LEZIONE 13/10

-- continuazione lezione sulle stringhe --

Approfondimento lettura e scrittura dei caratteri.

```
scanf("%d", &c); // lo scan usa il "<paramatro>" per capire cosa leggere
printf("stampa il carattere %c e il numero del carattere %d", c, c);
    // in questo caso stampa il primo printf con il carattere, mentre il secondo restituisce il codice ASCII del carattere, cioè il valore intero del carattere.
    // il codice funziona anche con il carattere \n e poi restituisce anche il suo valore ASCII del carattere \n.

    // ciclo while per stampare tutto, cioè stampa fino al valore nullo \0
i = 0;
while (stringa[i]){// questo avviene perchè ci sarà un while (0) alla fine. In C, 0=Falso
    printf("valore stringa %c", i);
    i++;
}

printf(stringa); //funziona, ma ci si aspettava che si rompesse. Per funzionare allora funziona, ma se si valuta la correttezza del linguaggio e formale allora il comando è sbagliato. In questo caso il compilatore lascia un "warning" cioè avvisa, ma compie anche un best-guest, cioè inserisce quello che ritiene più giusto.
```

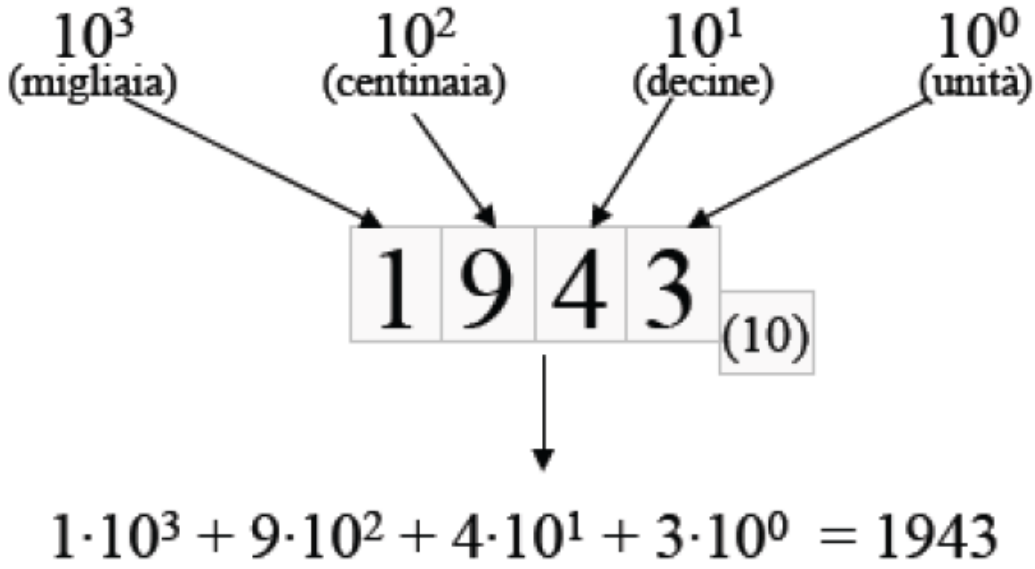
OPERAZIONI E CONVERSIONE TRA BASI DIVERSE:

- base decimale: uso di 10 cifre, 0-9.
- base binaria: rappresenta il modo fisico in cui la memoria e la cpu lavorano.
    - mi perdo qualcosa se rappresento in base binaria qualcosa prima rappresentato in base 10.

decimale	esadecimale	ottale	binario
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	8	10	1000
9	9	11	1001
10	A	12	1010
11	B	13	1011
12	C	14	1100
13	D	15	1101
14	E	16	1110
15	F	17	1111
16	10	20	10000
17	11	21	10001
18	12	22	10010
19	13	23	10011
20	14	24	10100

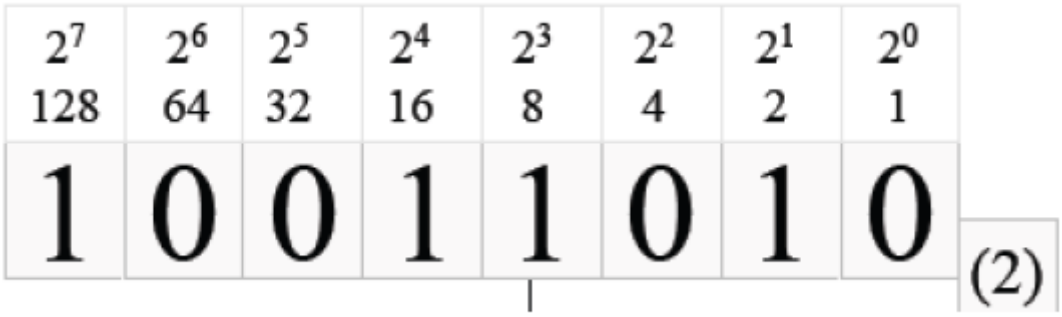
sistema decimale:

- si usano le dieci cifre
- valore dipende dalla posizione (notazione posizionale)
- cifre più significative verso sinistra, destra meno significative



- il pedice (10), indica la natura del numero, decimale in questo caso.

sistema binario

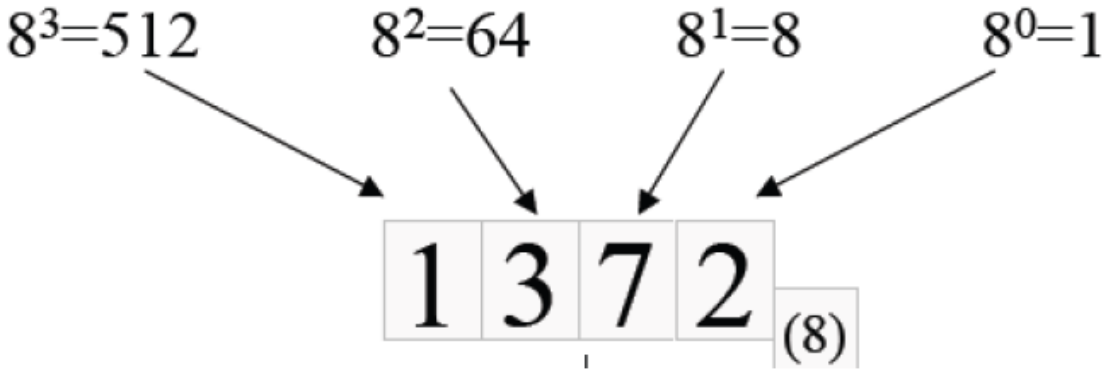
- si usano solo due cifre (1-0)
- valore posizionale della cifre
- più significative andando verso sinistra (More Significal BIt), il valore di destra è meno significativa (Less Significal Bit)
- 

- come prima il pedice (2) indica la natura binaria del numero.
- per passare da base binaria a decimale:

$$1\cdot2^7 + 0\cdot2^6 + 0\cdot2^5 + 1\cdot2^4 + 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 0\cdot2^0 =$$

$$128 + 16 + 8 + 2 = 154_{(10)}$$

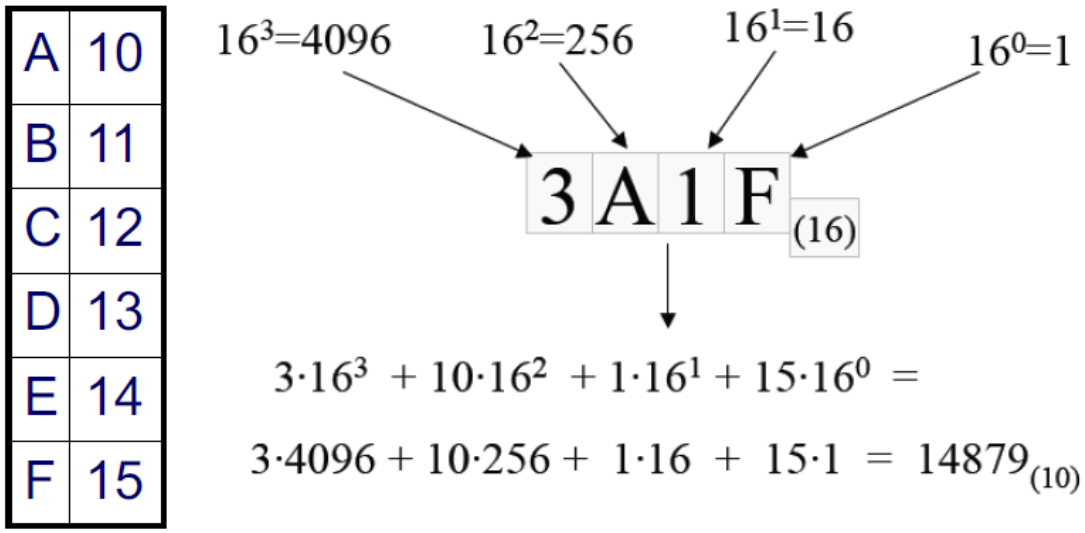
sistema ottale:



$$1 \cdot 8^3 + 3 \cdot 8^2 + 7 \cdot 8^1 + 2 \cdot 8^0 =$$

$$1 \cdot 512 + 3 \cdot 64 + 7 \cdot 8 + 2 \cdot 1 = 762_{(10)}$$

sistema esadecimale:

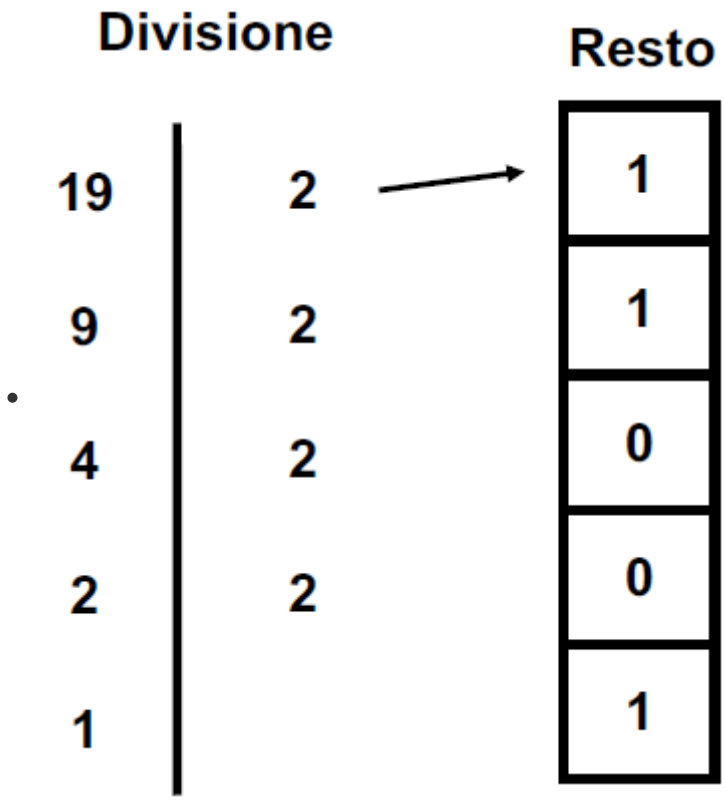
- uso di 16 basi: 0-9 + ABCDF
- 

- per trasformalo in base decimale l'algoritmo è sempre la conversione sul concetto di base posizionale esponenziale.
- $$3 \cdot 16^3 + 10 \cdot 16^2 + 1 \cdot 16^1 + 15 \cdot 16^0 =$$

  $$3 \cdot 4096 + 10 \cdot 256 + 1 \cdot 16 + 15 \cdot 1 = 14879_{(10)}$$

Conversione dei valori:

- dato valore in base decimale a valore binario:



- in questo algoritmo il più significativo è l'ultimo risultato, che va stampato per primo
- la conversione restituisce il valore (10011)

-- casissimo alla fine, durante la dimostrazione --

dimostrazione dell'algoritmo da decimale a binario e viceversa.

```
// ultimo file in c99, è presente il tipo di variabile bool
#include <stdbool.h>
_Bool = True;
```

anche la conversione da decimale a ottale avviene con un algoritmo molto simile a quello visto prima, si raccolgono i resti della divisione, poi si raccolgono i resti.

algoritmo da decimale a base esadecimale è anche questo identico a quelli visti prima.