CAPITOLO 1

SISTEMI ELETTRONICI DIGITALI E CODIFICAZIONE DELLE INFORMAZIONI

1.1 INTRODUZIONE

In generale per sistema elettronico s'intende un insieme di strumentazione elettronica di tipo Analogica, Digitale e Ibrida (vale a dire in parte analogica ed in parte digitale). La strumentazione analogica è stata la prima ad essere disponibile nei laboratori di ricerca ed ha svolto un ruolo molto utile ed essenziale. Tuttavia la successiva strumentazione digitale ha, nell'ultimo decennio, preso il sopravvento. Essa è attualmente in largo uso e ha già prodotto una vera rivoluzione sul modo di fare misure ed analisi dati. La strumentazione digitale è qualitativamente e quantitativamente più veloce ed affidabile.
La differenza essenziale tra i due tipi di strumentazione sta nel fatto che il dato di ingresso della strumentazione analogica, (sotto forma di tensione o di corrente) può assumere valori nel continuo (in un dato intervallo di valori) e può essere funzione continua o meno del tempo, mentre il dato di ingresso della strumentazione digitale deve essere prima campionato nel tempo e poi discretizzato in valore.
Questa semplice distinzione sembrerebbe avvantaggiare la strumentazione analogica giacché accettando e processando segnali di tensione o corrente nel continuo, sia di valori sia di tempo, è intrinsecamente più veloce. La strumentazione digitale d'altra parte, pur accettando segnali campionati nel tempo e discretizzati in valore (ha limiti di velocità di processamento e non è capace di elaborare segnali che non siano in forma opportunamente discretizzata e codificata) ha basato la sua fortuna sulla riproducibilità del trasferimento ed elaborazione delle informazioni e sulla velocità di processamento di questo tipo di dati.
Poiché la strumentazione digitale processa soltanto elementi discreti d'informazione è necessario che il mondo delle informazioni analogiche sia trasformato, in modo opportuno, prima di poter essere accettato dai sistemi digitali.
Per risolvere questo problema è stata progettata e costruita un particolare tipo di strumentazione detta genericamente d'*interfaccia*. Questa campiona nel tempo i segnali analogici (ad una frequenza opportuna o quando richiesto da un evento esterno) e ne discretizza i valori. Questi ultimi o sono usati da stadi successivi dello stesso strumento (come accade per i voltmetri digitali che dopo qualche semplice elaborazione li presentano sul proprio display) oppure, dopo averli codificati in modo che possano essere acquisiti e processati da altra strumentazione digitale, li trasferiscono a quest'ultima. In questo caso si

dice che sono dotati d'interfaccia verso altri strumenti. I tipi d'interfaccia sono dei più disparati ma essenzialmente tutti risolvono un problema della sincronizzazione: la strumentazione d'interfaccia memorizza il dato da trasferire, ed informa (in modo opportuno) che ha un dato pronto. Il trasferimento avviene quando la strumentazione ricevente dichiara all'interfaccia che è pronta a ricevere il nuovo dato. In modo figurato possiamo dire che l'interfaccia è come un attore che parla se prima non ci sono spettatori ad ascoltarlo. La strumentazione d'interfaccia può a sua volta ricevere delle informazioni (dati o comandi) da strumentazione esterna, usando lo stesso metodo di scambio d'informazioni. In questo caso la strumentazione d'interfaccia si dice programmabile.

Quando la strumentazione d'interfaccia non ha possibilità d'elaborazione e presentazione dei dati realizza, essenzialmente, un campionamento ed una *conversione analogica/digitale* dei dati. Esempio di questo tipo di apparecchiatura è un sistema che codifica l'angolo di rotazione di un asse, la distanza tra due traguardi spaziali o temporali (cronometro) etc..

L'orologio è un esempio di strumentazione digitale che oltre a fare la conversione analogica (lo scorrere continuo del tempo) in digitale (il numero di unità di tempo trascorso) elabora queste informazioni fornendo sul suo quadrante ore, minuti e secondi.

Da quanto detto finora, sulla strumentazione digitale, emerge come essenziale il problema di conoscere come il numero, che rappresenta la discretizzazione dell'informazione analogica, è codificato, allo scopo di poter essere facilmente ed efficacemente processato e/o trasferito da uno strumento ad un altro minimizzando la possibilità di errore.

Il metodo scelto usa due soli livelli codificati con i simboli numerici 0 ed 1.

Questo tipo di codifica è quella che riduce al massimo l'errore di interpretazione. Questi possono essere rappresentati da due diversi livelli di tensione (0, 5Volt) o dai due stati di magnetizzazione di un materiale magnetico, da una macchiolina bianca o nera, da un buco presente o assente in certe posizioni su di una scheda, da una lampadina accesa o spenta etc. etc. Il motivo di tale scelta sta nel fatto che, dovendo distinguere elettronicamente solo tra due valori o stati diversi, il sistema che ne viene fuori risulta molto più affidabile; presenta cioè una maggiore riproducibilità dei risultati o, che è lo stesso, il sistema ha una probabilità minore di commettere degli errori. Il perché può essere immediatamente illustrato con un esempio: si abbia da decidere se un segmento ha una lunghezza maggiore o minore di 10 cm. L'operazione consiste nel misurare il segmento per es. con un doppio decimetro e la risposta sarà "si" se risulterà maggiore o "no" se risulterà minore (tipico caso codificabile con la corrispondenza si => 1 e no => 0). Notare che, poiché l'esatta lunghezza del segmento non è richiesta, errori grandi di misura non forniranno risposte errate, a meno che il segmento non è lungo proprio 10 cm. Se avessimo richiesto invece la lunghezza del segmento, con la precisione consentita dallo strumento di misura, il nostro confronto bisognava di molta maggiore cura ed il risultato di misure successive, fatte anche dallo stesso operatore con lo stesso strumento o meno, avrebbe potuto fornire un valore diverso. La misura così fatta ha

sostanzialmente minore affidabilità e riproducibilità. Quest'esempio dimostra perché sono stati scartati i metodi multi livello, anche se offrono molte similitudini con quello analogico. In un sistema multi livello, per es. se l'intervallo di valori è tra 0 e 9, sono definiti 10 livelli di una grandezza fisica qualsiasi (tensione, corrente, altezze..). Per una rappresentazione grafica si pensi ai numeri 0,1..9 disposti su di un asse orientato. Ad ognuno di essi è associato un segmento la cui lunghezza cresce linearmente col numero. Questo sistema ha il grave difetto di aver bisogno di distinguere e quindi discriminare tra i diversi livelli. Se questa quantità è trasferita da un'apparecchiatura all'altra la probabilità che sia alterata è alta.

Questa scelta ha determinato l'uso di un sistema numerico in base due. Notare che l'uomo ha avuto necessità, non solo dei numeri (per rappresentare quantità numeriche), ma anche dei caratteri (per rappresentare, per iscritto, quanto esprimeva a voce). In un qualsiasi alfabeto le parole sono rappresentate come una sequenza di caratteri e le frasi sono rappresentate da una sequenza di parole. La scelta di simboli diversi per la rappresentazione dei caratteri e dei numeri è stato dettata dalla facilità con cui l'uomo può scrivere e leggere entrambi e dal fatto che, in tal modo, i numeri sono immediatamente distinguibili dal testo. L'adozione di simboli diversi, tuttavia, non è concettualmente necessaria; infatti, è possibile rappresentare o codificare anche i caratteri con i numeri. Basta, per es., che sia definita una corrispondenza biunivoca tra numeri e le lettere di un alfabeto, e che il testo sia racchiuso tra due caratteri speciali: uno che indica l'inizio testo e l'altro che ne indica la fine. Poiché l'uomo riesce a scrivere e leggere con la stessa facilità sia i caratteri alfabetici sia i numeri, una rappresentazione, come quella ora detta, non ha avuto successo. Per le macchine, invece, poiché sono in grado di distinguere, con affidabilità, soltanto un tipo particolare di simboli, si è ricorso ad un sistema di codifica simile a quello sopra enunciato. In tal modo è possibile elaborare, oltre che i valori numerici, anche qualsiasi altra informazione discreta (come i caratteri, i testi scritti, elenchi di personale, etc., purché opportunamente codificate). Per un certo numero di esse, caratterizzate da un uso frequente e generale, sono stati definiti degli standard di codificazione adeguati al loro processamento e memorizzazione.

1.2. Numeri binari

L'usuale rappresentazione di quantità attraverso numeri è detta rappresentazione numerica in base 10. Le quantità sono rappresentate usando una sequenza di solo 10 simboli diversi o cifre: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 che assumono valore diverso secondo la loro posizione nella sequenza. Per es. il numero 1234 rappresenta una quantità uguale a 1 migliaio + 2 centinaia + 3 diecine + 4 unità. Le unità, diecine centinaia, etc., che sono potenze di 10, sono implicite nella posizione dei coefficienti. In altre parole 1234 dovrebbe essere scritto come $1\cdot10^3 + 2\cdot10^2 + 3\cdot10^1 + 4\cdot10^0$. Tuttavia la convenzione è di scrivere solo i coefficienti e dalla loro posizione dedurre la relativa potenza di 10. In generale un numero, con la virgola decimale, è rappresentato da una serie di coefficienti come segue:

$$a_4a_3a_2a_1a_0,a_{-1}a_{-2}a_{-3}$$

in cui i coefficienti a_i rappresentano una delle 10 cifre (0,..9), l'indice rappresenta la posizione del coefficiente e quindi il valore della potenza di 10 per cui il coefficiente deve essere moltiplicato.

In generale chiamato "b" la base numerica, cioè il numero di simboli diversi usati nella rappresentazione, un numero espresso in base "b" ha la forma

$$a_n\cdot b^n + a_{n-1}\cdot b^{n-1}+.....+ a_2\cdot b^2 + a_1\cdot b^1 + a_0 + a_{-1}\cdot b^{-1} + a_{-2}\cdot b^{-2} +.....+ a_{-m}\cdot b^{-m}$$

in cui i coefficienti a_i hanno uno dei valori che vanno da 0 a *b-1*. Il sistema numerico così definito si dice in base *b*.

I numeri binari sono numeri scritti utilizzando un sistema numerico in base 2. Cioè i numeri sono espressi con la convenzione su specificata ma con l'uso di solo due cifre diverse: 0 ed 1. Pertanto il numero in base due:

$(11010,11)_2$ significa: $1\cdot2^4 + 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 0\cdot2^0 + 1\cdot2^{-1} + 1\cdot2^{-2} =$

$$16 + 8 + 0 + 2 + 0 + 1/2 + 1/4 = 26 + 3/4 = 26,75$$

Sebbene la base numerica può essere qualsiasi, hanno trovato largo uso anche quelle in base 8 o *ottali* e in base 16 o *esadecimali*. Nel caso esadecimale oltre ai simboli 0,..9, sono stati usati le prime lettere dell'alfabeto con la seguente corrispondenza:

A =>10 B=>11 C=>12 D=>13 E=>14 F=>15.

La Tab. 1.1 fornisce i primi 16 numeri interi nelle basi: decimale, binario, ottale ed esadecimale. Come è ovvio per rappresentare la stessa quantità numerica si avrà bisogno di un numero di cifre tanto maggiore quanto più piccola è la base del sistema numerico. Pertanto, da questo punto di vista, i numeri espressi in base 2 contengono il numero più elevato di cifre. Ciò nonostante questo sistema è stato utilizzato, per la rappresentazione numerica della strumentazione digitale, a causa della maggiore affidabilità nella distinzione elettronica delle singole cifre.

Notare che, per i sistemi binario, ottale ed esadecimale, le operazioni di base (somma, sottrazione, prodotto e divisione) restano le stesse di quelle del sistema decimale. La

convenzione del sistema numerico è la stessa e le normali regole per l'esecuzione delle operazioni non dipendono dalla scelta della base.

Decimale (Base 10)	*Binario (Base 2)*	*Ottale (Base 8)*	*Esadecimale (Base 16)*
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Tab. 1.1 rappresentazione numerica in diverse basi

Per es. se, dalla somma di due cifre in base *b*, si ottenesse una quantità che supera (*b-1*) il risultato si scrive con due cifre. Nella somma di due numeri in base *b* si procede a fare la somma, a partire dalla coppia di cifre meno significative, e se questa supera *b -1* si scrive, come risultato, la cifra meno significativa e quella più significativa si somma alla coppia di cifre immediatamente più significative degli addendi.

Per es. La somma dei due numeri binari 0110 e 0011 è uguale a 1001 (vedi Tab. 1.2) infatti (mimando l'usuale operazione di somma diciamo): la cifra meno significativa della somma si trova sommando la coppia di cifre (nella stessa posizione) degli addendi 0 + 1 = 1 e poiché non supera 1 scrivo 1 con il riporto di 0, la seconda cifra si trova sommando il riporto alla somma della coppia di cifre immediatamente a sinistra: 0 + (1 + 1) = 10, scrivo 0 col riporto di 1, la terza cifra si trova, analogamente, sommando 1 + (1 + 0) = 10 per cui scrivo 0 col riporto di 1 e sia la quarta sia l'eventuale quinta cifra si trova sommando 1+ (0 + 0) = 1 scrivo 1 col riporto di 0. Esempi ulteriori possono essere facilmente trovati dal lettore per le altre operazioni di sottrazione, prodotto e divisione.

Riporto	010
I° Add	0110
II° Add	0011
Risultato	1001

Tab 1.2

1.2.1 Conversione di Base Numerica

Delle volte serve trasformare un numero da una data base in un'altra. Noi sappiamo fare già la trasformazione di un numero da una base qualsiasi in decimale; basta applicare la definizione con cui un numero è scritto esprimendo il risultato in decimale come fatto in precedenza.

Qualche complicazione sorge quando bisogna trasformare un numero da decimale in binario, ottale o esadecimale. Si hanno metodi diversi per la conversione della parte intera e della parte frazionaria. Fatta separatamente, tale conversione, quella del numero in totale è ottenuta dalla somma delle conversioni parziali.

a) Per **trasformare la parte intera di un numero decimale:**

Bisogna dividere il numero per la base b e trovare il quoziente intero Q_0 ed il resto a_0. La cifra meno significativa della conversione è data da a_0. Per trovare le cifre più significative, si opera allo stesso modo sul quoziente,

cioè si divide Q_0 per *b* e si trova il nuovo quoziente Q_1 ed il nuovo resto a_1 che è la cifra immediatamente più significativa. Il processo ha termine quando si ottiene per quoziente $Q_n = 0$ e l'ultimo resto trovato, a_n, è la cifra più significativa della conversione del numero intero.

Consideriamo un esempio: sia N = 123 e lo si voglia trasformare in base 8. Nel seguito con *int(a)* intendiamo la parte intera di *a*.

$Q_0 = int(123/8) = 15$ $a_0 = 3$

$Q_1 = int(15/8) = 1$ $a_1 = 7$

$Q_2 = int(1/8) = 0$ $a_2 = 1$

123	8	
3	15	8
	7	1	8
		1	0

Tab 1.3

Allora N = $(173)_8$. Per provare la correttezza del risultato facciamo la conversione da ottale a decimale cioè

$$N = 1\cdot 8^2 + 7\cdot 8 + 3 = 64 + 56 + 3 = 123.$$

Come altro esempio consideriamo la conversione dello stesso numero in base 2.

$Q_0 = int(123/2) = 61$ $a_0 = 1$

$Q_1 = int(61/2) = 30$ $a_1 = 1$

$Q_2 = int(30/2) = 15$ $a_2 = 0$

$Q_3 = int(15/2) = 7$ $a_3 = 1$

$Q_4 = int(7/2) = 3$ $a_4 = 1$

$Q_5 = int(3/2) = 1$ $a_5 = 1$

$Q_6 = int(1/2) = 0$ $a_6 = 1$

123	2	
1	61	2
	1	30	2
		0	15	2
			1	7	2
				1	3	2
					1	1	2
						1	0

Tab 1.4

Allora N = $(1111011)_2$. Per provare la correttezza del risultato facciamo la conversione da binario a decimale cioè:

$$N = 1\cdot 2^6 + 1\cdot 2^5 + 1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 2 + 1 = 64 + 32 + 16 + 8 + 2 + 1 = 123.$$

Paragoniamo adesso i due risultati:

$$N = (1\ 111\ 011)_2 = (173)_8 = (001\ 111\ 011)_2.$$

L'ultima rappresentazione di N è stata ottenuta dalla conversione di ciascuna cifra ottale in binario con l'ausilio della Tab. 1.1. Dal paragone delle due rappresentazioni in base 2 notiamo che esse differiscono per la presenza di due zeri in posizione più significativa.

Poiché questi non alterano il valore numerico concludiamo che le due rappresentazioni sono equivalenti e che generalizzando opportunamente otteniamo:

1) la conversione in binario della parte intera di un numero ottale si ottiene convertendo le singole cifre da ottale in binario.
2) la conversione in ottale della parte intera di un numero binario si ottiene raggruppando le cifre binarie a tre a tre, ad iniziare da quella meno significativa e convertendo ciascuna terna nella corrispondente cifra ottale.

Il risultato, ora descritto, è un caso particolare della conversione di un numero intero, scritto in binario, in un numero intero scritto in una base numerica esprimibile con una potenza di due (b = 2, 4, 8, 16,....) e viceversa. Infatti, in generale possiamo dare la seguente regola:

1a) *La conversione in binario della parte intera di un numero in base $b = 2^n$ si ottiene convertendo le singole cifre da quella base in binario.*

2a) *La conversione in base* b = 2^n *della parte intera di un numero binario si ottiene raggruppando le cifre binarie a n a n, ad iniziare da quella* ***meno*** *significativa, e convertendo ciascuna ennupla nella corrispondente cifra nella base numerica di arrivo.*

La conversione della parte intera di un numero da una base *b1* = 2^n ad un'altra in base *b2* = 2^m può essere fatta con un procedimento a due passi:

I) Si converte il numero dalla base *b1* (*b2*) in binario,

II) Si converte il numero binario ottenuto in base *b2* (*b1*).

Come esempio di questa regola convertiamo il numero N = 123 = $(1111011)_2$ in esadecimale. Poiché, in base esadecimale, la potenza del 2 è 4 bisogna raggruppare le cifre a quattro a quattro ad iniziare da quella meno significativa e farne la conversione in esadecimale

$$N = (1111011)_2 = (0111\ 1011)_2 = (7B)_{16}.$$

Infatti $(7B)_{16}.= 7\cdot16 + 11 = 112 + 11 = 123$.

b) Per **trasformare la parte frazionaria di un numero decimale, F:**

Bisogna moltiplicare il numero per la base b; la parte intera, a_{-1}, del prodotto $F\cdot b$ fornisce la cifra decimale più significativa della parte frazionaria. Si sottrae la parte intera da $F\cdot b$ ricavando un nuovo numero decimale: $F_1 = F\cdot b - a_{-1}$. Questo è processato allo stesso modo per trovare tutte le altre cifre meno significative.

Il processo ha termine quando si ottiene una parte frazionaria nulla o si decide di troncare il processo poiché ci si accontenta della approssimazione trovata.

Consideriamo un esempio: sia F = 0,6875 e lo si voglia trasformare in base 8. Nel seguito con *frac(a)* intendiamo: parte frazionaria di *a*.

0,6875				
x8=				
5,5000	-5=	0,5x		
		8=		
		4,0	-4=	0

Tab 1.5

$F_1 = frac(0,6875 \times 8) = 0,5 \quad a_{-1} = 5$

$F_2 = frac(0,5 \times 8) = 0,0 \quad a_{-2} = 4$

Allora F = $(0,54)_8$. Per provare la correttezza del risultato facciamo la conversione da ottale a decimale cioè $F = 5 \cdot 8^{-1} + 4 \cdot 8^{-2} = (40 + 4)/64 = 44/64 = 0.6875$.

Come altro esempio consideriamo la conversione dello stesso numero in base 2.

$F_1 = frac(0,6875 \times 2) = 0,375 \quad a_{-1} = 1$

$F_2 = frac(0,375 \times 2) = 0,75 \quad a_{-2} = 0$

$F_3 = frac(0,75 \times 2) = 0,5 \quad a_{-3} = 1$

$F_4 = frac(0,5 \times 2) = 0,0 \quad a_{-3} = 1$

Allora F = $(0,1011)_2$. Per provare la correttezza del risultato facciamo la conversione da binario a decimale cioè

0,6875x							
2=							
1,3750	-1=	0,375x					
		2=					
		0,750	-0=	0,75x			
				2=			
				1,50	-1=	0,5x	
						2=	
						1,0	-1= 0

Tab 1.6

$$F = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} = (8 + 2 + 1)/16 = 11/16 = 0,6875.$$

Paragoniamo adesso i due risultati:

$$N = (0,101\ 1)_2 = (0,54)_8 = (0,101\ 100)_2.$$

L'ultima rappresentazione di N è stata ottenuta dalla conversione di ciascuna cifra ottale in binario con l'ausilio della Tab. 1.1. Dal paragone delle due rappresentazioni in base 2 notiamo che esse differiscono per la presenza di due zeri in posizione meno significativa. Poiché questi non alterano il valore numerico concludiamo che le due rappresentazioni sono equivalenti e che generalizzando opportunamente otteniamo:

3) *La conversione in binario della parte frazionaria di un numero ottale si ottiene convertendo le singole cifre da ottale in binario.*

4) La conversione in ottale della parte frazionaria di un numero binario si ottiene raggruppando le cifre binarie a tre a tre, ad iniziare da quella più significativa e convertendo ciascuna terna nella corrispondente cifra ottale

Il risultato, ora descritto, è un caso particolare della conversione di un numero intero, scritto in binario, in un numero intero scritto in una base numerica esprimibile con una potenza di due (*b* = 2, 4, 8, 16,....) e viceversa. Infatti, in generale possiamo dare la seguente regola:

3a) *La conversione in binario della parte frazionaria di un numero in base* b = 2^n *si ottiene convertendo le singole cifre da quella base in binario.*

4a) *La conversione in base* b = 2^n *della parte frazionaria di un numero binario si ottiene raggruppando le cifre binarie a n a n, ad iniziare da quella* ***più*** *significativa, e convertendo ciascuna ennupla nella corrispondente cifra nella base numerica di arrivo*

La conversione della parte frazionaria di un numero da una base *b1* = 2^n ad un'altra in base *b2* = 2^m può essere fatta con un procedimento a due passi:

I) Si converte il numero dalla base *b1* (*b2*) in binario,

II) Si converte il numero binario ottenuto in base *b2* (*b1*).

Come esempio di questa osservazione convertiamo il numero F = 0,6875 in esadecimale. Poiché in base esadecimale la potenza di 2 è 4, bisogna raggruppare le cifre a quattro a quattro ad iniziare da quella più significativa e farne la conversione in esadecimale

$$F = (0,1011)_2 = (0,B)_{16}.$$

Infatti $(0,B)_{16} = 11 \cdot 16^{-1} = 11/16 = 0.6875$.

c) Per **trasformare un numero decimale che ha una parte intera ed una parte frazionaria** *bisogna convertire separatamente la parte intera e la parte frazionaria e quindi sommare il risultato.* (per es. *N =123,6875)* Per l'esempio su trattato si ha:

$$N = 123,6875 = (173,54)_8 .= (1111011,1011)_2 .= (7B,B)_{16}$$

1.3 Codici Binari

Nei paragrafi 1.2.1 e 1.2.2 abbiamo visto un esempio di come sia possibile, mediante un'opportuna regola di codifica, rappresentare quantità maggiori del numero di simboli scelti. In particolare abbiamo evidenziato che anche con l'uso di cifre binarie è possibile rappresentare qualsiasi quantità numerica. Ovviamente al cambiare delle regola di codifica è possibile rappresentare quantità numeriche in modo diverso o rappresentare quantità non numeriche come per es. le lettere dell'alfabeto.

In questo paragrafo saranno introdotti alcuni codici binari largamente usati. Ciò anche allo scopo di fornire esempi di campi di applicazioni di macchine il cui funzionamento è basato sulla codifica binaria.

Si noti, incidentalmente, che uno sperimentatore, col compito di mettere assieme un sistema di acquisizione e di analisi dati automatica, ha di fronte a se il problema di rendere "compatibili" i segnali elettronici che provengono *da* e arrivano *ai* diversi tipi di strumentazione che intende utilizzare. Sebbene, come detto, la strumentazione elettronica usi codici binari non è garantito che lo stesso tipo di codice sia usato dalle diverse apparecchiature, per cui, sia nella scelta delle apparecchiature da usare che, nel loro collegamento è necessaria la conoscenza sia dei codici binari usati sia delle modalità di comunicazione tra le varie apparecchiature.

Poiché non è possibile presentare, per il loro numero, in modo completo tutti i codici conosciuti, noi tratteremo il problema della codifica delle informazioni da un punto di vista generale. Lo studente, capito sia il metodo sia lo scopo può, da solo, studiare ed apprendere codici che non saranno qui trattati.

Codificare un'informazione, nel nostro contesto, significa definire una corrispondenza biunivoca tra ogni uno dei modi, differenti e finiti, in cui l'informazione stessa si presenta ed una particolare configurazione o combinazione con ripetizione di *2* simboli a *n* a *n*.
Poiché noi abbiamo scelto di utilizzare solo due simboli diversi 0 ed 1, una sequenza di $n = 1$ simboli può codificare e quindi distinguere due sole informazioni diverse: vero o falso, si o no, bianco o nero, acceso o spento, giorno o notte, papà o mamma, etc.. L'elencazione potrebbe continuare ancora per molto. Noi non lo facciamo perché invitiamo lo studente a continuarla ancora un poco e a considerare che sebbene possiamo distinguere solo due entità (normalmente rappresentanti situazione opposte) i campi di applicazione sono i più svariati.
La cifra binaria 0 od 1 è chiamata un **bit** di informazione dall'inglese *Binary Digit*.
Se usiamo una sequenza di n = 2 simboli o che è lo stesso usiamo due bit possiamo codificare 4 situazioni diverse: 00, 01, 10, 11. Ogni una di queste combinazioni potrebbe essere scelta per rappresentare una su 4 situazioni diverse invece delle due viste precedentemente. Per es. in una descrizione simbolica di una situazione familiare, la combinazione 00 potrebbe rappresentare il padre, 01 la figlia femmina, 10 il figlio maschio e 11 la mamma. Anche ora potremmo sbizzarrirci a trovare una miriade di situazioni diverse a cui si può applicare un codice a 2 bit e lo lasciamo per esercizio al nostro lettore. Noi ci limitiamo ad osservare da una parte che, pur restando vasto il campo di applicazione, le situazioni che si possono codificare sono più complicate delle precedenti e dall'altra che l'uso di 2 bit invece che uno non ci vieta di descrivere le situazioni descrivibili con un solo bit; basta scegliere solo una coppia di combinazione e considerare le altre non utilizzate o sovrabbondanti. C'è di più: l'altra coppia di valori ridondante potrebbe risultare utile per es. per la rivelazione di errore di trasmissione ed eventualmente per la sua correzione. Come esempio supponiamo di voler trasmettere ad un nostro compagno di giochi, che si trova nella casa di fronte, che a casa propria c'è papà o mamma. Il sistema di trasmissione consiste in una lampadina tascabile che se accesa per un tempo breve codifica lo 0 mentre se accesa per un tempo più lungo codifica l'1. Poiché la durata dell'accensione è determinata senza l'aiuto di strumenti la durata dell'1 e dello 0 non risulterà esattamente quella concordata e pertanto c'è la possibilità che il compagno non capisca correttamente (deve giudicare dalla durata di un solo lampo di luce). Se invece ci mettiamo d'accordo che la presenza della mamma è codificata con un flash breve seguito da uno lungo e, per la presenza del papa con un flash lungo seguito da uno breve, si ha il vantaggio di paragonare la durata dei due flash tra di loro e quindi poter decidere quale dei due è stato più lungo. D'altra parte se entrambe le durate sono stimate uguali è chiaro che non è possibile dire di quale delle due situazioni il compagno ha voluto informare (non gli corrisponde alcuna informazione); *anche se non si viene in possesso dell'informazione non si rischia di averne una sbagliata*.
Il discorso potrebbe ora proseguire utilizzando 3 bit e poi 4 e così via. Noi non proseguiremo su questa strada ma utilizzeremo un approccio complementare: *supponiamo di avere un*

problema di codifica di informazioni e ricerchiamo sia il metodo sia il numero di bit necessari.

Premettiamo che se si decide di non usare ridondanza, il numero di situazioni diverse che possono essere distinte con un codice che usa *n* bit è dato dal numero di combinazione con ripetizione di 2 oggetti a *n* a *n* e cioè 2^n .

Supponiamo di volere codificare le lettere dell'alfabeto italiano e i 9 simboli numerici da 0 a 9. Poiché nel nostro alfabeto ci sono 21 lettere necessitiamo di codificare 31 situazioni diverse. Per trovare il numero di bit minimo dobbiamo risolvere la disequazione $2^n \geq 31$ il cui risultato è $n \geq 5$. Quindi una codifica con 5 bit risolve il nostro problema e ci lascia una sola combinazione ridondante. Ciò può essere considerato soddisfacente e pertanto si può proseguire alla stesura della codifica. Notare che non esiste alcuna condizione del problema che ci impone di fare una scelta invece che un'altra. Siamo completamente liberi di fare la nostra assegnazione; per es. a caso tra una qualsiasi sequenza di 5 bit e una qualsiasi lettera o simbolo numerico purché questi ne restino univocamente determinati. In realtà generalmente ci si dota di criteri di assegnazione (se non altro in vista di una più rapida ricerca della corrispondenza ai fini del riconoscimento). Nel nostro caso si potrebbe scegliere il seguente criterio: si scrivono in binario i numeri da 0 a 20 e si associa ad ognuno una lettera dell'alfabeto nell'ordine: *a, b, c*.... ed ai numeri binari 21, 22, ..., 30 si associa le cifre 0, 1, ..., 9.

I 5 bit trovati precedentemente potrebbero essere non sufficienti se volessimo per es. distinguere le lettere maiuscole da quelle minuscole; adesso il problema richiede di distinguere 31 + 21 = 52 casi diversi e la disequazione $2^n \geq 52$ fornisce $n \geq 6$. Poiché $2^6 = 64$ avremo 12 combinazioni ridondanti.

Dal semplice esempio appena illustrato risulta chiaro che si possono inventare le corrispondenze più disparate per codificare lo stesso problema. In tale situazione la compatibilità tra le diverse apparecchiature sarebbe pressoché nulla. Per evitare di produrre una situazione da Babele esistono delle commissioni internazionali che accettano, elaborano e propongono, per problemi di carattere generale, degli standard. I costruttori di apparecchiature, normalmente, vi si riferiscono e quindi producono strumentazione compatibile con quella di altri costruttori. Per sfortuna esistono diverse organizzazioni internazionali perciò per lo stesso problema sono talvolta proposti standard diversi. Uno standard è caratterizzato normalmente da condizioni più numerose di quanto traspaia dal nostro semplice ragionamento; più in là, in questo lavoro, avremo occasione di parlarne.

1.3.1 Cifre Decimali Codificate in Binario

L'utilità di codificare in binario le singole cifre decimali con cui viene scritto un numero in base 10 è molteplice:

- l'utente umano preferisce assegnare e ricevere risultati decimali (codificazioni dei dati in ingresso ed in uscita)
- esistono macchine che, anche all'interno continuano a lavorare su numeri decimali e necessitano però che ogni cifra sia codificata in binario.

Questo tipo di scelta è dettata dalla facilità sia di conversione in ingresso e in uscita che dalla necessità di mantenere arrotondamenti e troncamenti, durante le operazioni matematiche, secondo le regole dell'aritmetica decimale.

I codici binari per cifre decimali richiedono un minimo di 4 bit che determinano una ridondanza di 6 combinazione diverse. Come già visto, sia per la libertà di scelta che per la presenza della ridondanza, è possibile fare le scelte più disparate. Nella successiva Tab. 1.7 ne sono presentate alcune.

Il codice BCD (Binary Coded Decimal) è l'assegnazione diretta dell'equivalente binario. É possibile assegnare, come regola per costruire la codifica, dei pesi ai bit binari in accordo alla loro posizione.

Cifre Decimali	(BCD) 8421	Eccesso-3	84-2-1	2421	Bisquinario 5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

Tab. 1.7 Codici binari di cifre decimali

I pesi nel codice BCD sono 8, 4, 2, 1. Per es. l'assegnazione di bit 0110, può essere interpretata, usando i pesi ora detti, come 6, infatti, è 0 x 8 + 1 x 4 + 1 x 2 + 0 x 1 = 6. Tutti gli altri codici ad eccezione dell'Eccesso di 3 sono codici pesati, i pesi essendo anche gli identificativi di colonna. Nel caso del codice con intestazione di colonna 8 4 -2 -1 la sequenza 0110 rappresenta: 0 x 8 + 1 x 4 + 1 x (-2) + 0 x (-1) = 2.

Il codice Eccesso di 3 è non pesato ed è stato usato largamente nei vecchi computer. Esso si ricava sommando, in binario, al codice BCD 3 vale a dire 0011. La sua particolarità risiede nel fatto che complementando a bit a bit (0 diventa 1 e 1 diventa 0) il numero, si ottiene un numero decimale, ancora in codice eccesso di 3, che è il complemento a 9 del numero di

partenza. Per es. 1011 che corrisponde a 8 diventa, dopo la complementazione a bit a bit, 0100 che corrisponde a 1 vale a dire il complemento a 9 di 8 (9 - 8 = 1)
Anche il codice pesato 2421 ha la stessa caratteristica del codice ad eccesso di 3 e vale a dire il suo complemento a bit a bit fornisce un numero decimale, ancora in codice 2421, che è il complemento a 9 del numero di partenza. Per es. 0100 che corrisponde a 4 diventa, dopo la complementazione a bit a bit, 1011 che corrisponde a 5 cioè il complemento a 9 di 4 (9 - 4 = 5)
Il codice Bisquinario è un esempio di un codice a 7 bit che usa le ridondanze per la rivelazione d'errore. Ciascuna cifra decimale consiste di **due** 1 e di **cinque** 0, posti in posizione con peso opportuno. La proprietà di rivelazione d'errore di questo codice può essere capito ricordando che un sistema digitale presenta la cifra zero con un segnale distinto dalla cifra 1. Durante la trasmissione del segnale, da un posto all'altro, può avvenire un errore. Uno o più bit possono essere rivelati con un valore diverso. Un circuito ricevente che rivela la presenza di più (o di meno) di due 1 ha ricevuto una combinazione di bit che non è in accordo con una delle combinazioni ammesse e così rivela l'errore.

1.3.2 Codici con rivelazione d'errore

Le informazioni binarie possono essere trasmesse in vario modo. N'abbiamo visto uno di tipo ottico (flash di luce di diversa durata nel paragrafo 1.3.1). Questo è un esempio di *trasmissione ottica seriale con ritorno a zero e modulazione di durata*. Infatti:

- E' una trasmissione di tipo *ottico* giacché si fa uso di un mezzo ottico: la lampada
- E' una trasmissione *seriale* giacché i bit interessati sono trasmessi uno per volta (uno di seguito all'altro),
- è a *ritorno a zero*. La possibilità di distinguere quando la lampada è spenta da quando è accesa realizza un sistema binario (per es. lampada spenta = 0, lampada accesa =1). Poiché la trasmissione di un bit d'informazione avviene durante lo stato "lampada accesa" (stato 1), per distinguere un bit d'informazione dall'altro la lampada deve ritornare ad essere spenta (stato 0).
- E' a modulazione giacché il bit d'informazione modula la durata dello stato "lampada accesa"

Il numero di modi ed i supporti usati come mezzo di trasmissione sono molteplici e la trattazione completa di tale argomento potrebbe essere essa stessa oggetto di un intero corso. Sono coinvolti argomenti che vanno dalla teoria dell'informazione alla statistica, allo studio dei materiali ed altro ancora. Noi ci limitiamo ad affermare che, normalmente, le informazioni binarie sono codificate attraverso due stati diversi di tensione o di corrente, che come mezzo di trasmissione è usato del filo od onde radio e che la codifica dell'informazione è

generalmente ad impulsi. Qualsiasi rumore esterno introdotto nel sistema di comunicazione fisico può cambiare il valore di un bit da 0 ad 1 o viceversa. Per la rivelazione di tali errori è allora necessario l'uso di un codice che ne dia la possibilità. In generale non si richiede che ci sia anche la possibilità di correzione dell'errore. I codici capaci di correggere uno o più bit d'errore hanno tutti una naturale inefficienza per la presenza di un'elevata ridondanza.
Le strategie usate per la trasmissione d'informazioni sono diverse. In generale dipendono dal tipo di comunicazione che è possibile tra i due posti remoti. Com'esempio ricordiamo quelli che usano l'*handshake*. Questo è usato nei sistemi di comunicazione che hanno capacità di rice/trasmissione, hanno cioè la possibilità di ricevere informazione (sullo stesso canale o su un altro parallelo), oltre che di trasmetterla. Con questo modo di trasmissione, alla stazione trasmittente, il messaggio completo è spezzato in più parti o pacchetti. Trasmesso il primo pacchetto, avanti dell'invio del secondo, si attende dalla stazione ricevente una risposta sulla qualità della ricezione. Se questa è positiva (non è stato rivelato alcun errore) si trasmette il pacchetto successivo altrimenti si rinvia lo stesso pacchetto. Tale modalità è ripetuta per ogni pacchetto. La stazione ricevente scarterà i pacchetti che ha giudicato non buoni e, per la ricostruzione dell'intero messaggio, metterà assieme tutti i pacchetti buoni.
In altri sistemi in cui non è possibile questo scambio d'informazioni si usano altre strategie. Una di queste consiste nel ripetere l'invio della stessa informazione un numero fissato di volte. Alla stazione ricevente le informazioni, che sono state ricevute senza errori, sono paragonate tra loro e se più di un certo numero di esse sono identiche l'informazione da essa rappresentata è ritenuta corretta ed accettata.

In generale bisogna adottare le strategie più ricercate nella ricezione di telecomandi; la loro errata attuazione, in alcuni casi, potrebbe essere disastroso.

Nella maggior parte dei casi, in cui un'informazione errata non provoca grossi problemi, alla stazione ricevente è fatto il conteggio del numero di errori rivelati nell'unità di tempo e se questo è giudicato piccolo non è fatto alcunché se invece è grande bisogna correre ai ripari; generalmente attraverso altri canali di comunicazione si provvede ad avvertire che la qualità è pessima e se è possibile la trasmissione è interrotta e ripetuta dopo che le condizioni che provocano gli errori sono stati rimossi.

Un bit di *parità* è un bit che si aggiunge al puro messaggio in modo da rendere il numero totale di 1, presenti nell'intero messaggio, *pari* o *dispari*. Nel primo caso si affermerà che la parità è **pari**, nel secondo che la parità è **dispari**.

Parità dispari		Parità pari	
P	cifra	P	cifra
1	0000	0	0000
0	0001	1	0001
0	0010	1	0010
1	0011	0	0011
0	0100	1	0100
1	0101	0	0101
1	0110	0	0110
0	0111	1	0111
0	1000	1	1000
1	1001	0	1001

Tab. 1.8 Codice BCD con Parità

In Tab. 1.8 è rappresentato il caso in cui si voglia trasmettere delle informazioni che consistono di cifre decimali codificate BCD. Ai 4 bit che definiscono una cifra si aggiunge un quinto bit che è la

parità. Questo quinto bit potrebbe essere messo in testa o in coda al messaggio, ma per vari motivi è opportuno metterlo in coda (come ultimo bit trasmesso).
Durante il trasferimento dell'informazione da un luogo ad un altro il bit di parità è usato come segue: all'estremità trasmittente viene calcolato ed aggiunto il bit di parità. Il messaggio, incluso il bit di parità (nel nostro caso 5 bit) viene trasmesso. Nell'unità ricevente viene ricevuto l'intero messaggio e sui bit del messaggio, escluso il bit di parità (nel nostro caso 4 bit) viene generato il bit di parità. Questo viene paragonato con quello ricevuto. Se i due bit non sono uguali viene rivelato un errore. Un altro metodo è di calcolare la parità dell'intero messaggio (5 bit) e rivelare un errore se non è uguale a quello convenuto. Il metodo del bit di parità è capace di rivelare la presenza di un numero dispari di errori in qualsiasi posizione. *Un numero pari di errori è non rilevabile*.

1.3.3 Codici Riflessi: Codice Gray

Molti sistemi fisici forniscono dati in uscita in modo continuo. Questi dati debbono essere convertiti in forma digitale o discreta prima di poter essere applicati a sistemi digitali. Le informazioni continue o analogiche vengono convertite in forma digitale per mezzo di convertitori Analogico/Digitali. L'informazione così convertita, sarà conforme ad un particolare codice che nel caso di flusso continuo di informazione si è trovato conveniente essere quello riflesso che ora presenteremo.

Il vantaggio del codice riflesso rispetto agli altri codici binari è che un numero, in codice riflesso, differisce solo in un bit nel passaggio da questo al successivo.

Il codice riflesso mostrato in Tab. 1.4 è solo uno dei codici riflessi possibili e prende il nome di codice Gray.

La regola che permette di generare la codificazione Gray di Tab. 1.4 è la seguente:

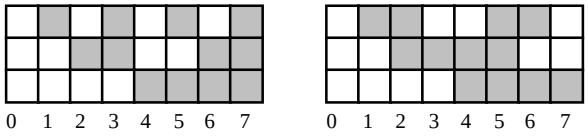
1) Per codificare i numeri da 0 ad 1 ci servono un numero $n = 1$ di bit e le due configurazioni sono: 0, 1.
2) Per la codifica di tutti gli altri numeri diamo una regola iterativa. Per codificare i numeri da 0 a $2^n - 1$ servono n bit. Le 2^n configurazioni diverse si ottengono "riflettendo" il codice già scritto per $n - 1$ bit in un immaginario specchio (disposto come indicato in Tab. 1.4) e premettendo 0 alle configurazioni originarie ed 1 a quelle riflesse.

Decimale	Gray
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

Tab. 1.9 Codice Gray

Un'applicazione tipica del codice riflesso è la rivelazione della posizione di un indice in continua variazione su di una scala. La scala è suddivisa in segmenti. A ciascun segmento è assegnato

un numero che l'indice legge. Nella Fig. 1.1 sono disegnati due rettangoli che rappresentano la nostra scala. Nella parte (a) il codice usato è quello binario puro, mentre nella parte (b) viene usato il codice Gray. L'indice passando sulla scala legge con un sistema ottico e fa corrispondere ad un quadratino letto come bianco 0 ed a quello letto come scuro 1. Quando l'indice attraversa la linea di separazione tra 1 e 2:



(a) Fig. 1.1 (b)

- nella parte (a), a causa di imperfezioni di realizzazione, invece del passaggio da 001 a 010 potremmo avere il passaggio a 000 o a 011 cioè una lettura sbagliata.
- nella parte (b) avremmo, in ogni caso il passaggio da 001 a 011. A causa di imperfezioni di realizzazione, potremmo avere che la lettura del numero successivo venga anticipato o posticipato ma mai un errore di lettura.

1.3.4 Codici Alfanumerici

Molte applicazioni richiedono di trattare dati che consistono non solo di numeri ma anche di lettere. Per es. la comunicazione dei dati ad un computer avviene, di norma, attraverso una tastiera. In questa, molto simile a quella delle vecchie macchine per scrivere, sono presenti sia tasti corrispondenti a lettere sia a numeri. Perché il computer capisca quale lettera o quale numero è stato digitato bisogna che ogni tasto e/o combinazione di tasti abbia una ben precisa codifica. Un altro esempio è la trasmissione digitale di testi. Ad ogni carattere alfabetico, numerico, segno di interpunzione o qualsiasi altro, bisogna che gli corrisponda un ben determinato codice. Il problema della compatibilità in questo campo si è rivelato così importante che il codice alfanumerico è stato uno dei primi ad essere standardizzato. Il più diffuso al mondo è il così detto ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) che è riportato in Tab 1.10. Notiamo che il numero minimo di bit proviene dalla disuguaglianza $64 < 2^n < 128$. Infatti, il numero di combinazioni diverse da distinguere: le lettere dell'alfabeto inglese (sia minuscolo sia maiuscolo) 26 x 2, più i 10 caratteri numerici + i segni di interpunzione (spazio, punto, virgola, punto e virgola, due punti, etc. etc.) è certamente > di 64 e < 128. Per ciò il numero minimo di bit è 7.

Approfittando della ridondanza alcune delle combinazioni sono state usate per codificare delle informazioni utili al controllo del sistema ricevente ed alla formattazione dei testi inviati digitalmente o come si dice oggi per posta elettronica e che sono rappresentati in tabella. Si noti che poiché questo codice veniva usato anche per comunicazione tra unità diverse e distanti è stato trovato utile fornire una possibilità di rivelazione di errore ed è stato aggiunto un bit di parità (non indicato in tabella in quanto questa può essere sia pari sia dispari). Pertanto il codice ASCII si dice che è un codice ad 8 bit o, che è lo stesso, per rappresentare un carattere alfanumerico sono necessari 8 bit.

	Colonne	0	1	2	3	4	5	6	7
Righe	Bits 765=> 4321|| \/	000	001	010	011	100	101	110	111
0	0000	NUL	DLE	SP	0	@	P	\	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
10	1010	LF	SUB	*	:	J	Z	j	z
11	1011	VT	ESC	+	;	K	[	k	{
12	1100	FF	FS	,	<	L	\	l	|
13	1101	CR	GS	-	=	M	]	m	}
14	1110	SO	RS	.	>	N	101	n	
15	1111	SI	US	/	?	O	-	o	DEL

Tab. 1.10 ASCII (American Standard Code for Information Interchange)

Esempio Codice per A = $(100\ 0001)_2 = (4\ 1)_{16} = (65)_{10}$

Legenda per i codici di controllo

Colonna 0

NUL= Null	SOH= Strat of Heading	STX= Start of text	ETX= End of Text
EOT= End of Transmis.	ENQ= Enquire	ACK= Acknowledge	BELL= Bell (audible signal)
BS= Backspace	HT= Horzontal Tab	LF= Line Feed	VT= Vertical Tab.
FF= Form Feeed	CR= Carriage Return	SO= Shift Out	SI= Shift In

Colonna 1

DLE= Dat Link Escape	DC1= Device Control 1	DC2= Device Control 2	DC3= Device Control 3
DC4= Device Control 4	NAK= Negative Acknowl.	SYN= Syncronous Idle	ETB= End Transmis. Block
CAN= Cancel	EM= End of Medium	SUB= Substitute	ESC= Escape
FS= File Separator	GS= Group Separator	RS= Record Separator	US= Unit Separator

Colonna 2 SP = space Colonna 7 DEL = delete

1.3.5 Flip-Flop, Registri Memorie Binarie e loro organizzazione

Gli elementi discreti di informazione siano essi sotto forma di singoli bit, numeri binari, codici binari o una qualsiasi loro sequenza, devono poter essere scritti e/o letti in qualche modo e da qualche parte. Chiamiamo memoria il posto in cui uno o più elementi discreti di informazione possono essere scritti e/o letti.

Un bit può essere memorizzato in qualsiasi posto in cui è possibile distinguere due stati diversi: una macchiolina nera o bianca su di un foglio di carta, presenza o assenza di un buco in un certa posizione su di una striscia di carta o qualsiasi altro supporto, la presenza di un tipo di magnetizzazione o un'altra su di un pezzettino di materiale magnetico o in una

particolare posizione su di un nastro magnetico etc. etc. Secondo il tipo di memoria scelta è necessario usare un metodo opportuno sia per la scrittura sia per la cancellazione (se si vuole usare più di una volta lo stesso posto di memoria per valori diversi, in tempi diversi), che per la lettura del bit di informazione.

Al di là del tipo di supporto usato per la memoria, questa può essere classificata secondo le modalità di accesso sia in lettura sia in scrittura. Pertanto abbiamo la memoria a sola lettura o ROM (*Read Only Memory*); memoria ad *accesso sequenziale* come può essere quella che un nastro magnetico o di carta in cui i singoli bit sono letti sequenzialmente, memoria ad accesso casuale o RAM (*Random Access Memory*); etc. etc..

Quando la memoria si riferisce a piccole quantità di informazioni ed è utilizzata per leggere, scrivere e scambio di informazione facile e veloce prende nomi speciali. Il posto in cui viene immagazzinata la minima quantità di informazione, cioè un bit, si chiama *cella di memoria*, *registro ad un bit* o, se ci si riferisce ad una particolare realizzazione molto usata, *Flip-Flop*. Le celle di memoria possono essere organizzate in modo che è possibile leggere o scrivere più bit contemporaneamente (in parallelo). In questo caso vengono chiamate Registri. L'organizzazione di bit normalmente utilizzata è: l'insieme di 4 bit, chiamata *NIBLE*, di 8 bit chiamata *BYTE*, di 16 bit chiamata *WORD*. Ogni una di tale organizzazione è stata scelta e chiamata in modo speciale per rendere facile l'individuazione di certe situazioni. Per es. Il nible è stato definito perché è più facile leggere i numeri binari in codice esadecimale o decimale (a 4 a 4), il byte perché è il numero minimo di bit necessari per codificare un carattere alfa numerico. L'organizzazione a 16 bit = 2 byte è stata chiamata genericamente word (parola) sia per distinguerla dalle altre, che perché si è deciso di chiamare word l'organizzazione di bit a multipli interi di Byte (per distinguere le diverse organizzazioni si dichiara contestualmente il numero di Byte o di bit di cui è composta). Ai tempi odierni il Nible non è molto usato ed il byte viene considerato la minima quantità di memoria organizzata. I calcolatori digitali hanno la memoria, in cui scrivono o da cui leggono dati e istruzioni organizzata, generalmente, a byte. Questo significa che per leggere o scrivere un'unità di memoria viene letto o scritto un intero byte in parallelo. Il byte, nel caso della lettura, viene trasferito in un registro ad 8 bit, e da qui viene letto se necessario il singolo bit. Per distinguere i singoli byte di cui la memoria è composta (può essere anche di diecine o centinaia di mega byte) le singole posizioni di memoria vengono individuate per indirizzo. In altri termini è come in una città in cui c'è una sola strada, con gli ingressi agli stabili disposti in un solo lato. Per la distinzione dei singoli ingressi basta un numero che ne indica la posizione sequenziale o indirizzo. L'indirizzo è utilizzato da particolari circuiti di instradamento della richiesta di lettura o scrittura permettendo l'accesso al singolo byte. All'interno del registro in cui viene trasferita l'informazione, per l'individuazione dei singoli bit si usa l'indirizzo di bit; avviene la stessa cosa (limitata al numero di posizioni nel registro) che avviene per l'individuazione delle abitazioni interne di uno stesso palazzo.

PROBLEMI

P1.1) Scrivere i primi 20 numeri decimali in base 3

P1.2) Sommare e moltiplicare i seguenti numeri nella base indicata senza convertirli in decimale.

a) $(1230)_4$ e $(23)_4$; b) $(135.4)_6$ e $(43.2)_6$; $(367)_8$ e $(715)_8$; $(296)_{12}$ e $(57)_{12}$

P1.3) Convertire il numero decimale 250.5 alle basi 3, 4, 7, 8 e base 16.

P1.4) Convertire i seguenti numeri binari in decimale 10.10001; 101110.0101; 1110101.110; 11011101.111

P1.5) Convertire i seguenti numeri dalla base data alle basi richieste:

a) decimale 225.225 a	binario,	ottale, e	esadecimale
b) binario 11010111.110 a	decimale,	ottale, e	esadecimale
c) ottale 623.77 a	decimale,	binario e	esadecimale
d) esadecimale 2AC5.D a	ottale,	decimale,	binario

P1.6) Rappresentare il numero decimale 8620 in (a) BCD, (b) eccesso di 3, (c) in 2421 e (d) in binario.

P1.7) Per ogni uno dei codici pesati (a) 3,3,2,1 e (b) 4,4,3,-2 determinare tutte le possibili tabelle per le cifre decimali in modo tale che il complemento a 9 di ciascuna cifra decimale si ottenga cambiando gli 1 con 0 e gli 0 con 1.

P1.8) Usando i pesi 5421, ottenere il codice binario pesato per i numeri ad una cifra in base 12.

P1.9) Determinare il bit di parità dispari che si genera quando il messaggio è una delle 10 cifre decimali in codice 8, 4, -1, 1.

P1.10) Ottenere un codice binario che rappresenti tutte le cifre in base 6 in modo tale che il complemento a 5 è ottenuto cambiando gli 1 con gli 0 e gli 0 con 1 del codice.

P1.11) Scrivere il vostro nome e cognome in un codice ad 8 bit derivato dal codice ASCII e aggiungendo come ottavo bit (posizione più significativa) un bit di parità pari. Separare il nome dal cognome con uno spazio.

P1.12) Considerare il seguente contenuto di un registro a 12 bit: 0101 1001 0111. Trovare il valore numerico da esso rappresentato interpretandolo come (a) tre cifre decimali in BCD, (b) Tre cifre decimali in eccesso di tre, e (c) tre cifre decimali in codice 2,4,2,1.