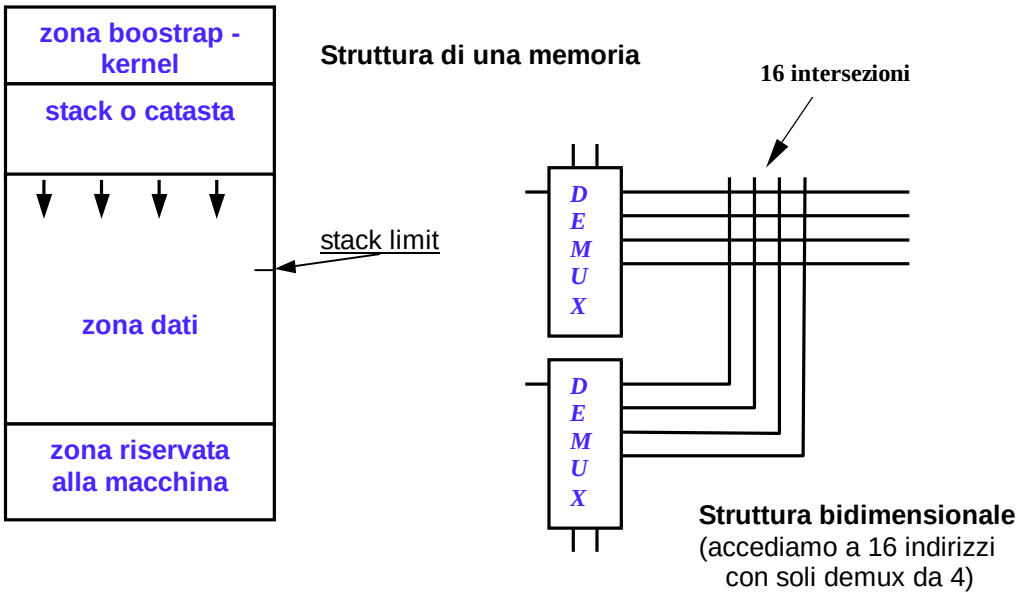
- **shift register**
  - registro che viene caricato per scorrimento (shifting) dei dati al suo interno. Può funzionare da convertitore seriale-parallelo e parallelo-seriale



- **Arithmetic Logic Unit**
  - **input** = 2 operandi di N bit, una stringa di codice dell'operazione
  - **output** = stringa del risultato, uscite logiche Neg, Zero, Carry, OverflowComplemA2
  - **registri** = uno o più accumulatori per le operazioni implicite

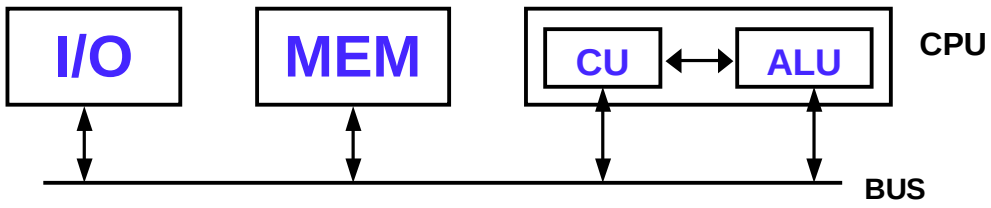


- **memoria, registri di memorizzazione**
  - realizzata come un vettore indicizzato di registri di memorizzazione, dove ogni indice è un indirizzo di memoria. L'accesso alle locazioni di memoria è regolato dai DEMULTIPLEXER spesso collegati in strutture multidimensionali per accedere ad un numero consistente di indirizzi attraverso DEMUX semplici. La lettura e la scrittura dei dati è regolata dai BUFFER TRI-STATE che permettono o inibiscono l'accesso alle locazioni



- **macchine programmabili**
  - macchine in cui è possibile caricare una sequenza di operazioni che la macchina può eseguire serialmente

- **architettura della Macchina di Von Neumann**



Il Bus non è strettamente necessario perché le varie parti potrebbero comunicare anche con i Multiplexer

- **struttura della Control Unit**
  - ***Program Counter*:** contatore prefissabile, cadenzato dal clock di sistema, che punta alla locazione di memoria che contiene la prossima istruzione da caricare nel decodificatore delle istruzioni
  - ***Decodificatore delle istruzioni*:** converte le istruzioni in comandi
  - ***Codificatore dei comandi*:** dirotta i comandi all'ALU e agli altri elementi della MVN
  - ***Stack Pointer*:** contiene la posizione corrente in memoria del puntatore di catasta
  - ***Registro degli errori*:** contiene il codice dell'errore che si è verificato per ultimo nella macchina
  - ***Registri dell'utente*:** memorizzano i dati per aiutare nella programmazione. L'accesso ad essi è molto veloce. Il loro numero varia secondo l'architettura della macchina.
  - ***Registro di stato*:** memorizza le informazioni principali della macchina (abilitazione dell'interruzione, abilitazione stato supervisore)

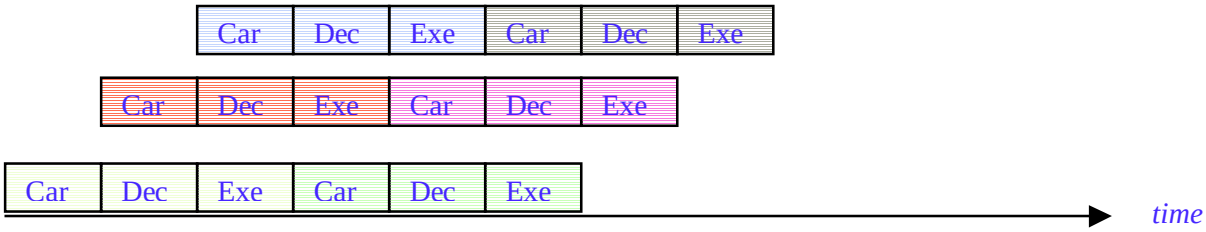
- **istruzioni e fasi**
  - **Fetch:** il PC punta ad un indirizzo di memoria, il cui contenuto viene caricato nel decodificatore

- **Decode:** l'istruzione viene convertita in singoli comandi dal decodificatore che vengono trasmessi al codificatore dei comandi
- **Execute:** i comandi e gli operandi vengono trasmessi dal codificatore alle altre unità della MVN tramite il bus

I dati e gli operatori sono entrambi in memoria e possono essere confusi nella fase di fetch in caso di cattivo allineamento del PC. Questo è il principale difetto della Macchina di Von Neumann

• canalizzazione o pipelining

E' una tecnica di sovrapposizione delle fasi Caricamento-Decodifica-Esecuzione con la quale le istruzioni possono essere processate a gruppi di tre:



• classi di istruzione

- **Istruzioni Logico-Aritmetiche:** coinvolgono l'ALU o l'Unità Aritmetica. (es. ADD, SUB, OR)
- **Istruzioni di trasferimento dati:** spostano i contenuti di un registro o di una locazione di memoria.
  LOAD: LocazioneDiMemoria->RegistroCPU
  STORE: RegistroCPU->LocazioneDiMemoria
  MOVE: RegistroCPU->RegistroCPU
- **Comandi:** istruzioni generali di gestione macchina. (es. WAIT, STOP, POWER DOWN)
- **Istruzioni di salto:** caricano una nuova stringa nel PC e continuano il programma da lì
  ***Salto incondizionato*:** salta senza condizioni ad un nuovo indirizzo di memoria (es. JUMP, BRANCH)
  ***Salto condizionato*:** salta se si verificano certe condizioni (controlla FLAG) (es. BNEZ, BEQ)
  ***Salto a subroutine*:** prima di saltare salva nello stack i registri utili che non devono essere sovrascritti (stato volatile della macchina) e l'indirizzo successivo a quello dell'istruzione di salto, per potervi ritornare alla fine della subroutine

• referenziamento degli operandi

- ***implicito*:** gli operandi non sono specificati direttamente ma l'istruzione sa già dove trovarli
- ***parzialmente implicito*:** vengono specificati solo alcuni degli operandi
- ***esplicito*:** gli operandi vengono specificati direttamente nell'istruzione

• modi di indirizzamento

Esistono 16 modi di indirizzamento principali, che hanno per parametri registri della CPU e indirizzi di memoria

Esempio: istruzione OPC (fittizia)

Legenda: (x) = contenuto dell'indirizzo x
[x] = contenuto del registro x

1. ***Immediato*:** OPC number
   Si passa come parametro all'istruzione direttamente il valore
2. ***Assoluto*:** OPC (address)
   Il parametro è l'indirizzo di memoria che contiene il valore
3. ***A registro*:** OPC [reg]
   Il parametro è il registro CPU che contiene il valore
4. ***A registro differito*:** OPC ([reg])
   Il parametro è il registro che contiene l'indirizzo di memoria che contiene il valore

5. ***Indiretto*:** OPC ((address))
   Il parametro è l'indirizzo di memoria che contiene un altro indirizzo in cui c'è il valore
6. ***Differito indiretto*:** OPC (([reg]))
   L'indirizzo indiretto è a sua volta contenuto in un registro della CPU
7. ***Relativo*:** OPC (PC+number)
   Il valore si trova all'indirizzo che sta "number" posti dopo la posizione corrente del PC
8. ***Relativo indiretto*:** OPC ((PC+number))
   Il valore si trova all'indirizzo contenuto nell'indirizzo che sta "number" posti dopo la posizione corrente del PC
9. ***Relativo a registro differito*:** OPC (PC+[reg])
   Il valore è contenuto nell'indirizzo dato dalla somma del valore corrente del PC + un numero specificato nel registro reg
10. ***Relativo a registro differito indiretto*:** OPC ((PC+[reg]))
    Il valore è contenuto nell'indirizzo specificato all'indirizzo dato dalla somma del valore corrente del PC + un numero specificato nel registro reg
11. ***Relativo indicizzato*:** OPC (INDEX+number)
12. ***Relativo indiretto indicizzato*:** OPC ((INDEX+number))
13. ***Relativo a registro differito indicizzato*:** OPC (INDEX+[reg])
14. ***Relativo a registro differito indiretto indicizzato*:** OPC ((INDEX+[reg]))
    Come i modi da 7 a 10, con un registro INDEX che fa le veci del PC
15. ***Differito a registro con autoincremento*:** OPC ++([reg])
    Un registro contiene l'indirizzo dell'operando. Dopo l'uso il valore dell'indirizzo viene incrementato. Questo è un indirizzamento utile per scorrere tabelle o matrici residenti in memoria
16. ***Differito a registro con autodecremento*:** OPC --([reg])
    Un registro contiene l'indirizzo dell'operando. Dopo l'uso il valore dell'indirizzo viene decrementato. Questo è un indirizzamento utile per scorrere tabelle o matrici residenti in memoria

- ## registro di stato e registri speciali

- ***Registro di stato*:** contiene informazioni come la priorità dell'operazione corrente, i bit logici in uscita dall'ALU (flags), i bit di abilitazione/mascheramento delle interruzioni e altre informazioni della macchina
- ***Registro degli errori*:** contiene il codice dell'ultimo errore verificatosi durante l'esecuzione di un programma. Fra i possibili errori ci sono lo stack overflow, il cattivo allineamento nella lettura o scrittura dei dati, gli errori aritmetici come la divisione per zero, l'overflow nel calcolo di numeri interi o in complemento a 2 ecc.
- ***Stack pointer*:** contiene l'indirizzo corrente del puntatore di catasta, cioè l'ultima locazione di memoria usata per immagazzinare dati nell'area di memoria riservata alla pila di sistema
- ***Index*:** è l'indirizzo usato per i modi di indirizzamento indicizzato degli operandi (modi dall'11 al 14)

- ## pila di sistema o catasta

è una zona di memoria, situata nella parte alta prima del boostrap, nella quale vengono caricati dei dati ogni volta che il programma salta ad una sottoprocedura o alla routine di gestione di un'interruzione. La catasta è gestita come una pila LIFO e contiene di volta in volta lo STATO VOLATILE della macchina, cioè l'insieme dei registri che cambiano il proprio valore ad ogni chiamata a sottoprocedura e che quindi devono essere salvati per essere ripristinati quando si ritorna alla procedura principale o chiamante. L'indirizzo dell'ultima locazione di memoria occupata per l'"impilamento" dei dati è data dallo ***stack pointer***, che viene decrementato ogni volta che un blocco di dati viene salvato. Quando lo stack pointer sta per avvicinarsi allo ***stack limit***, l'indirizzo di memoria oltre il quale inizia il segmento dati, si ha un SYSTEM WARNING o errore giallo. Il raggiungimento dello stack limit porta invece all'ERRORE ROSSO, che non è recuperabile. Lo stack è particolarmente utile nell'esecuzione di sottoprocedure ANNIDATE (ad esempio quelle ***ricorsive***)

- ## unità aritmetica e richiamo dei numeri in complemento a due

- l'unità aritmetica (UA) svolge i calcoli che non rientrano nelle possibilità dell'ALU, come ad esempio le operazioni sui numeri floating point o a doppia precisione, il calcolo di radici quadrate e così via. E' collegata al bus ed è considerata come un qualsiasi dispositivo che comunica con la CPU, tant'è che mentre essa svolge i suoi calcoli la CPU può continuare a lavorare. L'UA può essere gestita dalla macchina in ***modalità nativa***, se il decodificatore della Control Unit prevede già l'utilizzo di istruzioni per un'unità aritmetica, o in ***modalità attaccata***, se invece occorre un decodificatore a parte per le nuove istruzioni.

- il ***sistema di numerazione in complemento a due*** permette di rappresentare i numeri negativi in maniera alternativa al sistema modulo-segno (primo bit riservato al segno, i restanti n-1 bit al modulo). In una stringa di N bit letta come numero in complemento a due si ha che il primo bit ha PESO NEGATIVO, cioè viene considerato come numero negativo nella traduzione in decimale. Per convertire un numero binario nel suo opposto in complemento a due basta invertire i suoi bit e aggiungere 1. Il vantaggio di questo sistema di numerazione è la rappresentazione del numero 0 in maniera univoca come "000…0". Nel sistema modulo-segno, invece, sono equivalenti le scritture "1000…0" (zero con segno negativo) e "000…0" (zero con segno positivo).

- ## moltiplicazione, divisione

Il risultato di una moltiplicazione di numeri a n bit necessita di 2n bit per essere memorizzato

- ## interruzioni interne ed esterne, trap

Le interruzioni sono segnali inviati all'unità centrale di processo che cercano di fermare il programma per far eseguire alla macchina altre operazioni. Prima di gestire l'interruzione e passare ad eseguire un'altra operazione il programma deve salvare nello stack lo STATO VOLATILE della macchina per ripristinarlo al ritorno dall'interruzione

- ***interne (eccezioni)*****:** sono causate da errori interni della macchina, ad esempio errori di calcolo e stack overflow. Sono SINCRONE, perché si verificano in concomitanza con un ciclo di clock, essendo causate dall'esecuzione di un'istruzione
- ***esterne*****:** sono inviate alla CPU dall'esterno della macchina, ad esempio dai dispositivi di I/O, e sono ASINCRONE
- ***trap*****:** sono interruzioni software gestite dall'utente che vengono inserite nel programma solitamente per caricare pagine di memoria o per far partire delle routines di gestione della macchina
- ***vettorizzate*****:** l'arrivo di queste interruzioni carica nel registro degli errori un codice, che porta la macchina ad identificare, tramite il VETTORE DELLE INTERRUZIONI (situato nella zona di memoria riservata) l'indirizzo di partenza della procedura che gestisce l'interruzione arrivata.
- ***a polling*****:** l'arrivo di queste interruzioni fa partire una procedura di ricerca dal dispositivo che ha lanciato l'interruzione. Il metodo è poco conveniente se i dispositivi da gestire sono molti
- ***priorità*****:** un'interruzione deve portare con sé un codice di priorità, che deve essere confrontato con i bit dell'operazione corrente (scritti nello status). Se l'interruzione arrivata ha importanza maggiore (ha un codice di priorità più alto) viene gestita, altrimenti viene memorizzata tramite lo stack per poter essere eseguita alla fine dell'operazione corrente. Il sistema può decidere, tramite opportuni settaggi nel registro di stato, di INIBIRE COMPLETAMENTE LE INTERRUZIONI ESTERNE (***masking***) per non "disturbare" un'operazione importante
- ***macchina multiprogrammata*****:** utilizza le trap per eseguire più processi nello stesso tempo. Quando uno di essi è impegnato in operazioni di I/O o in altri compiti che non coinvolgono la CPU si fa partire un altro processo con una trap per ottimizzare i tempi di computazione e per non lasciare ferma la CPU

- ## operazioni di I/O

- l'I/O è gestita come un'estensione della memoria fisica. Se un programma cerca di leggere o scrivere su una locazione di memoria con un'indirizzo troppo grande allora, con una trap, il sistema operativo inizia la comunicazione con un dispositivo di I/O. Solitamente ogni dispositivo ha un registro o una locazione di memoria (porta) che contiene il dato letto o il dato da scrivere, e un proprio registro di stato, che deve contenere fra gli altri:
  - il bit di abilitazione all'invio di interruzioni alla CPU
  - il bit di dispositivo pronto alla lettura dei dati (se prevista)
  - il bit di dispositivo pronto alla scrittura dei dati (se prevista)

Il ***metodo di lettura o scrittura programmata*** dei dati sull'I/O interrompe il programma corrente finché la periferica non riceve o invia il dato. Un metodo alternativo consiste nel far continuare a lavorare la CPU e attendere l'interruzione esterna inviata dalla periferica per gestire lo spostamento dei dati. Nei sistemi non multiprogrammati si può ovviamente utilizzare solo il primo metodo, mentre negli altri il secondo metodo è sicuramente più conveniente in termini di tempo

- ## accesso diretto alla memoria (DMA)

- è una modalità di trasferimento dei dati dall'I/O alla memoria e viceversa, solitamente utilizzata sui dispositivi veloci, che non coinvolge la CPU, ma è coordinata da un GESTORE di DMA. Perché possa

partire un'attività DMA il gestore deve contenere tre dati in opportuni registri:
- lo ***stato*** del DMA (abilitazione, verso di trasferimento, dispositivo pronto in lettura/scrittura)
- l'***indirizzo*** di memoria dal quale bisogna iniziare la lettura o la scrittura dei dati
- un ***contatore*** di parole trasferite che decrementa ad ogni spostamento di dati

Il dispositivo DMA invia l'interruzione alla CPU e, se essa è accettata, il trasferimento dei dati inizia. Il contatore decrementa e quando raggiunge lo zero il dispositivo DMA manda un'interruzione per segnalare la fine del trasferimento.
Perché inizi l'attività DMA il programmatore deve accertarsi che ci sia un'area di memoria sufficientemente grande nell'unità di destinazione per "ospitare" i dati in arrivo.
Notare che durante il DMA la CPU è libera ma il BUS è impegnato

- **condivisione del bus**

- un ARBITRATORE DEL BUS risolve i conflitti dei vari moduli della MVN per l'utilizzo del bus a seconda della priorità delle operazioni. Il sistema operativo e le sue operazioni hanno ovviamente priorità assoluta

- **microprogrammazione**

- nella Control Unit si può sostituire (legge dell'***equivalenza hardware-software***) alla coppia decodificatore-codificatore un sistema software, la ***MICROMACCHINA***, che al riconoscimento di un'istruzione fa partire un ***microprogramma*** associato (contenuto in una ***micro-ROM*** scandita da un ***micro-PC***) fatto di singoli comandi che possono essere inviati direttamente ai componenti della MVN. Il vantaggio è che, volendo potenziare il calcolatore aggiungendo nuove istruzioni non bisognerà più rimuovere la complessa unità DEC-COD, ma basterà riprogrammare la micro-ROM. Il ritorno alle macchine RISC ha portato al progressivo abbandono delle architetture microprogrammate. L'uso della micromacchina è ***trasparente*** al programmatore

- **macchine CISC e RISC**

- ***Complex Instruction Set Computers*****:** il gruppo DEC-COD all'interno della CU è molto complesso e accetta un grande numero di istruzioni
- ***Reduced Instruction Set Computers*****:** il gruppo DEC-COD è molto più semplice e le istruzioni riconosciute sono in numero ridotto e poco complesse, tanto che la loro esecuzione impiega solo un colpo di clock. Occorrono più colpi di clock se si vuole simulare un'istruzione complessa tipo CISC. Il ritorno a questo tipo di architetture negli ultimi anni è stato dettato dal fatto che le istruzioni complesse sono usate pochissimo, e che il codice di una macchina RISC può essere trasportato con molta facilità su un'altra di queste macchine (portabilità del software)

- **linguaggio macchina**

- è il linguaggio di livello più basso, l'unico che il calcolatore riesce a comprendere. Ogni codice deve essere tradotto in linguaggio macchina per funzionare su un calcolatore. E' un insieme di stringhe binarie del tipo <istruzione> <op1>…<opn> che vengono eseguite sequenzialmente. Le stringhe degli comandi possono essere facilmente confuse con quelle degli operandi.

- **modulo eseguibile, collegatore**

- un programma scritto in un qualsiasi linguaggio, al momento della compilazione (traduzione in linguaggio macchina) può non essere pronto per l'esecuzione. Esso infatti può fare riferimento a delle variabili globali specificate in un altro codice, o a delle librerie di sistema i cui indirizzi non sono noti. Un codice macchina tale viene detto MODULO OGGETTO, ed ha bisogno dell'azione di un LINKER o collegatore che lo metta in relazione con altri moduli oggetto per risolvere gli indirizzi o le variabili specificate in quei moduli. Un codice è detto ***RILOCABILE*** o ***PIC*** **(Position Independent Code)** se può essere caricato ed eseguito in una qualsiasi zona del segmento dati. Il codice PIC contiene istruzioni di salto con offset molto limitato, in modo da non invadere zone di memoria protette

- **sottoprogrammi**
- **assembler e suo traduttore**

- l'assembly è il linguaggio più vicino al codice macchina. Al posto delle stringhe binarie corrispondenti alle istruzioni ci sono degli mnemonici, detti OPCODES, che rendono più chiaro al programmatore qual è l'operazione da svolgere. Le direttive sono comandi speciali che sono diretti al traduttore del linguaggio assembly e non alla macchina (es. "**.word**" "**.byte**"). Diversamente da ogni compilatore dei linguaggi

evoluti nell'assembler la corrispondenza istruzione-comandi è uno a uno

- **boostrap**

era un programma che veniva caricato manualmente nella parte alta della memoria e serviva a caricare dalle periferiche di massa le librerie di sistema, i drivers e le procedure di gestione ogni volta che la macchina veniva accesa. Successivamente il boostrap venne corredato da un monitor che stabiliva come e quando caricare i dati. I boostrap monitors divennero man mano sempre più evoluti fino a diventare sistemi operativi.

- **memoria cache**

è una memoria di piccole dimensioni che contiene gli ultimi dati della RAM a cui c'è stato un accesso. Essa ha una velocità di accesso molto maggiore di quella della RAM, e comunica direttamente con essa tramite un bus molto largo. Solitamente viene copiato nella cache memory un "intorno" del PC, cioè un insieme più o meno grande di locazioni di memoria adiacenti all'indirizzo specificato nel PC, e la CPU comunica direttamente con essa. La cache velocizza l'accesso ai dati ma è molto costosa. L'uso della cache è TRASPARENTE al programmatore

- **memoria a pagine**

- è un sistema di gestione della memoria usato quando la memoria disponibile è **PIU' GRANDE** di quella accessibile. Esso prevede l'utilizzo di un ulteriore registro contenuto in una ***Memory Management Unit*** (MMU), chiamato registro di estensione. La memoria disponibile viene divisa in un numero di pagine, o blocchi, grandi quanto la memoria indirizzabile, e si utilizza solo UNA PAGINA DI MEMORIA PER VOLTA. Sono vietate, ovviamente, le istruzioni di salto da una pagina di memoria all'altra. Il caricamento delle pagine di memoria e la loro gestione ***non è trasparente*** al programmatore

- **memoria virtuale**

- quando occorre caricare in memoria un programma più grande della memoria stessa si ricorre alla memoria virtuale. Serve innanzi tutto una memoria di massa che ospiti temporaneamente i dati, e si può ricorrere a due diversi espedienti:
  - ***metodo dell'overlay*****:** il programma viene suddiviso in più moduli disposti ad albero, e viene caricata in memoria solo una parte di essi, costituita da una root (modulo correntemente in esecuzione) e da altri moduli che sono direttamente coinvolti nell'esecuzione di root (es. con un salto da root a quel modulo). Ogni volta che bisogna caricare un nuovo gruppo di moduli (uno dei cammini dalla root ad una foglia dell'albero) lo si fa grazie ad una trap.
  - ***metodo delle pagine di memoria*****:** se il programma occupa più della memoria si carica dalla memoria di massa una sola parte del programma grande come la memoria fisicamente disponibile. Quando si finisce di eseguire quella parte di programma si ricarica nella memoria, grazie ad una trap, un'altra pagina di programma dalla periferica.

Entrambi i metodi ***non sono trasparenti*** al programmatore

- **esempio di processore: serie PDP11 della Digital Equipment Company**

Tipo di architettura**:** CISC, word a 16 bit, possibilità di indirizzamento a word (solo indirizzi pari) o a byte. 8 livelli di priorità (3 bit riservati alla priorità corrente nel registro status (indirizzo FFFE))

8 Registri**:** R0-R5 ad uso comune, R6 (Stack Pointer), R7 (Program Counter). Erano TUTTI ACCESSIBILI AL PROGRAMMATORE IN LETTURA E SCRITTURA.

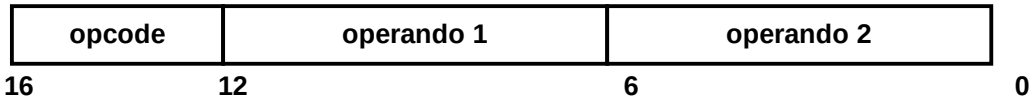
8 Modi di indirizzamento **+** ***4 orientati al PC*****:**

0.	a registro	**Rx**
1.	a registro differito	**(Rx)**
2.	ad autoincremento	**(Rx)+**
3.	differito ad autoincremento	**@(Rx)+**
4.	ad autodecremento	**-(Rx)**
5.	differito ad autodecremento	**@-(Rx)**
6.	a indice	**X(Rx)**
7.	a indice differito	**@X(Rx)**

0PC immediato
1PC assoluto
2PC relativo
3PC relativo differito

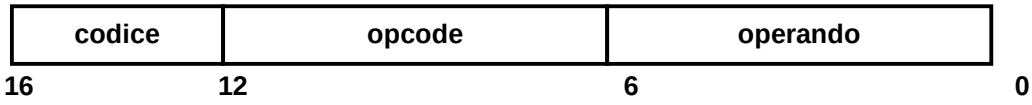
***Formato delle istruzioni*:**

Due operandi

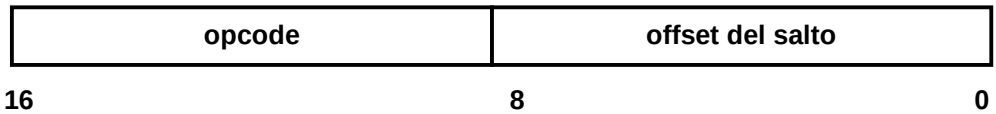


I 6 bit degli operandi erano così suddivisi: 3bit = modo di indirizzamento, 3bit = num. registro
Se il bit 16 era 0 l'operazione era orientata alla ***word***, se era 1 al ***byte***.
Se i bit dal 15 al 13 valevano 000 l'operazione veniva decodificata normalmente, se valevano 111 veniva usato il decodificatore dell'Unità Aritmetica ***attaccata***. In questi casi l'operazione era ad ***un operando***

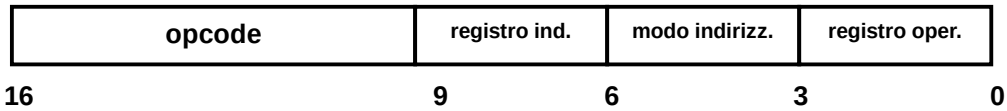
Un operando



Salti



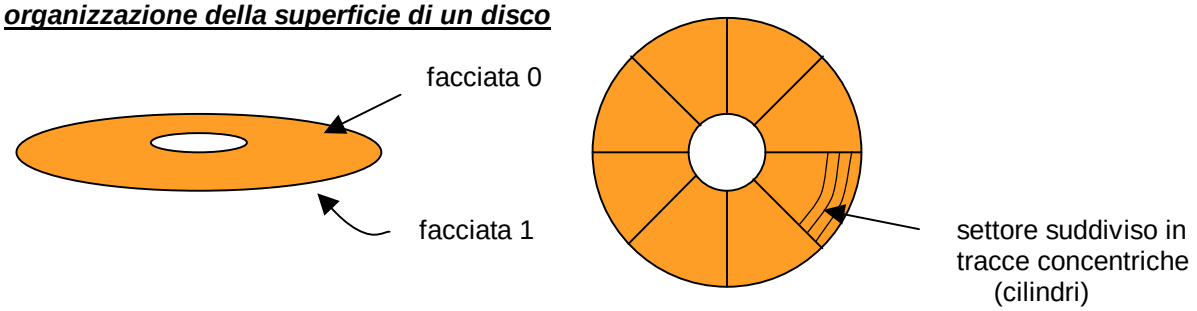
Salti a sottoprocedura



registro ind. è il numero del registro che contiene l'indirizzo della subroutine
modo indirizz. è il modo di indirizzamento con cui accedere agli operandi
registro oper. è il numero del registro che contiene l'indirizzo degli operandi della subroutine

- **unità disco**

organizzazione della superficie di un disco



Ogni informazione sul disco è reperibile in un "indirizzo" dato da una terna (faccia, settore, cilindro)
Si utilizzano delle testine per leggere e scrivere i dati. Il disco ruota e la testina si muove in senso radiale sulla superficie del disco finché essa non si posiziona sul record da usare.

formattazione, accesso casuale

E' un'operazione che consiste nella suddivisione logica della superficie del disco (o delle due superfici) in record, e nell'assegnazione di una terna (faccia, settore, cilindro) a ogni record del disco. La formattazione determina la capacità di un disco, e dipende dal sistema operativo utilizzato. Ogni traccia scrivibile del disco ha una capacità prefissata (di solito 512 bytes) che non dipende dalla dimensione fisica della traccia. Le tracce più vicine al centro del disco contengono 512 bytes come quelle più lontane dal centro. La scrittura di informazioni su un disco può ovviamente richiedere più di 512 bytes, e per questo si ricorre a dei puntatori al prossimo record usato, che vengono scritti alla fine del record usato. Per memorizzare i dati si usa il prossimo record libero. L'accesso all'informazione viene definito quindi CASUALE, perché una stessa informazione può essere memorizzata in differenti punti del disco, che hanno tutti lo stesso tempo di accesso.

deframmentazione

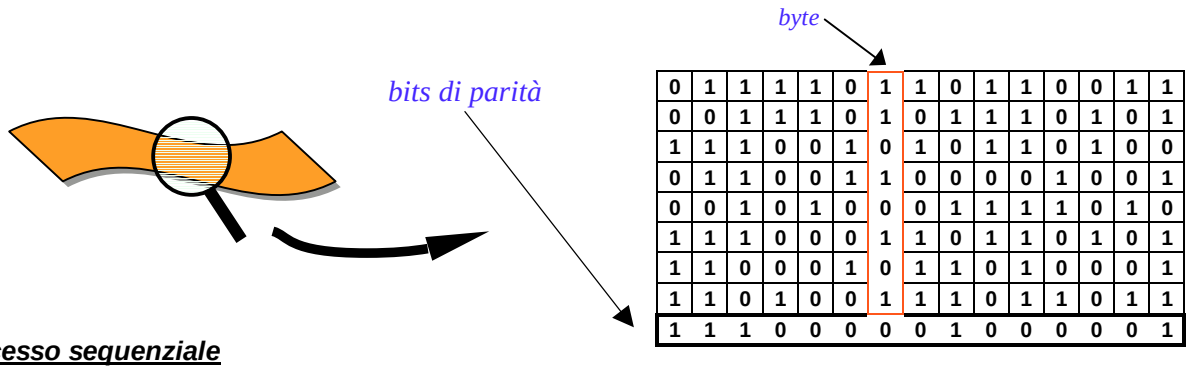
Una serie di scritture e cancellazioni di dati sul disco possono portare alla memorizzazione non contigua delle informazioni. Uno stesso file viene memorizzato spesso in record molto lontani fisicamente fra loro, in mancanza di record vicini liberi. L'accesso alle informazioni risulta quindi rallentato a causa delle continue rotazioni che il disco deve compiere per far sì che la testina legga i dati. La deframmentazione pone rimedio a questo problema, spostando i record appartenenti ad uno stesso dato in modo da metterli più vicini possibile.

hard disk, floppy disk e loro struttura

L'hard disk ha di solito capacità molto grande ed è usato per le operazioni di swap con la memoria principale e per l'immagazzinamento dei dati a lungo termine. Il disco fisico è chiuso in una scatola di metallo impermeabile alla polvere, e la testina di lettura/scrittura non tocca la superficie del disco, ma si mantiene a qualche micron di distanza, per consentire una maggiore velocità di rotazione del disco.
Il floppy disk invece ha una capacità più limitata, ed è usato prevalentemente per l'immagazzinamento di informazioni a breve termine. La velocità di lettura è minore di quella dell'hard disk perché c'è contatto fisico fra testina e superficie del disco. Il disco magnetico è racchiuso in un involucro di plastica.

- **unità nastro**

Un altro supporto magnetico usato è il nastro, una lunga fettuccia di plastica rivestita di materiale magnetico.Il nastro scorre mentre una testina legge o scrive informazioni, che sono memorizzate in questo modo:



0	1	1	1	1	0	1	1	0	1	1	0	0	1	1
0	0	1	1	1	0	1	0	1	1	1	0	1	0	1
1	1	1	0	0	1	0	1	0	1	1	0	1	0	0
0	1	1	0	0	1	1	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	0	1	1	1	1	0	1	0
1	1	1	0	0	0	1	1	0	1	1	0	1	0	1
1	1	0	0	0	1	0	1	1	0	1	0	0	0	1
1	1	0	1	0	0	1	1	1	0	1	1	0	1	1
1	1	1	0	0	0	0	0	1	0	0	0	0	0	1

accesso sequenziale

Uno stesso file non può essere memorizzato in maniera “sparsa” sul nastro. Per leggere un dato bisogna scorrere tutti i dati che lo precedono sul nastro. Per questo l'accesso ai dati si dice sequenziale.

bit di parità orizzontale e trasversale

Solitamente vengono memorizzate su nastro le informazioni che devono essere mantenute per tempi lunghissimi, dell'ordine di anni, ad esempio i database, gli archivi e i backup. E' per questo che il nastro deve garantire la sicurezza delle informazioni memorizzate. I bit di parità servono appunto per verificare anomalie nei dati memorizzati e per correggerli quando possibile. Oltre gli otto bit che si leggono trasversalmente sul nastro ce n'è un altro, quello di parità trasversale, che vale 1 se il numero di bit "1" (o di bit “0”, tanto è lo stesso) nel byte corrispondente è pari. Un eventuale byte errato viene scoperto grazie a questo nono bit. L'errore si può correggere, il più delle volte, grazie al bit di parità orizzontale. Si tratta di un byte di controllo, scritto automaticamente dopo 8 byte di informazioni, che ha la stessa funzione dell'altro bit di parità ma in senso longitudinale. Si riesce così, la maggior parte delle volte, a individuare il bit fasullo e ad invertirlo. Questo è un esempio su una sezione di nastro registrato:

1	1	1	1	0	0	1	0	0	0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	0	1	0	1	0	1
1	1	0	1	1	0	0	0	1	0	1	0	1	0	0	1	0	0	0
1	1	0	1	0	1	1	1	0	1	1	1	0	1	1	0	1	1	1
1	0	1	0	1	0	0	1	1	0	0	1	0	0	1	0	0	1	1
1	1	1	1	0	1	0	1	1	1	0	1	0	1	0	1	1	0	0
0	1	0	1	1	0	1	0	1	1	0	1	1	0	0	0	1	1	1
1	1	0	1	0	1	0	1	0	1	1	0	1	1	0	1	0	0	1
1	0	1	1	0	1	1	0	X	0	1	0	0	1	1	0	1	X	1

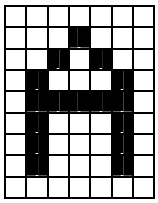
Ai due bit di parità (segnati in rosso) non tornano i conti. Il bit all'incrocio (in blu) va invertito.
Il maggiore difetto di questa tecnica di correzione è la possibilità che i bit di controllo vengano danneggiati. In questo caso il controllo di parità andrebbe a modificare dei dati corretti, senza possibilità di recupero.

differenze con i dischi
La velocità di accesso è minore, ma l'informazione resiste di più sul supporto del nastro.

• terminale video, stampanti laser

pennello elettronico
Il principio di funzionamento di queste due periferiche è lo stesso: il terminale video è suddiviso in un certo numero di righe e colonne, e un pennello elettronico scandisce le righe una ad una disegnando l'immagine. Il pennello elettronico ovviamente compie centinaia di scansioni al secondo, in modo che il cambio di immagine sia il più possibile rapido. Nella stampante laser, in modo analogo, la testina scrive sul foglio riga per riga, passando eventualmente più volte su ogni riga per una maggiore definizione.

bitmap carattere per carattere
Per rappresentare i caratteri occorre uno spazio nella memoria principale che memorizzi la loro forma, in modo da disegnarla sullo schermo quando necessario. Ad ogni carattere viene quindi associata una bitmap (insieme di bit) di dimensione definita. Questa ad esempio potrebbe essere la bitmap per un'A maiuscola:



bitmap schermo intero
L'avvento dei sistemi a interfaccia grafica hanno portato i progettisti ad abbandonare l'idea di una bitmap per ogni carattere, e si è scelto di destinare una zona di memoria a contenere la bitmap di tutta la corrente schermata. Si associa così un bit ad ogni punto dello schermo invece di comporre l'immagine con le bitmap dei caratteri. In questo modo è anche possibile cambiare i fonts dei caratteri sullo schermo o disegnare immagini, semplicemente ricalcolando la screen bitmap

bitmap multicolore
Per gestire più colori sullo schermo si è pensato di moltiplicare il numero di screen bitmap residenti in memoria, associando più bit a ogni punto dello schermo. Ad ogni combinazione di bit corrisponde un colore diverso. Negli schermi moderni vengono associati 3 byte (quantità di rosso, quantità di verde, quantità di blu) a ogni pixel (punto colorato) dello schermo. Si parla di bitmap a 24 livelli (3 bytes di 8 bit ciascuno)

Quantità di rosso	*Quantità di verde*	*Quantità di blu*

24 bit per ogni pixel

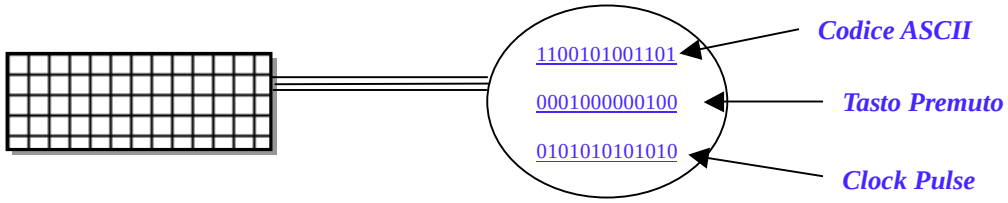
- **tastiera**

La tastiera è il dispositivo primario di input di ogni sistema operativo. Essa comunica con il computer tramite una porta, a cui invia dei caratteri che vengono trasformati in codici ASCII digitali da un'unità chiamata ***UART***. La pressione di un tasto (un semplice interruttore elettrico) viene notificata a un codificatore che riconosce quale degli interruttori è stato premuto, e genera il codice ASCII del tasto su otto linee parallele. Queste linee vanno in uno shift register che provvede a trasformare il segnale da parallelo a seriale e a inviarlo allo standard input. La pressione di tasti speciali come CTRL, SHIFT ecc. in combinazione dei normali tasti indica al codificatore di utilizzare un sottoinsieme di codici diverso.

***modi di trasmissione*:**

1 filo (sincrono non cadenzato)
Il convertitore Parallelo-Seriale utilizza un solo filo elettrico per comunicare con il decodificatore, su cui viene trasmesso il codice ASCII del tasto premuto più una particolare sequenza di bit che indica l'inizio di un nuovo carattere. Di solito quest'ultima non serve perché la lunghezza delle stringhe è prefissata

3 fili (sincrono cadenzato)
La trasmissione dei dati avviene su tre fili. Sul primo di essi viaggia un segnale di clock, che sincronizza (cadenza) i bit trasmessi dalla tastiera, sul secondo il flusso di bit corrispondenti ai codici ASCII dei tasti premuti, e sul terzo un altro flusso di bit che segna 1 quando un tasto è stato premuto, e 0 quando non è stato premuto alcun tasto



- **stampanti a matrice**

la matrice ad aghi della stampante corrisponde alla bitmap di un carattere e ha le sue stesse dimensioni (esempio: i caratteri 7x9 pixels possono essere stampati da una matrice di 7x9 aghi)

- **periferiche grafiche: plotter**

Periferica usata per le applicazione grafiche e ingegneristiche. E' una penna robotica che scrive su un piano su cui è appoggiato un foglio da disegno. Per il plotter ci sono 3 soli tipi di istruzioni: muovi la penna a sinistra/destra, muovi la penna su/giù, alza/abbassa la penna dal foglio. Per i plotter multicolore c'è eventualmente l'istruzione cambia colore

- **interfacce di comunicazione: modem**

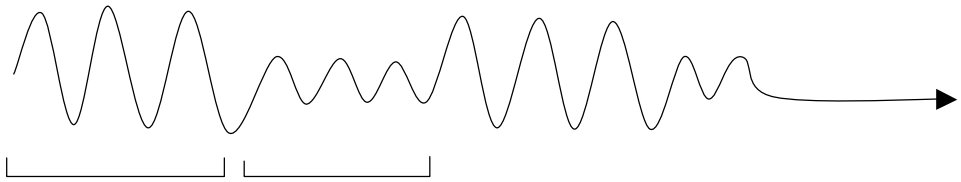
E' il componente hardware che permette a più computers di comunicare attraverso la linea telefonica. Quest'ultima, essendo molto estesa nel mondo, ha il vantaggio di poter collegare computers anche molto lontani fisicamente fra loro, ma allo stesso tempo presenta lo svantaggio di non essere l'ideale per il trasferimento dei dati. Il **modem** (= Moduler DEModuler) ha il compito di trasformare i dati digitali inviati dal computer in segnali analogici e di adattarli alla linea telefonica (invio dati) e di ritrasformare il segnale analogico prelevato dalla linea telefonica in dati digitali da trasmettere al computer.

- *Banda base (stringa binaria continua)*
E' il primo modo di trasportare i dati sulla linea telefonica, e consiste nel trasferire il segnale trasformato sulla linea, senza ulteriori trasformazioni. L'onda che il modem trasmette alla rete telefonica sarà composta semplicemente dai segnali analogici in cui i dati sono stati tradotti. Questo metodo è poco affidabile perché la normale linea telefonica presenta sull'onda trasportata una componente estranea dovuta al rumore, che è tanto più influente quanto è più grande la distanza di comunicazione.

- *portante*
Per questo motivo si è pensato di sommare ai segnali in uscita dal modem un'onda portante, i cui parametri (frequenza, ampiezza) sono noti ai computer che comunicano, tale da assicurare un segnale risultante molto forte rispetto alle possibili sorgenti di rumore. Al modem dell'elaboratore che invia i dati spetta il compito di trasformarli in segnali analogici e di sommarli all'onda portante, mentre il computer che riceve i dati sottrae all'onda in arrivo l'onda portante e traduce i segnali in dati.

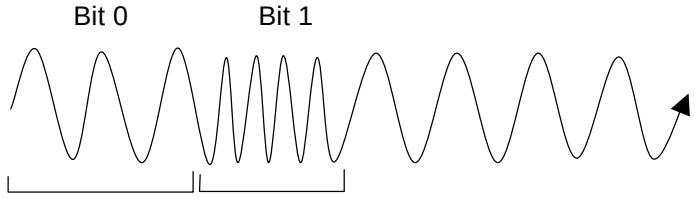
L'associazione di bit 0 e 1 ai segnali analogici del modem può avvenire in tre modi principali:

- *modulazione di ampiezza*
si fa corrispondere ai bit 0 un segnale di ampiezza minima, e ai bit 1 un segnale di ampiezza nettamente superiore.

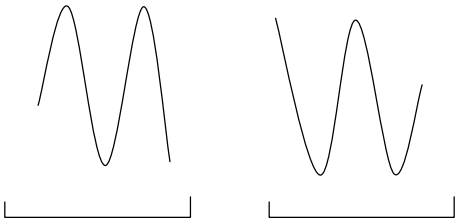


Questo è il tipo di modulazione più vulnerabile al rumore. L'ampiezza delle onde viene facilmente "camuffata" dal disturbo sul segnale

- *modulazione di frequenza*
si fa corrispondere ai bit 0 un segnale di frequenza minima, e ai bit 1 un segnale di frequenza sensibilmente superiore.



- *modulazione di fase*
si fa corrispondere ai bit 0 un segnale di fase zero, e ai bit 1 un segnale di fase massima.
Si chiama *fase* il valore dell'onda all'istante dell'emissione.



Questo è il tipo di modulazione meno vulnerabile alle fonti di rumore.

ABOUT THIS DOCUMENT…

L'ultima revisione di questo documento è stata effettuata il 19/2/2003. Esso non è soggetto a nessuna tutela di copyright ed è liberamente modificabile da chiunque, anzi è bene accetta qualunque integrazione e correzione. Lo scopo di questo documento è quello di dare una buona preparazione per l'esame di Architettura Degli Elaboratori II del prof. De Biase, ma non ha alcuna pretesa di completezza. Quindi, in sostanza, accetto complimenti ma non lamentele… ^___^

RINGRAZIAMENTI

- Il mio PIII 550Mhz equipaggiato con MSWord che ha permesso la scrittura di questo prontuario.
- Il prof. G. De Biase che mi ha fatto passare l'esame con 28
- Il prof. A. Sterbini… anzi, lui non lo ringrazio affatto!!!
- S. Fracassa, che con i suoi appunti mi ha dato un aiuto non da poco.
- Maddalena, perché è l'ispiratrice di tutti i miei sogni.

Sergio Di Mico
sedimico@tin.it