

# **Modulo:** Approfondimenti sui Sistemi Aritmetici di un computer: tipo intero [P2\_02]

## **Unità didattica:** Sistema Aritmetico Intero [3-AT]

### **Titolo:** Il Tipo Intero nei linguaggi di programmazione

#### Argomenti trattati:

- ✓ Sistemi Aritmetici di un computer e tipi numerici in C
- ✓ Finitezza della rappresentazione in memoria degli interi
- ✓ Aritmetica modulare
- ✓ Tipi di rappresentazione in memoria dei numeri interi

**Prerequisiti richiesti:** aritmetica binaria, operatori binari

# SISTEMA ARITMETICO (S.A.)

=

- criterio di rappresentazione in memoria dei dati di tipo numerico
- definizione delle operazioni aritmetiche

La finitezza della memoria impone che gli insiemi numerici rappresentabili nei computer siano **finiti** e **discreti**

**SISTEMA  
ARITMETICO**

**S.A. INTERO (I)**

**S.A. REALE**

**FLOATING POINT (F)**

# Principali tipi di dati numerici nel linguaggio C

## Intero (I)

**char**  
**short**    **short int**  
**long**    **long int**  
          **int**  
(signed    unsigned)

## Reale Floating Point (F)

**float**  
**double**  
**long double (...)**  
  
**complex**

# **SISTEMA ARITMETICO INTERO**

## Esempio: provare...

```
#include <stdio.h>
void main()
{short s; unsigned short u;
 printf("immetti intero ...");
 scanf("%hd",&s); u=s;
 printf("signed = %d\nunsigned = %u\n",s,u);
}
```

immetti intero ...13

signed = 13

unsigned = 13

**ok!**

immetti intero ...-13

signed = -13

unsigned = 65523

?

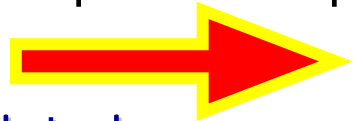
# Come spiegare i risultati?

# Rappresentazione in memoria degli interi (**Tipo Intero**)

Si basa sulla *rappresentazione binaria posizionale*:

es.  $10011_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19_{10}$

ed **occupa un numero finito  $n$  di bit**. Con  $n$  bit disponibili, è possibile rappresentare solo  $2^n$  interi diversi.



Esiste un **minimo  $i_{\min}$**  ed un **massimo  $i_{\max}$**  tra gli interi rappresentabili oltre i quali (per definizione) si verifica un **overflow di intero**.

Su  $n$  bit sono rappresentabili, **in base 2**, solo i numeri naturali  $0, 1, 2, \dots, 2^n - 1$

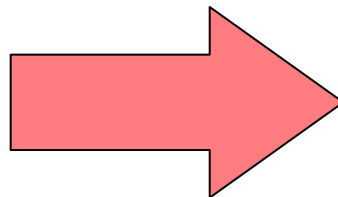
**Esempio:**

**$n=4$**

**$2^n - 1 = 15$**

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

**n numero finito di bit**  
per la rappresentazione



**Aritmetica binaria**  
**modulo  $m=2^n$**

## Cos'è l'aritmetica modulo $m$ ?

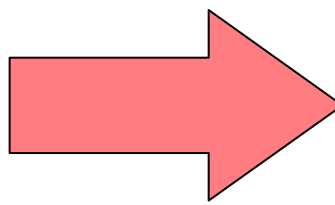
L'insieme  $\mathbb{N}$  dei numeri naturali viene proiettato nell'insieme dei numeri  $\{0, 1, 2, \dots, m-1\}$ :

$$\Phi_m : k \in \mathbb{N} \longrightarrow \Phi_m(k) = k_{\text{mod } m} \in \{0, 1, 2, \dots, m-1\}$$

è il resto della divisione intera per  $m$

I numeri  $\{0, 1, 2, \dots, m-1\}$  sono i rappresentanti di **classi di equivalenza**, ciascuna contenente tutti i numeri che forniscono lo stesso resto nella divisione intera per  $m$ .

**numero finito  
di bit**



**Aritmetica binaria  
modulo  $m=2^n$**

**Esempio  $m = 4$**

$$\Phi_4 : k \in \mathbb{N} \longrightarrow \Phi_4(k) = k_{\text{mod } 4} \in \{0, 1, 2, 3\}$$

$\forall h, k \in \mathbb{N} : h_{\text{mod } 4} = k_{\text{mod } 4}$  si scrive  $(h \equiv k)_{\text{mod } 4}$

**classi di equivalenza**

$$[0] = \{0, 4, 8, 12, 16, 20, \dots\}$$

$$[1] = \{1, 5, 9, 13, 17, 21, \dots\}$$

$$[2] = \{2, 6, 10, 14, 18, 22, \dots\}$$

$$[3] = \{3, 7, 11, 15, 19, 23, \dots\}$$

```
m=4;  
h=25; mod(h,m)  
ans =  
      1
```

```
h=35; mod(h,m)  
ans =  
      3
```

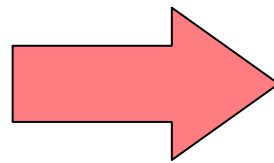
```
m=4;  
h=25; rem(h,m)  
ans =  
      1
```

```
h=35; rem(h,m)  
ans =  
      3
```

**MATLAB**



**numero finito  
di bit**



**Aritmetica binaria  
modulo  $m=2^n$**

**Esempio C: tipo char  $\Leftrightarrow m=2^8=256$**

$$\Phi_{256} : k \in \mathbb{N} \longrightarrow \Phi_{256}(k) = k_{\text{mod } 256} \in \{0, 1, 2, \dots, 255\}$$

```
void main()  
{unsigned char k;      /* tipo char occupa 8 bit */  
                        /* aritmetica modulo 256 */  
  unsigned short h, mod, m=256; /* 256 = pow(2,8) */  
  h=...;      k=(char)h;      mod=h%m;  
  printf("short h = %d,\tchar k = %d,\th%%m = %d\n\n",h,k,mod);  
}
```

**output**

short **h** = 250, char k = 250, h%m = 250

short **h** = 257, char k = 1, h%m = 1

**numero finito  
di bit**

**Aritmetica binaria  
modulo  $2^n$**

Es.:  $n=4$

$$\begin{array}{r} 6+ \\ 11= \\ \hline 17 \end{array}$$

**$17 \equiv 1 \pmod{16}$**

0	1	1	0
1	0	1	1
<del>1</del>	0 <sup>r</sup>	0 <sup>r</sup>	1

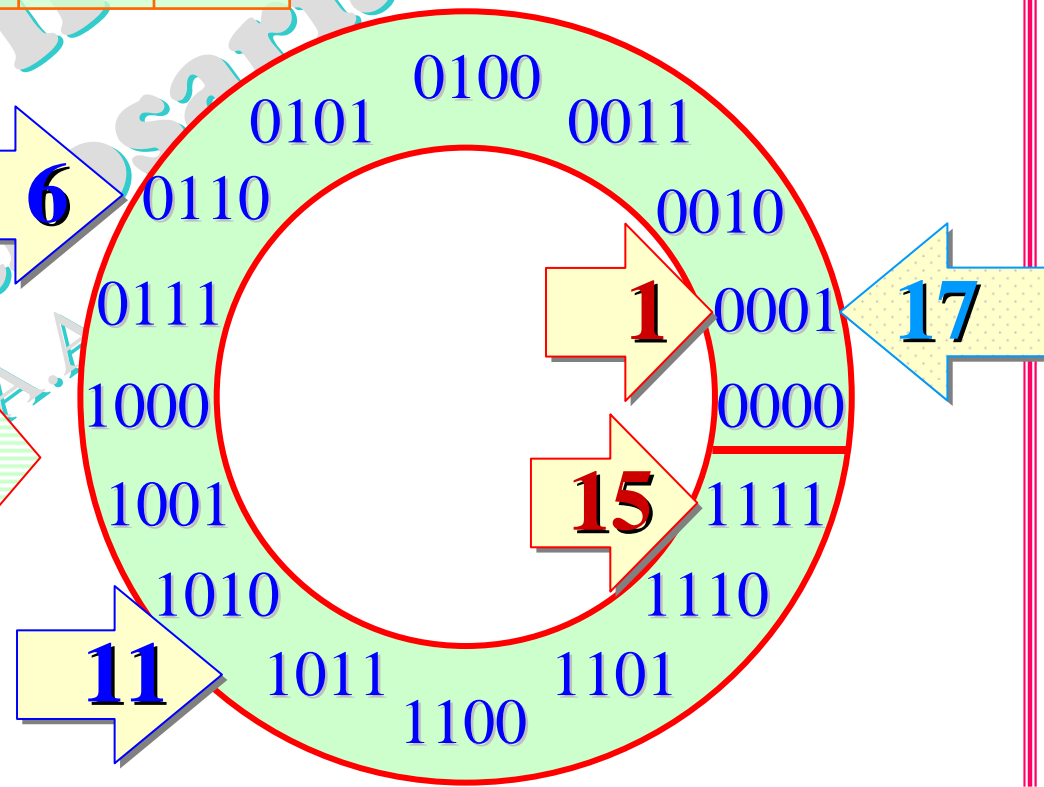
+  
=

Aritmetica mod. $2^n$ :  
aritmetica ..."facile"  
da realizzare su un  
numero finito di bit

Numeri Naturali N

0	0000
1	0001
2	0010
3	0011
...	...
13	1101
14	1110
15	1111

I numeri si  
dispongono  
su una  
circonferenza





# Tipi di rappresentazione



## Rappresentazione per segno e modulo:

usata per il **campo mantissa** di un numero reale floating-point.



## Rappresentazione per complemento a 2:

usata per memorizzare i **numeri interi** con segno.




## Rappresentazione biased:

usata per il **campo esponente** di un numero reale floating-point.

# Rappresentazione degli interi per segno e modulo

Per rappresentare **interi** ( $\in \mathbb{Z}$ ) **simmetricamente negativi e positivi**, la soluzione più semplice consiste nell'usare uno dei bit per il segno e gli altri per la rappresentazione in base 2 (**rapp. per segno e modulo**).

**Esempio:**  $+5_{10} \longrightarrow (0, 101)_2$        $-5_{10} \longrightarrow (1, 101)_2$



Gli interi rappresentabili su **n** bit, per segno e modulo, sono

$$\underbrace{-2^{n-1}+1, \dots, -2, -1}_{2^{n-1}-1}, \underbrace{-0, +0}_{2 \text{ zeri !!!}}, \underbrace{1, 2, \dots, 2^{n-1}-1}_{2^{n-1}-1}$$

ed il **range**  $[i_{\min}, i_{\max}] = [-(2^{n-1}-1), 2^{n-1}-1]$  è simmetrico.

Metodo di rappresentazione **non adeguato** per l'**aritmetica modulo  $2^n$**

Nelle altre due rappresentazioni degli interi:

- ✱ rappresentazione per complemento a due
- ✱ rappresentazione “biased” con bias B (eccesso B)

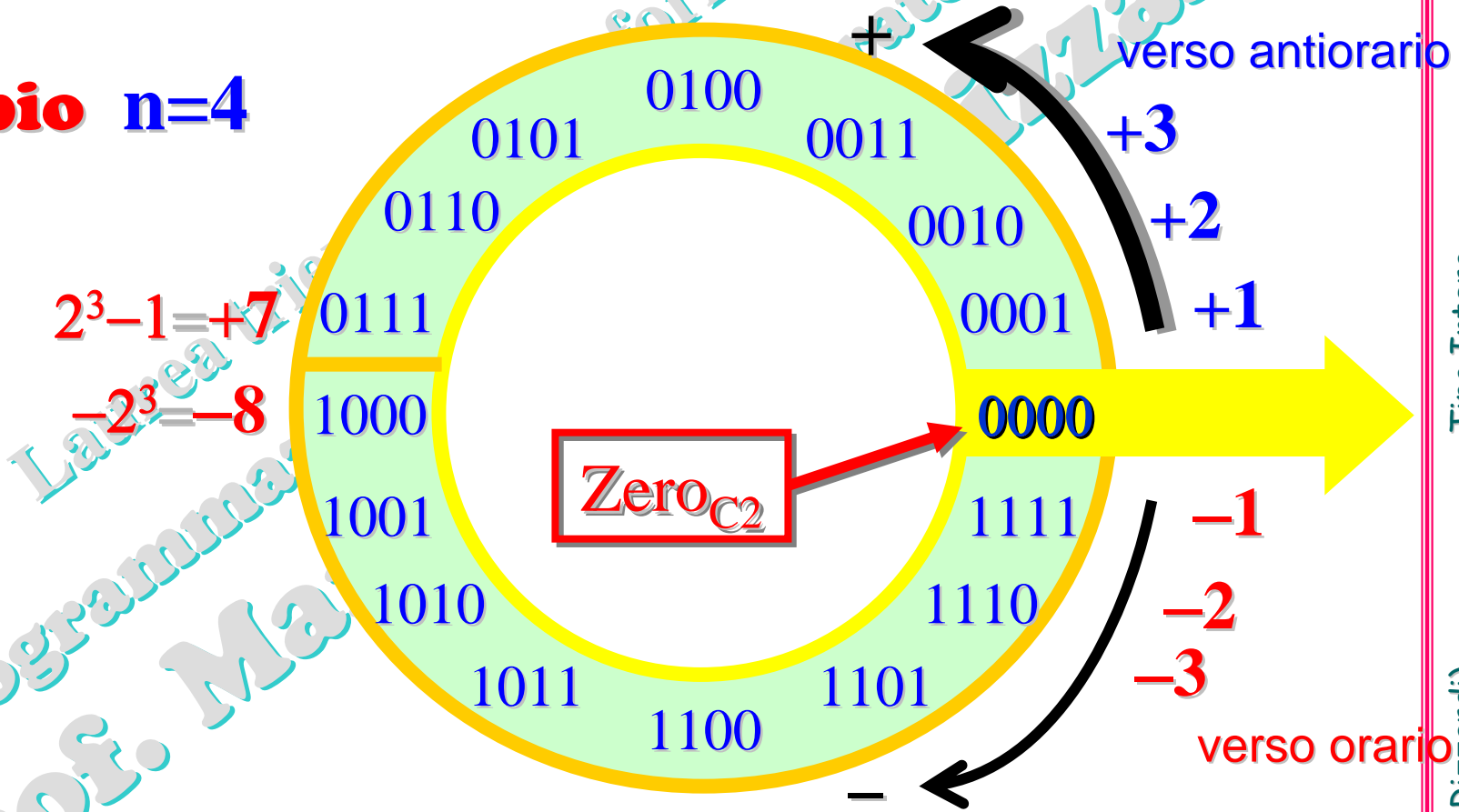
tutti i valori interi (rappresentabili) hanno un'unica rappresentazione (anche lo zero!) mediante sequenze consecutive di **n** bit nell'aritmetica modulo  **$2^n$** : quindi il range non sarà simmetrico.

Quello che cambia tra le due rappresentazioni è solo la corrispondenza tra le sequenze di **n** bit di  **$\{0, 1, \dots, 2^n-1\}$**  e gli interi  **$\{i_{\min}, i_{\min}+1, \dots, i_{\max}-1, i_{\max}\}$** .

# Rappresentazione per complemento a due

Nella **rappresentazione per complemento a due (C2)** lo zero è associato alla sequenza di **n** zeri.

**Esempio n=4**



ed il **range**  $[i_{\min}, i_{\max}] = [-2^{n-1}, 2^{n-1} - 1]$  non è simmetrico.

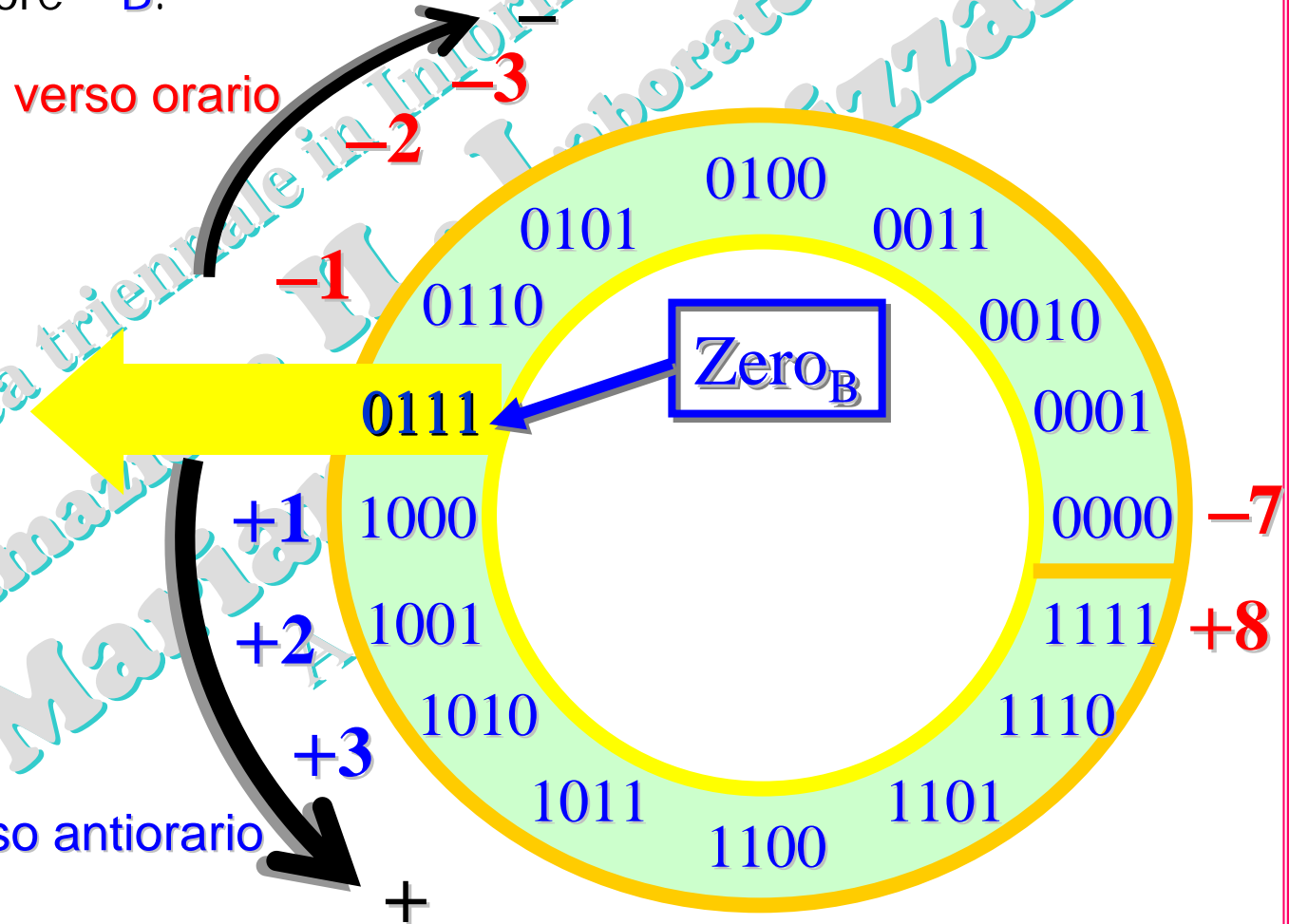


# Rappresentazione B-biased

Nella rappresentazione biased con bias  $B=2^{n-1}-1$  (rappresentazione eccesso B) il bias B rappresenta il valore zero mentre gli n zeri rappresentano il valore  $-B$ .

## Esempio

$n=4, B=7$



ed il **range**  $[i_{\min}, i_{\max}] = [-(2^{n-1} - 1), 2^{n-1}]$  non è simmetrico.



**n=3 bit**

## Esempio

configurazioni di bit disponibili

bit	Segno e modulo	Complemento a 2	Biased
000	<b>+0</b>	<b>0</b>	$i_{\min}$ -3 <b>-B</b>
001	<b>+1</b>	<b>+1</b>	<b>-2</b>
010	<b>+2</b>	<b>+2</b>	<b>-1</b>
011	$i_{\max}$ <b>+3</b>	$i_{\max}$ <b>+3</b>	<b>0</b>
100	<b>-0</b>	$i_{\min}$ <b>-4</b>	<b>+1</b>
101	<b>-1</b>	<b>-3</b>	<b>+2</b>
110	<b>-2</b>	<b>-2</b>	<b>+3</b>
111	$i_{\min}$ <b>-3</b>	<b>-1</b>	$i_{\max}$ <b>+4</b>

# Esempio

sequenza di 8 bit	hex	int. senza segno	segno e modulo	compl. a 2	biased B=127
0000 0000	00	0	+0	0	$i_{\min}$ -127
0000 0001	01	1	1	1	-126
:	...	:	:	:	:
0111 1110	7e	126	126	126	-1
0111 1111	7f	127	$i_{\max}$ 127	$i_{\max}$ 127	0
1000 0000	80	128	-0	$i_{\min}$ -128	1
1000 0001	81	129	-1	-127	2
:	...	:	:	:	:
1111 1111	ff	255	$i_{\min}$ -127	-1	$i_{\max}$ 128